UNIVERSITY OF Southampton

University of Southampton Research Repository
ePrints Soton

Copyright © and Moral Rights for this thesis are retained by the author and/or other copyright owners. A copy can be downloaded for personal non-commercial research or study, without prior permission or charge. This thesis cannot be reproduced or quoted extensively from without first obtaining permission in writing from the copyright holder/s. The content must not be changed in any way or sold commercially in any format or medium without the formal permission of the copyright holders.

When referring to this work, full bibliographic details including the author, title, awarding institution and date of the thesis must be given e.g.

AUTHOR (year of submission) "Full thesis title", University of Southampton, name of the University School or Department, PhD Thesis, pagination

http://eprints.soton.ac.uk

UNIVERSITY OF SOUTHAMPTON

Self-organising Agent Communities for Autonomic Computing

by

Mariusz Jacyno

A thesis submitted in partial fulfillment for the degree of Doctor of Philosophy

in the

Faculty of Engineering and Applied Science

Department of Electronics and Computer Science

February 2010

UNIVERSITY OF SOUTHAMPTON

ABSTRACT

FACULTY OF ENGINEERING AND APPLIED SCIENCE
DEPARTMENT OF ELECTRONICS AND COMPUTER SCIENCE

Doctor of Philosophy

by Mariusz Jacyno

Efficient resource management is one of key problems associated with large-scale distributed computational systems. Taking into account their increasing complexity, inherent distribution and dynamism, such systems are required to adjust and adapt resources market that is offered by them at run-time and with minimal cost. However, as observed by major IT vendors such as IBM, SUN or HP, the very nature of such systems prevents any reliable and efficient control over their functioning through human administration.

For this reason, autonomic system architectures capable of regulating their own functioning are suggested as the alternative solution to looming software complexity crisis. Here, large-scale infrastructures are assumed to comprise myriads of autonomic elements, each acting, learning or evolving separately in response to interactions in their local environments. The self-regulation of the whole system, in turn, becomes a product of local adaptations and interactions between system elements.

Although many researchers suggest the application of multi-agent systems that are suitable for realising this vision, not much is known about regulatory mechanisms that are capable to achieve efficient organisation within a system comprising a population of locally and autonomously interacting agents.

To address this problem, the aim of the work presented in this thesis was to understand how global system control can emerge out of such local interactions of individual system elements and to develop decentralised decision control mechanisms that are capable to employ this bottom-up self-organisation in order to preserve efficient resource management in dynamic and unpredictable system functioning conditions. To do so, we have identified the study of complex natural systems and their self-organising properties as an area of research that may deliver novel control solutions within the context of autonomic computing.

In such a setting, a central challenge for the construction of distributed computational systems was to develop an engineering methodology that can exploit self-organising principles observed in natural systems. This, in particular, required to identify conditions and local mechanisms that give rise to useful self-organisation of interacting elements into structures that support required system functionality. To achieve this, we proposed an autonomic system model exploiting self-organising algorithms and its thermodynamic interpretation, providing a general understanding of self-organising processes that need to be taken into account within artificial systems exploiting self-organisation.

Contents

List of Figures

List of Tables

Acknowledgements

Many thanks go to my supervisors, Seth Bullock, Terry Payne and Michael Luck, for their guidance and help; my colleagues from both IAM and SENSe group that offered an interdisciplinary insight into my work and inspired to pursue my own fascinations. Finally, I wish to thank my parents for their support and love that gave me strength and encouraged to never give up.

A smart machine will first consider which is more worth its while: to perform the given task or, instead, to figure some way out of it.

— Stanislaw Lem, The Futurological Congress, 1971

Chapter 1

Introduction

Modern software systems are among the most complex human artefacts [25, 38]. This is evident in today's information systems that depend on so many modules, sources of data, network connections, input and output devices that it has become very difficult to predict or control their interactions. This observation is manifested and emphasised through the ongoing evolution from standalone computer applications to systems composed of a large number of distributed and interacting components [118, 21]. Although such modular architectures offer opportunities to tackle system complexity by decomposing the overall structure into specialised components, they also present challenges in the maintenance of reliable and predictable operation in changing conditions.

As a result, it is not surprising that over seven years ago IBM released a manifesto arguing that the main obstacle to further progress in the IT industry is a looming software complexity crisis. To respond to this, an Autonomic Computing initiative was announced with the vision of systems capable of regulating their own functioning. Here, large-scale infrastructures are assumed to comprise myriads of autonomic elements, each acting, learning or evolving separately in response to interactions in their local environments [133]. The self-regulation of the whole system then becomes an emergent product of local adaptations and interactions between system elements.

Although multi-agent systems research offers many suitable models for realising this vision [23], the functioning of complex IT systems may become too complicated to be predicted by 'divide and rule' analysis [6, 133], and controlled through existing coordination mechanisms. One reason for this is the existence of highly non-linear global system dynamics, resulting in emergent system behaviour that is difficult to understand and predict [35, 41, 131]. Furthermore, recent research in pervasive, ubiquitous and grid frameworks argues that the computational landscape is beginning to significantly change [101, 20]. This is caused by the departure from closed and deterministic environments, requiring systems to function in dynamic and unpredictable conditions. For this reason, they must be modeled as open systems, able to change their composition at run-time

[57]. Consequently, whilst multi-agent architectures offer a natural decomposition into autonomous entities, autonomic systems may demand more robustness and agility than the existing agent-based control techniques may currently offer.

For this reason, the tenets of IBM's Autonomic Computing manifesto are inspired by the phenomenon of homeostatic control that has evolved over millennia to maintain system equilibrium in biological organisms. The observation of natural systems (for example ecosystems, insect colonies, complex adaptive systems) in biology [105, 74, 34] and physics [56, 39], suggests that they have developed internal control mechanisms, allowing them to organise and adapt, relying only on local interactions between the system components. This self-organisation process [39, 107] depends on certain principles that have begun to be understood over the past few years [38], leading to the emergence of biologically inspired control mechanisms that are decentralised and robust, yet still restricted to domain specific applications (for example ant path planning for Internet traffic routing in telecommunications networks, industrial manufacturing control or directing the behaviour of robots) [94, 70, 77].

Encouraged by these preliminary results, the development of Autonomic Computing has initiated interdisciplinary research offering alternative means of controlling distributed computational systems. In such a setting, a central challenge is to develop an engineering methodology that can exploit self-organising principles observed in natural systems for the construction of effective autonomic software infrastructures.

However, for this approach to become useful, we must first understand the underlying principles of natural self-organisation, and, in particular, how to apply these principles in the context of open IT systems. This is the main issue addressed in this thesis, focusing on the application of self-organisation to distributed networked computer systems engineering, where the large-scale functional structure of the system is allowed to emerge from the interactions between system components at the local scale.

1.1 Architecture of Modern IT Systems

To understand the increasing difficulty at controlling modern IT systems, below we outline the general building blocks of such infrastructures and explain the origins of their management complexity.

An IT system can be thought of as a collection of computing resources tied together to perform a specific set of functions [58]. Depending on the granularity and scope, both an individual server as well as a microprocessor containing varying integrated elements on a single chip are rightful system representatives. From an autonomic system point of view, these lower-level systems combine to form larger systems: multiprocessors combine to form servers, servers combine with storage devices and client or access devices to form

networked systems, and so on.

In this thesis the focus is on system architectures that comprises a number of networked servers. These hardware machines are considered as computational nodes on which e-business application servers are deployed. The nature of modern application servers (based eg., on Java 2 Enterprise Edition [100] architecture) allows developers to abstract away from physical connectivity of the machines on which servers are running and thus form a homogeneous software environment comprising a network of interacting platforms (deployment containers [124]), each hosting a number of applications. Below we outline some of the important building blocks of IT infrastructures that are of our interest.

1.1.1 Services

The functionality of IT infrastructure is provided by server nodes that offer limited quantity of resources (eg. software programs, hard drive storage, or CPU power) to users that request them. The access to these distributed resources is facilitated through software elements (service providers) that are responsible for the provision of requested resources. For this purpose, it is assumed that each server node hosts a single service provider that, in turn, offers the access to node's resources in the form of a service.

1.1.2 Service Registries

Services offered by a system are distributed over a number of networked physical machines. As a result, access to any of these applications requires possession of an appropriate endpoint address, defining the exact networked machine handling the requested service. However, large IT infrastructures are characterised by the dynamism reflected in constant system reconfiguration, involving the introduction of new servers or replacement of existing ones in order to cope with hardware failures and varying service demand. Because of this, resources, as well as applications, are migrate to different physical locations, rendering already known endpoint locations stale. To avoid disinformation about changing system structure, Service Oriented Architectures (SOA) [75, 67] assume the existence of service registries, which are dedicated system components providing information about other resources within the system [3]. Based on this, it is assumed that if any service is being migrated to a different physical address or introduced/removed from the system, this change would be reflected within a registry service. This facilitates dynamic service selection, where service consumers are informed about services upon interaction with the registry without the need to know *a priori* about the physical locations of these services [87].

1.1.3 Service Configuration Through Switching

Although application servers allow for a number of different services to be deployed and provisioned at the same time, due to security restrictions, it is often assumed that each server will provide only one type of service with the number of simultaneous provisions controlled by the number of available resources. If demand requirements change and the server is demanded to offer a different service, there is a certain time-lag associated with the current service undeployment and redeployment of a new one. This procedure of changing the offered service is often termed *service switching*.

1.1.4 Power Management

Maintaining IT infrastructure with a large number of servers introduces a substantial energy cost required to maintain system servers in operation. As observed in [98] the rapidly rising cost and environmental impact of energy consumption in these systems has become a multi-billion dollar concern globally. As a result, efficient power management has become a highly desirable, if not critical, characteristic of any large scale IT infrastructure.

To achieve this, individual server nodes are allowed to perform local power management adjustments involving alteration of CPU cycles, disabling of unnecessary server components (eg. additional CPU) or the eventual temporary shut-down of the node. It is assumed that these power management activities need to be continuously evaluated during system operation such that the host of servers that are kept on-line offers sufficient amount of resources to satisfy the changing demand that is imposed by the infrastructure users.

1.1.5 Service Provisioning

The complicated process of provisioning system resources demands a level of abstraction sufficient for non-expert users to be able to utilise services offered by the system. This is achieved by specialised components (business components [72]) which, though not providing any resources, are responsible for automating interaction with service providers and provisioning offered by them resources. These components, considered as *service consumers* [87], exist at the interface between users (or other services) and the service providers, and are responsible for:

- intercepting resource requests from users/clients;
- locating existing and available resources; and
- reliably providing execution results.

1.1.6 Physical Limitations

As a result of running on hardware components, the computation performed by the system is subject to the following constraints:

- interaction between system elements takes time and is subject to failure due to the distribution of system services over different physical nodes that may lose connectivity or have insufficient bandwidth [122];
- there is a limited number of requests that may be simultaneously served during each service provisioning (the number of how many simultaneous service executions may take place is often controlled through the maximum number of instances that the system is configured to handle in parallel [100]);
- execution of services is not instantaneous but takes time [5];

1.2 IT System Management Problems

Given the above defined general building blocks of modern IT systems, let us now outline key management duties that are involved whilst preserving correct operation of such infrastructures and the possible problems that may arise during such process.

In this thesis we are assuming that the the system is an open infrastructure that is controlled by a single infrastructure provider. This means that all services offered by the system (eg., calendaring, e-mail or photo applications) are owned by this infrastructure provider and thus all service providers (offering particular system resources) act so as to contribute to the overall system welfare. The defined in this way infrastructure is an open computational environment accessible by external users that aim to satisfy their individual service allocation requests. These users demand instantaneous access to system resources (eg., calendaring, e-mail or photo application) and thus the system has to deliver them in an on-demand manner by identifying available service providers that are free and capable to offer demanded services.

Under these conditions, the general aim of the infrastructure provider is to maximise the usage of the offered by him services and, at the same time, minimise the infrastructure running costs. Among a number of difficulties that may arise during system operation, it has been observed [60, 41, 61] that the following three ones are considered to have a direct impact on its functioning cost and thus *efficiency*:

1. Both, infrastructure users and as well as system servers will demand (and offer) various amounts of resources. Consequently, inappropriate allocation of user requests to service providers that have less (or more than required by the user)

resources may either overutilise or underutilise particular system servers. This unbalanced distribution of resources would decrease satisfaction of utilising them users (since overutilised servers may offer degraded quality of the service), as well as prevent maximal usage of the system services because resources of underutilised servers would not be used to their full extent.

2. Infrastructure users may have different goals and thus be interested in allocating various service types. Inappropriate configuration of the type of services that are being currently offered by the system with respect to the service interests imposed by users will result in loss of potential customers as well as additional maintenance cost of servers that were wrongly configured and hence unused. This, in turn, generates further maintenance costs in the form of a wasted energy needed to maintain these servers in an on-line state.

3. The number of users as well as resource quantities they demand may change during the system operation. Whilst keeping all servers on-line will generate substantial energy cost [98] and may generate a surplus of unused resources, insufficient number of available on-line system resources might introduce shortage of available on-line resources. Both configurations are thus inefficient and should be avoided during system operation.

If the IT infrastructure operated under deterministic conditions, eg., the demand imposed by users would remain static and unchanged over the system operation, and the system would be closed and hence its internal configuration would remain static, all three aforementioned problems could be easily avoided through correct and static configuration of the individual servers. For example, if the infrastructure provider knew beforehand how many users would require an e-mail client application and how many would utilise instant messenger application, it could configure the servers appropriately such that the system would achieve optimal efficiency at the lowest management cost. Furthermore, if such situation would never change the system, in principle, could operate at this optimal configuration forever.

However, as we have already suggested, and will more closely investigate in the next section, modern IT systems are open and dynamic, where the number of users as well as their interests may change over time. Furthermore, the system is prone to failures, software upgrades and addition of new elements that prevent its structure to remain static. As a consequence, management of such systems cannot be realised through static configuration but represents a non-trivial challenge involving the continuous adjustment and reconfiguration of it internal elements to represent the best response to the prevailing conditions.

This leads us to the three general management issues that we will focus on in this thesis and attempt to provide decision-making mechanisms capable to automate control over them:

1. **Load-balancing**

   Resource allocation requests arriving to the system should be evenly and fairly distributed across a population of suitable service providers. Such distribution of requests, and thus load on particular providers, should prevent situations in which certain providers become underutilised at the expense of others being overutilised and thus unable to provision services to service consumers requesting them.

2. **Service provisioning**

   The population of service providers should offer services that are currently demanded by infrastructure users such that the amount of demanded resources is matched by the supply.

3. **Power management**

   The system has to be configured in a manner that minimises costs involved in maintaining unused servers in an on-line mode, thus limiting the energy cost required to maintain the whole infrastructure.

So far, the management over these three functions has been mostly devolved to human administrators. However, there are certain characteristics of modern IT systems that render this approach unsuitable and unprofitable. We discuss these issues in the next section.

1.3 The Origins of Management Complexity

Modern IT infrastructures are undoubtedly state-of-the-art artefacts, integrating the most advanced software engineering techniques such as object oriented programming, networking, persistence and web-services [72, 87]. Their distributed nature, functional decomposition into resource providers, consumers, information brokers and finally multi-layered architecture, represent high-end standards for modern software development [75].

Despite these advantages, it has been recently observed that the management costs inflicted by these infrastructures set their wide applicability and further adoption in question [42, 1]. This issue, already approached by the main software vendors such as IBM, SUN and HP [60, 44], implies that due to properties introduced by these applications, software systems engineering has reached its limits, beyond which novel approaches for the construction and control of large-scale computational systems may be needed [58, 134]. The genesis of the problem stems from the combination of matured and previously successful control mechanisms with functioning conditions that were never previously experienced by software systems. The system management difficulties arising from the combination of the both above provided properties are explained below.

1.3.1 System Administration

The capabilities offered by these infrastructures come at a cost of tuning the behaviour of all system components in accordance with some system-level metrics of efficiency [1]. As we have outlined in the previous section (Section 1.2) such efficiency, whilst considering resource management, is dependent on the regulation of the three system-level functions: load-balancing, service provisioning and power management. This, within the distributed system comprising of a population of servers, involves their local adjustment and reconfiguration such that the global system efficiency is preserved.

A common technique to facilitate this for a system that is too complicated to be fully controlled by an individual or a group of administrators is to divide the management problem into parts and devolve responsibility of controlling each to a dedicated group of administrators [126, 1]. This top-down control technique, successfully applied in a wide range of decision-making problems, follows a simple *divide and rule* strategy which is facilitated by logically dividing the distributed system into administrative domains, each comprising a small enough group of system nodes small enough to be effectively managed by a team of administrators. The achievement of the global system efficiency thus depends on flexible control of system parameters (eg., type of deployed services on particular servers, server utilisation level or security access constraints) within dedicated administrative domains that, in concert, contribute towards global system performance.

However, the efficiency of the underlying *divide and rule* approach is defied as soon as a system becomes large enough to be considered profitable on a large scale. Paradoxically, the reason for this stems not from ill defined architecture, but from conditions within which these systems function, as considered next.

1.3.2 Functioning Conditions

Large-scale distributed infrastructures no longer operate in predictable and controlled conditions [133, 14]. As such, they challenge existing means of control and prove them inefficient in realistic deployment environments. Below we discuss functioning conditions that have a direct effect on the system behaviour and its management efficiency.

1.3.2.1 Interdependence

The true benefit of relying on these infrastructures stems from their distributed nature. As such, the infrastructures offer mechanisms that promote deployment of a system onto clusters comprising interconnected computational nodes. Application servers deployed on individual nodes do not operate in isolation but form an *organic* network of interconnected sub-systems, each sub-system offering a diverse set of services that are vital for the functioning of the whole infrastructure.

Both, increasing scale and interconnectivity catalyse dependence between individual subsystems, where functional code is distributed among different system nodes. As a result, control and parameter tuning involves a set of interdependent systems rather than a single and isolated machine. From a management perspective, this requires a group of administrators, tasked to control fractions of a networked environment to effectively, and in a timely manner, coordinate their actions and adjustments.

To give an example, consider a situation where one of the infrastructure administrators reconfigures a system node to offer a different service. This local adjustment requires coordination with other node administrators to verify whether there are no other services dependent on the current service, and if so, to take actions to relocate the existing service into other nodes and update service registries about the change. However, even though this operation has been successful, the performed change may cause other servers, offering the same type of service, to become overutilised and hence less responsive, leading to inefficient load-balancing and the formation of a bottleneck. Thus, not only is a timely response required, but so is the ability to predict the consequences of local changes on the global system level.

1.3.2.2 Physical Constraints

In contrast to single-machine operating systems, each distributed infrastructure is based on a physical network that connects interdependent nodes and defines limited routes through which data can flow between application servers. Although modern application servers preserve a sufficient level of abstraction for system engineers to 'forget' about the intricacies of the underlying physical topology whilst engineering distributed components, it has been recently observed that computation over the network has a significant impact on global system behaviour and thus requires more attention than has been given so far [45, 113, 48, 49].

In the infrastructure models discussed above, the underlying network of interconnected computational nodes will have a direct effect on constraining information flows between nodes and thus influence patterns of interactions between distributed system components. For example, the physical limitation of a finite bandwidth within the infrastructure may lead to the emergence of nodes that exploit high bandwidth at the cost of others utilising less of it and thus interacting to a lesser extent.

All these constraints imposed on the physical network and its constituents will result in the emergence of specific interaction patterns between system components distributed over these nodes. These patterns can be characterised as logical network layer, imposed over the physical one, where the more frequent the interactions between particular system components, the *stronger* the connectivity of relations between the network components involved. Given this, both physical and logical networks do not exist in

isolation, but exert feedback on each other. For example, the bandwidth allocation on a physical network may constrain the number of interactions of system components on a logical network. Also, increasing the frequency of interactions between particular system components, and thus bandwidth consumption, may lead to the emergence of strong connectivity between certain system nodes, and the possible reduction or loss of connectivity (or data trafficking) between others.

In such systems, these constraints do not remain static during system functioning. Due to changing conditions, both security policies and bandwidth limits for particular nodes might be modified along with the new specification of incoming hardware nodes or software running on them. Consequently, both logical and physical networks are more likely to continuously evolve, driven by their causal relations and changing conditions.

Understanding how these networks change over time might become a useful tool for predicting and controlling global system behaviour through exposed tunable parameters.

1.3.2.3 Openness

The interconnected nature of system nodes encourages openness of system components to interactions beyond the nodes they currently reside on. For example, services collocated on different nodes might form a coherent functional workflow requiring distributed invocation of each other. Consequently, the behaviour of services might become subject to modification by processes external to the node and thus beyond its direct control. This will require precise coordination of administrators controlling different nodes to avoid overutilisation or to react to node failures and system reconfiguration in a timely manner.

The system is in constant interaction with the distributed environment it is embedded within. For a set of communicating software components, the underlying environment is represented through machines referring to nodes, the physical network enabling distributed computation and the operating system hosting the application server. The local actions of system components may indirectly influence the functioning of such an environment, the results of which may in turn feed back into the system. For example, overutilisation of a particular node's CPU or a subset of distributed network bandwidth may negatively affect the functioning of processes that host the application server (hardware components, operating system, network routers). Furthermore, each such element might already host other processes, not related to the infrastructure, and thus may indirectly affect the behaviour of the system. For example, for operating systems (OS) on which application servers are hosted, the primary objective would be to preserve reliable and efficient functioning even at the cost of compromising the application server's performance or reliability. Also, physical interconnectivity might be exploited by other networked processes, stimulating its bandwidth. Finally, all the described com-

ponents are subject to unexpected failure that may propagate to the system and affect its performance [22, 21].

Since these infrastructures are dedicated to providing their functionality to servers requesting it, the system might be viewed as being under pressure driven by *load* and *demand* imposed by its users. In general, load defines the number of requests the system has to serve at the given time, whereas demand is the variety of offered functionality being utilised. When increased both put more pressure and thus work on the system. Since users are external to the system, the stress they impose on the system is not *a priori* known and may vary over time, requiring sufficient adjustment from the administrator side to cope with changes.

1.4 Autonomic Computing vision

To address the software management crisis, extensive research has been undertaken [58, 133, 91] in seeking alternative ways of engineering systems that are no longer fully dependent on skilled IT administrators but self-manage their vital functions. Due to the limited application of existing AI control techniques, this has resulted in a revolutionary shift of paradigm pioneered by a growing interest in natural complex systems and the emergence of alternative software architectures, in the form of multi-agent systems, supporting more flexible computation [134].

The observation of natural systems (for example ecosystems, insect colonies, complex adaptive systems) in biology [105, 74, 34] and physical systems [56, 39], suggests that they have developed internal control mechanisms, allowing them to organise and adapt, relying only on local interactions between system components. This self-organisation process [39, 107] depends on certain principles that have begun to be understood over the past few years [38], leading to emergence of biologically inspired control mechanisms that are decentralised and robust, yet still restricted to domain specific applications (for example ant path planning for Internet traffic routing in telecommunication networks, industrial manufacturing control or directing the behaviour of robots) [94, 70, 77]. Encouraged by these preliminary results, Autonomic Computing initiates interdisciplinary research offering alternative means of controlling distributed systems. In such a setting, a central challenge for the construction of distributed computational systems is to develop an engineering methodology that can exploit self-organising principles observed in natural systems for the construction of autonomic software infrastructures.

1.5 Aims of work

In this thesis we are interested in designing principled tools and methods for building autonomic computational systems that are capable of regulating their own functioning in a manner that maximises their utility. In particular, we aim to provide local decision-making mechanisms that are able to regulate service provisioning in dynamic and indeterministic conditions consistent with those experienced by modern and distributed service provisioning infrastructures.

To achieve this, we conduct an interdisciplinary study in which we apply self-organising techniques observed in natural complex systems to automate the control over the three key resource management functions: load-balancing, service provisioning and power management that we discussed in Section 1.2.

Throughout this study we assume that the system is open to structural change and operates in a dynamic environment. For this reason, the control mechanisms should be able to adaptively regulate the operation of the three aforementioned resource management functions in the following manner.

1. **Load-balancing**

   We expect the system to configure such that resource allocation requests arriving to it are evenly and fairly distributed across a population of suitable provider agents. Such distribution of requests, and thus load on particular providers, should prevent situations in which certain providers become underutilised at the expense of others being overutilised and thus unable to provision services to consumer agents requesting them.

2. **Adaptive Service Provisioning**

   The population of provider agents offering various services should be adaptive. Whenever demand for particular service types changes, we expect the population to reconfigure and adjust appropriately such that the supply of demanded services matches the demand imposed on the system.

3. **Power Management**

   Not only should the system respond and adjust to changing user interests, but also to the intensity of demand imposed by them. In situations where only a subset of available system resources is required to satisfy user demand, the remaining part of the system resources should remain in a power-saving mode, thus limiting the energy cost required to maintain the whole infrastructure.

Although there exists a number of criteria based on which the efficiency of an autonomic system can be evaluated (eg., system mean time response, system throughput)

we decided to evaluate system efficiency at achieving the three above resource management tasks. By doing so, we were able to explicitly focus on key resource management challenges that need to be addressed within modern IT systems and to understand how self-organisation can be applied to deliver them.

As an outcome of this study, we provide a physical interpretation of self-organisation phenomena that may be applied as a guideline for engineering artificial self-organising systems. The research contributions resulting from this study are provided in Section 1.5.1.

1.5.1 Research Contributions

In this thesis we have addressed the problem of applying self-organisation to preserve reliable control over the resource management within decentralised autonomic systems. To do so, we have studied characteristics of natural and open complex systems that exhibit self-organising features and showed that these can be understood as an outcome of certain physical laws existent within nature.

Based on this understanding, two computational models were provided to help us understand how the self-organising features of these complex systems can be applied within modern IT systems.

As an outcome of this study, we offer design principles for engineering self-organising autonomic software systems in which control over their functioning is decentralised and facilitated through local interactions and adaptations between its autonomous elements.

The research contributions achieved through this study are outlined below.

1.5.1.1 Decentralised Autonomic System Model

We offer a decentralised autonomic system model that is tasked to control resource management in dynamic and open environments. Using this model, we study how adaptive system-level response arises out of local interactions of individual system elements that employ for this purpose only simple stimuli-response mechanisms and local information exchange.

1.5.1.2 Self-organising Agent Communities

Understanding how local interactions between system elements give rise to certain global system dynamics becomes one of the major difficulties whilst engineering decentralised systems. In this thesis we address this problem by extending the system analysis to

include an intermediary level (*meso-level*), residing between the level at which the behaviour of individual agents is analysed (micro-level) and the level at which the collective response of the whole system is considered (macro-level).

At this intermediary level we identify system organisation into agent communities that are the key structures that support adaptive and efficient maintenance of the three system functions: load-balancing, adaptive service provisioning and power management.

1.5.1.3 Importance of Spatial Embeddedness for Self-organisation

We show the relevance of spatial embeddedness for achieving system self-organisation. To achieve such spatial property, the system agents are instructed to establish interaction topology according to which the peers that are closer to each other are more likely to interact and affect each other behaviour than the elements that are located at a greater distance. Only when such underlying interaction topology arises, can system components organise into globally efficient collective structures referred to as communities.

To realise such topology in our model we propose an *affinity* algorithm (described in detail in Section 5.3.3.6) the role of which during community formation is discussed in Section 9.2.2.3.

1.5.1.4 Thermodynamics in Computational System

Whilst thermodynamics is mostly related to the study of heat engines and provides basis for understanding how mechanical work can be extracted from the supplied energy, we stress the relevance of this discipline in achieving computational system self-organisation.

More specifically, we show that natural self-organising systems employ the same work extraction principles for achieving adaptive respnose and that understanding of conditions and mechanisms influencing such process will contribute towards principled engineering of adaptive and decentralised autonomic systems.

For this purpose we propose a set of design principles which, when incorporated into our model based on exemplary mechanisms, achieve system organisation that is analogous to the one existent in the natural systems. As such bottom-up organisation is driven by thermodynamic principles of self-organisation, we provide a thermodynamic interpretation of agent communities as artifacts which act analogously to thermodynamic engines but in stead of energy are fed with information that they transform into informational gradients from which useful work can be extracted.

1.5.1.5 System Stabilisation Through Positive and Negative Feedback

A system in which its constituents employ only locally available information whilst conducting resource allocation decisions is prone to instablilities or even chaotic response resulting from resource competition. In this study not only we show that organised system structures in the form of agent communities are critical in suppressing this pathological behaviour within a model where no central or hierarchical control is imposed, but also suggest how reliable control over the system can be provided based on positive and negative feedback loops that stabilise the formation and operation of such communities.

In particular, *affinity* algorithm (discussed in Section 5.3.3.6) is proposed that gives rise to a positive feedback responsible for triggering the agent community formation, whereas agent stress-based information inflow regulatory mechanism (discussed in Section 5.3.3.7) is introduced to facilitate negative feedback that preserves stable operation of such communities.

1.6 Overview of Document

The remainder of the document is organised as follows.

Chapter 2 presents a background material related to the research undertaken in the area of complex systems, outlining theory behind autonomic computing, multi-agent systems, complexity and self-organisation. In Chapter 3 a review of existing decentralised computational models is provided with both their strengths and weaknessess outlined. In Chapter 4 a simple decentralised model is presented and its self-organising capabilities studied in the context of load-balancing problem. Presented in this chapter work investigates the applicability of thermodynamic interpretation of global system behaviour arising as a product of simple and naive local algorithms employed by agents. A more advanced autonomic system model is presented in Chapter 5 and its efficiency in achieving load-balancing, adaptive service provisioning and power management is evaluated in the three following chapters: Chapter 6, Chapter 7 and Chapter 8. The thermodynamic interpretation of self-organisation observed within the model is provided in Chapter 9 that concludes with a set of design principles aimed at providing general guidelines for engineering artificial self-organising systems. The conclusion and further work is presented in Chapter 10

A number of publications has arisen as a result of the work discussed in this thesis. As these publications relate to particular topics discussed in several chapters, below we outline them together with a reference to the relevant chapter. A paper discussing thermodynamics of self-organisation and its application to computational systems [13] is a direct outcome of the literature review presented in Chapter 3. A work presented in [47] offers the analysis of bottom-up control approach of the first simplistic multi-agent

system model discussed in Chapter 4. Finally, the outcome of the work involving more realistic model, with particular attention to adaptive service provisioning problem that was presented in Chapter 7 was published in [65] and [66] [1].

[1]It is important to note that the autonomic system model used in the last two publications was an intermediate version between simple model presented in Chapter 4 and the more realistic model discussed in Chapter 5. To minimise the volume of the content presented in this thesis, we did not discuss the intermediate model architecture within this document, as it was provided within the two relevant publications

Chapter 2

Issues in Complexity Studies

2.1 Introduction

In this chapter we introduce general properties of complex natural self-organising systems. As these systems differ from the deterministic human made artifacts, we focus on the distinguishable features that need to be considered whilst designing artificial self-organising systems of this kind. In particular, we explain the concepts of *complexity*, *emergence* and *openness* that define such systems architectures and influence their behaviour. Followed by this, we discuss the phenomenon of self-organisation that plays a key role in maintaining stable and adaptive behaviour.

2.2 Complex Systems

The main difficulty in building autonomic computational systems is associated with the *complexity* exhibited by such systems. As this property challenges traditional and often centralised methods of mapping functionality to designed components [133, 14, 96], below we provide a more detailed definition of this term.

2.2.1 What is complexity?

According to the Latin root *complexus*, complexity means 'entwined' or 'embraced'. Therefore, in order to have a complex system, there must be [25]:

- two or more distinct parts,
- joined in such a way that it is difficult to separate them.

Here we find the basic duality between parts which are at the same time distinct and connected. The analytical method alone will not allow us to understand the functioning of a complex system, as by separating the components, their connections and the behaviour arising from them will be destroyed. When components are mutually entangled, a change in one component will propagate through a topology of connections to other components which, in turn, will affect even further components, including the one that initially changed. This makes the global behaviour of the system very hard to track in terms of atomic elements [38, 25] even where a complex system's behaviour is familiar and hence to some extent predictable (eg. termites build a mound, birds flock, etc.) it remains difficult to understand (eg., how is the mound achieved?).

Furthermore, the complexity of a system increases with the number of distinct components, the number of connections between them, the complexities of the components, and the complexities of the connections.

2.2.2 Characteristics of Complex Systems

2.2.2.1 Distinction Between Micro and Macro Level in Complex System

A complex system is not characterised solely by the number of components (i.e. being large), but by an architecture of *organised complexity* as a system of systems [74, 33]. It is more than just the items being interconnected: their organisation in a set of interconnected subsystems and the resulting distinct behaviour of the overall system are the defining characteristics of a complex system. Based on this we can observe that there are at least two levels in a complex system [129]:

- micro-level at which the behaviour of atomic system elements is considered (eg., individual agents within a multi-agent system); and
- macro-level that is the level at which global system response is observed (eg., behaviour of the whole multi-agent system).

2.2.2.2 Hierarchical Structuring of Complex Systems

The very first question that needs to be answered is how, despite their architecture, natural complex systems can adapt and effectively manage and organise their functioning. Research in cybernetics focusing on the analysis of natural systems [34] shows that, a multilevel structure of control arises that is composed of a number of the aforementioned micro and macro levels. Such a hierarchical approach allows the decision problem to be factored into different levels, such that decisions at the higher level constrain the decisions at the lower level [34]. The decision at the higher level is easy to make since it

only considers an abstract version of the repertoire of possible actions. The decision at the lower level is easy because only a small part of the problem space remains after the higher level decisions have been made. In this way, the number of choices at each level can be kept acceptably small, while the range of the entire system, which is the product of the choices at the different levels, can be made arbitrarily large.

It is important to note, however, that this recursive definition of hierarchical control states only how a system may be observed or interpreted. The functional analysis shows that such levels of control do not act independently but form heterarchies in which the relations between levels are non-linear and characterised by a high degree of feedback [33]. This is because even within the simplest complex system (comprising only a micro and macro level), both levels do not function independently but influence each other's actions in a non-linear and interactive manner, where the resulting global behaviour emerges from the micro-level and affects the functioning of low-level components.

2.2.2.3 Emergent Behaviour

To understand the functioning of complex systems, it is not possible to approach it through decomposition of structure into its smallest building blocks (such as entities, agents) and further examination of such components independently [96]. The main obstacle to this reductionist approach is caused by *emergent behaviour*, often described by the ancient dictum "the whole is more than the sum of the parts". To explain this phenomenon, assume a multi-agent system to be representative of a complex system with well defined behaviours of the individual agents. In this case, emergent behaviour denotes the appearance of a relatively new global behaviour of the system that is difficult to infer from the most complete knowledge of the behaviours of the individual agents, taken separately or in other partial combinations [33, 129]. Traffic jams are another example of emergent behaviour, taking place within a natural open system. Assuming that a group of cars forms a complex system, we can observe that such a system is capable of exhibiting emergent behaviour in the form of a traffic jam that cannot be inferred from the individual behaviours of cars.

It is important to note that emergent behaviour is a dynamic process whose formation and effects are difficult to predict, since they bring relatively new characteristics to the already functioning system [14, 134]. These novel characteristics can not be understood by analysing behaviour of individual system elements, but are the result of their interactions and arising from it collective dynamics.

2.3 Openness

Because modern IT systems co-exist with each other within a dynamic networked environment, it is important to consider a system as being embedded and influenced by its environment. Such a view has important implications when considering reliable control over system functioning.

Openness in computational systems may be defined in a number of ways, including openness to new information (i.e. learning) and openness to new components or agents [57]. In practice, many consequences (such as dynamics of the environment, complexity of interactions and emergent behaviour) are often neglected while building and modeling modern distributed systems. As a result, this lack of understanding causes most systems to be open to new components and knowledge, but in practice unscalable, uncontrollable and exhibiting unpredictable global behaviour [131, 14]. This is because, so far, software engineering is still in the early stages of defining and envisioning openness and often relies on similarities with closed systems, resulting in a continuation of the *system centric* vision, where the focus is on well-defined systems situated in predictable and controlled environments. Openness here is viewed as an enhancement of closed system capabilities where systems interact in a highly controllable and predictable way with external environmental processes. Progress in building open systems is thus slow and often misguided through the application of control mechanisms that are not scalable and flexible in dynamic and unpredictable environments.

2.3.1 Consequences of Openness

The notion of openness allows us to observe that there are two aspects that define an open system: the system itself and the environment in which it is situated. This can be explained by the illustration in Figure 2.1. Assuming a multi-agent system as an architectural representation of an open system, the figure illustrates the environment populated with agents and services, from which we can distinguish that some of them form an open system, and some are external to it.

2.3.1.1 Openness to Information

According to Hewitt *et al.*'s definition of an open system [32], one can capture the life-cycle of such a system in terms of information that flows to and from the system.

In Figure 2.1 openness to information is represented through the symmetric arrow (IF), indicating the existence of the bidirectional information flow between the environment and the open system. To explain such openness, we can consider system users as con-

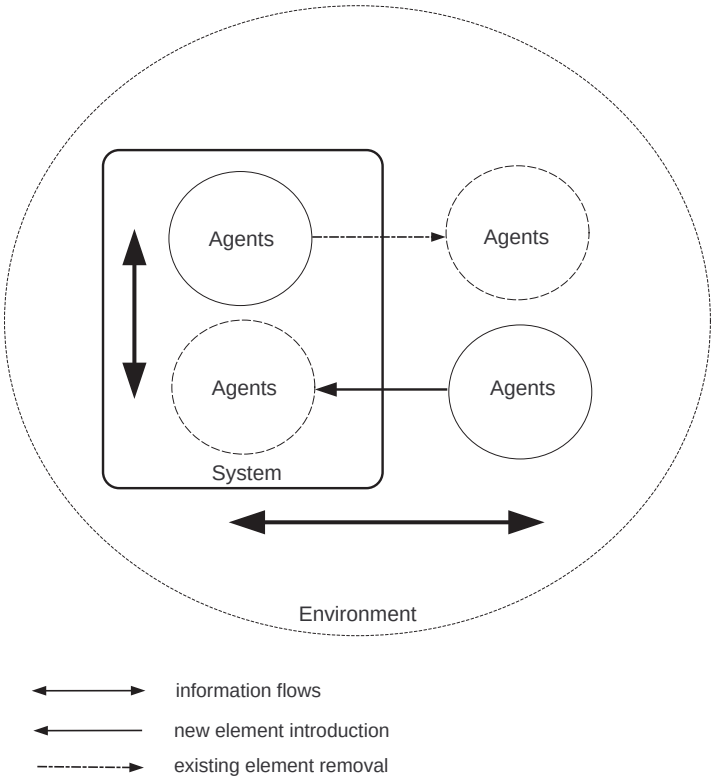

FIGURE 2.1: Representation of an open system

stituting a dynamic environment that exerts pressure on the system to configure appropriately to the demand imposed by users.

2.3.1.2 Openness to Structural Modification

Another important consequence of openness is described in [102] where, apart from openness to information, systems are also open to structural modification. Depending on the characteristics of the system-environment interplay, the system may structure (form) itself using environmental components, causing the boundary between the system and environment in Figure 2.1 to be dynamic and change over time. In the figure, we can observe that the inflow of new information to the system may in turn trigger changes in its composition. For example, increasing demand imposed by system users will require more agents to be created and introduced to the system, whereas reduced demand will lead to a decrease in the number of agents required to consume resources.

2.3.2 Characteristics of Open Systems

2.3.2.1 Embedded Computation

New emerging scenarios in the domain of grid, pervasive and ubiquitous computing show that computation is starting to be injected into the physical world, where components and collectives operate autonomously [101, 20]. Such embodiment also occurs in pure software systems (disconnected with the real world), where open systems may share environments with other systems, which concurrently evolve and adapt to the changing environment. According to [33], such evolution and adaptation is in general parallel and distributed: there is not just one system and its environment, but a multitude of systems evolving simultaneously, partially autonomously, partially in interaction. This *network* structure entails that no absolute distinction can be made between internal and external, i.e., between system and environment, and as a result, what is *system* for one process is *environment* for another.

2.3.2.2 Dynamics

Open systems are in constant flux. The complex interplay between the environment and the system increases its dynamics [86], since it is not clear what will trigger system processes and when. Such dynamic interplay between an open system and its environment is a very complex process, where it has been observed that constant interaction may be propagated and reinforced, causing the system to be never optimally adapted to an environment, since the process of evolution of the system will itself change the environment, so that a new adaptation is needed, and so on [33]. As a result of this continual evolution, a change in the environment may influence the same system to generate a different behaviour (emergent behaviour), without any change in the behavioural characteristics of its constituents [129].

From the system point of view, maintenance of the dynamic state is crucial if the system is to adapt to the changing conditions and avoid critical states [113], where although the interaction with the environment may negatively affect the system functioning, it also sustains its dynamics. As a result, the system is no longer isolated from the underlying environment, but rather its adaptation is driven by interactions with it [33].

2.4 Self-organisation

Analyses of natural open and complex systems suggest that due to their complexity, scale and dependence on other systems situated within an environment, no externally derived mechanism can control their behaviour [105, 17]. Instead, a system itself needs

to impose the force that organises it into a functional structure and resists environmental changes. This is where self-organisation plays a crucial role and can be defined as the spontaneous creation of a globally coherent pattern from local interactions between otherwise disordered collections of interacting parts [33].

Because we are just starting to face the realisation that complex software systems require more flexible and autonomic [58] approaches to the control of their global behaviour [131, 69], self-organisation is an attractive but fairly new and relatively unexplored concept in this domain. This section briefly presents key concepts related to self-organisation and explains the requirements for its occurrence.

2.4.1 Characteristics of Self-organising Systems

2.4.1.1 Global Order From Local Interactions

The emergence of the global organisation is provided by local interactions of the elements that belong to a self-organising system. This suggests that self-organising systems are capable to maintain their organisation relying on decentralised coordination mechanisms arising from the actions of individual entities.

2.4.1.2 Non-linearity and Feedback Loops

The dynamics of a natural self-organising system is typically non-linear because of circular or feedback relations between the components. This results in a less straightforward relationship between cause and effect, where small causes can have large effects, and large causes can have small effects. Such non-linearity can be understood from the relation of feedback that holds between the system's components, where each component affects other components, but these components in turn affect the first component. Thus the cause-and-effect relation is circular and any change in the first component is fed back via its effects on the other components to the first component itself. Feedback can have two basic values: *positive* (that reinforces or amplifies the initial changes and makes deviations grow in an explosive manner) or *negative* (that suppresses change and stabilises the system, bringing deviations back to their original state). In complex self-organising systems, there are several interlocking positive and negative feedback loops, so that changes in some directions are amplified while changes in other directions are suppressed. This can lead to very complicated and difficult to predict emergent behaviour that effectively maintains organisation within the system.

2.4.1.3 Distributed Control, Robustness and Resilience

In self-organising systems, control of the organisation is typically distributed over the whole of the system, where all parts contribute evenly to the resulting arrangement. Robustness or resilience means that self-organising systems are relatively insensitive to perturbations or errors, and have a strong capacity to restore themselves. There are three reasons for such behaviour:

- self-organising systems rely on redundant, distributed organisation allowing undamaged regions to make up for damaged ones;
- self-organisation thrives on randomness, fluctuations or noise, which generate a large enough variety of actions to compensate for unpredictable situations; and
- the overall organisation is stabilised by positive and negative feedback loops.

Neural networks are good examples of computer simulations of self-organising systems. Such networks, trained to achieve a certain task, will in general still be able to perform that task when damaged, for example by the random removal of nodes and links. Increasing damage will decrease performance, but the degradation will be *graceful*; that is, the quality of the output will diminish gradually without a sudden loss of function. A traditional computer program or mechanical system, on the other hand, will stop functioning if random components are removed.

2.4.2 Self-organisation Process Overview

2.4.2.1 Self-organisation Processes

Organisation can be described as the characteristic of being ordered or structured so as to fulfill a particular function [25]. By being structured we mean that the components of a system are arranged in a particular order, represented by both connections, which integrate the parts into a whole, and separations that differentiate them, so as to avoid interference. Function, in turn, means that this structure fulfills a purpose.

Research investigating the process of self-organisation in natural open systems [33], suggests that the organisation within these systems arises through the self-organising process of blind *variation* (the generation of a large number of possible states or actions) and natural *selection* of the most preferred ones.

Variation explores different regions in the system's state space until it enters a state space in a range of an attractor, which precludes further variation outside the attractor, and thus restricts the freedom of the system's components to behave independently. Because not all attractors within the system result in an organised structure, the selection process

is responsible for influencing the variation process to reach the attractor desired for the organisation purposes. Variation can be motivated and achieved either by change a of internal system configuration (internal variation, for example interactions between components), or through change in the relationship between system and environment (external variation), for example interaction with external components or systems.

Selection, on the other hand, acts as an inward (produced by the system itself) pressure that acts on individual system elements by constraining their degrees of freedom. The resulting from this loss of autonomy of individual elements enables them to establish stable patterns of interactions that are coherent with global system organisation.

2.4.2.2 Requirements for Self-organisation

Mechanical control systems, such as a thermostat or an automatic pilot, have both *variety* and *selectivity* built in by the system designer, but dynamic systems need to autonomously evolve these capabilities. For this to happen, there are certain requirements with respect to system architecture and functioning conditions that have been observed by Nobel prize winner Ilya Prigogine [56] in the area of *dynamical far-from-equilibrium systems*.

Firstly, Prigogine showed that variety within such systems can be fostered by keeping the system in dynamic state so that it has plenty of stationary states to choose from. In physical systems the increase in system dynamics is mostly achieved through input of energy flow that 'agitates' individual system elements due to a surplus heat conducted through system elements through their interactions. This leads us to the second self-organisation requirement that is, the system has to be open. Finally, as Prigogine observed in his studies, the system has to belong to class of *dissipative systems*, in which a certain quantity, usually its energy (often in a degraded form), is dissipated from the system, since only in such systems are *attractors* or points to which the system converges present.

2.4.3 Thermodynamic Account of Self-organisation

Observations of our real world supply us with enough evidence that surrounding us natural systems such as ecologies [56], social systems [28, 7] or even markets [111] exhibit a strong tendency to organise and to continue producing even more complicated and sophisticated patterns of order. This is somewhat surprising considering the difficulty in arriving at that ordered state based only on the variation and selection processes explained in this chapter.

Even more puzzling becomes the observation [116] that such order producing processes are not limited only to living systems, but arise in non-living physical or chemical com-

pounds [117]. Whereas in living systems the self-organisation could be explained as a mechanism that emerged through evolution and was employed as an internal regulatory mechanism aimed to preserve individual's survival, the analogous explanation of this phenomenon within the physical domain becomes invalid as these systems do not exploit any evolutionary based mechanisms, yet still produce ordered structures.

This already implies that self-organisation cannot be attributed only to intelligent planning or local mechanisms that were adjusted through long and complex evolutionary process but arises even from the simplest interactions between non-complicated and non-living elements. Despite this, we can observe that there exists an end-directed *aim* of both physical and biological systems to choose organised states over the ones that are not. Considering the fact that achieving and maintaining organisation is less probable than relaxing into disordered state, there is a growing understanding that self-organisation is a consequence of certain laws of physics [54] studied in the area of thermodynamics. Based on these studies, in the next chapter we will address two of the most important questions: 1) why and under what conditions do systems choose ordered states over the ones that exhibit less organisation; and 2) how can we interpret order production process accompanied during self-organisation in a manner that would allow us to apply it for artificial systems engineering.

2.5 Thermodynamics of Self-organsation

One powerful strength of a thermodynamic account of self-organisation is its potential to apply across physical, chemical, biological, social, and socio-technological domains. However, it is most clearly and straightforwardly articulated in the absence of the beliefs, desires, and functions that are proper parts of the 'higher' systems. Here we first present the thermodynamic framework in the context of physical and then biological systems before demonstrating its application in the context of the particular class of socio-technological system explored in the remainder of this thesis.

2.5.1 Thermodynamics of Self-organisation

Studies investigating the thermodynamics of self-organisation in far-from-equilibrium systems can be found in [80, 116, 55]. Irrespective of whether the investigated system is described in terms of 'dissipative structures' [80], autonomous agents [55] or an autocatakinetic system [116], self-organisation is interpreted as a process of organised energy flow from which work can be extracted and employed by the system for its structure maintenance [56, 128, 117]. Central to understanding this process are the following concepts derived from thermodynamics: displacement from equilibrium, energy transfer, gradient dissipation, constraint formation and work.

2.5.2 Displacement from Equilibrium

According to classical thermodynamics, the behaviour of physical systems can be explained as transformations of energy between the system and its surroundings. Hence, when both are allowed to interact, what is exchanged between them is energy [56]. Energy, here, has a general meaning, defining the capacity of the system to perform work, and may be added to the system by increasing its temperature, pressure or a chemical potential.

Considering the energy of the system and its environment, we can measure the relative difference between both, often defined as a potential or gradient. If the gradient is equal to zero, meaning that both the system and its environment have the same energy (e.g., temperature, or pressure) we consider them to be at equilibrium. In this state, the system is indistinguishable from its environment and has no capacity to perform work. Any deviation from equilibrium implies that free energy is stored, and that there may be the potential to release this energy through useful work. The extent to which a system is displaced from equilibrium is reflected in the gradient (difference) between the state variables defining its energy state (e.g., temperature) and that of its environment.

2.5.3 Energy Transfer

To displace a system from equilibrium requires that it be supplied with energy (be it thermal, mechanical or chemical), distinguishing it from its surroundings. According to the first law of thermodynamics, energy transfer can proceed in two different ways: through heat (Q) and work (W). This is captured in the formula summarising the first law:

$$dU = dQ + dW,$$

where dU is an infinitesimal change in the internal energy of the system, dQ is the infinitesimal amount of heat added to the system and dW is the infinitesimal amount of work done on the system. Although heating up a system and performing work on it will each increase its energy, each differs in the manner in which energy is being distributed in the system and thus whether the system moves away from equilibrium.

This difference is reflected through entropy (S) which can be interpreted as a measure of the uncertainty about how energy is distributed in the system [50, 51]. Adding heat (Q) to the system increases our overall uncertainty about the energy content of the system and causes proportional increase in entropy. This is manifested through the following relation:

$$dS = dQ/T,$$

where S is the entropy, dQ is the infinitesimal amount of heat added to the system and T is the absolute temperature of the system. For this reason, it represents the amount of

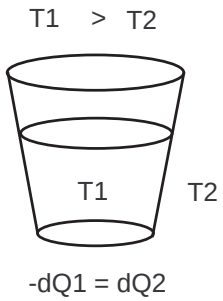

FIGURE 2.2: A glass of liquid at temperature $T1$ is placed in a room at temperature $T2$, where $T1 > T2$. The disequilibrium produces a field potential that spontaneously drives a flow of energy in the form of heat, $-dQ1$, from the glass to the room so as to drain the potential until it is minimized (the entropy is maximized). At this point thermodynamic equilibrium is reached and all flows stop. The expression $-dQ1 = dQ2$ refers to conservation of energy in that the flow of heat from the glass equals the flow of heat into the room.

energy that we lose information about when it is transferred and that we are thus unable to extract. When, on the other hand, work is done on the system (W) our knowledge about the energy content of the system increases, thus we are better able to distinguish between the system and its environment. In this case, work done on the system does not affect internal system entropy and thus represents the only way to move a system further from equilibrium [56].

2.5.4 Gradient Dissipation

The second law of thermodynamics states that if two systems are allowed to interact and exchange energy, that is if the constraints imposed between them are removed, then the systems will evolve to equilibrium, a new state in which we cannot differentiate between the systems. A statistical consequence of this physical law is that entropy will increase.

The active nature of the second law is intuitively easy to grasp and empirically easy to demonstrate. Figure 2.2 shows a glass of hot liquid placed in a room at a cooler temperature. The difference in temperatures in the glass-room system constitutes a potential and induces a flow of energy in the form of heat. This 'drain' on the potential flows from the glass (source) to the room (sink) until the potential is minimized (the entropy is maximized) and the liquid and the room are at the same temperature. At this point, all flows and thus all entropy production stops and the system is at thermodynamic equilibrium. The same principle applies to any system where any form of energy is out of equilibrium with its surroundings (e.g., whether mechanical, chemical, electrical or energy in the form of heat).

The second law alone does not tell us which of the available energy transfer paths a system will select in order to move back to equilibrium. The idea can be demonstrated in a classic experiment on self-organisation first devised by Henri Bénard in 1900 [117].

A viscous fluid is held between a uniform heat source below and the cooler temperature of the air above. That is, there is a potential difference between fluid and air with a field force of a magnitude, F, determined by the difference between the two temperatures. When F is below a critical threshold heat flows from the source (fluid) to the sink (air) in the form of disordered collisions between the constituent molecules, and entropy is produced. If F exceeds the critical threshold Bénard 'cells' emerge spontaneously, each cell consisting of hundreds of millions of molecules moving collectively together in the form of rotating vertical convection columns. In this organised mode, the transfer of energy through the system and its dissipation to its surroundings is much more efficient than through unorganised collisions [103]. Such behaviour does not violate the second law. As long as a self-organising system produces entropy (minimises potentials) at a rate that is sufficient to compensate for its own ordering (persistence away from equilibrium) then the balance demanded by the equation of the second law is not violated [56, 117].

2.5.5 Work

So far we have discussed displacement from equilibrium, constraint on energy transfer and gradient dissipation as distinct concepts describing the active nature of physical laws. But how can they be employed to control energy movement within systems, such that useful work could be extracted from their functioning [52]? Consider a system consisting of two connected tanks of equal volume but with different numbers of gas molecules. This difference defines a gradient between both tanks. As soon as a conduit between them is opened, gas whooshes through it, equalising the number of molecules in the tanks and erasing the gradient between them. Gas can rush through even if it has to turn a turbine along the way, thereby doing mechanical work. The energy to do that work came from the thermal energy of the environment, but the conversion from thermal to mechanical energy was paid for by the increase of disorder as the system equilibrated. Now, if we repeat the first process again by first closing the conduit and transferring energy from one tank to the other, we can repeat the same process of work extraction and gradient dissipation. Although simplified, this principle of work extraction constitutes a thermodynamic work cycle, which underpins the supply of most of the world's electric power and almost all motor vehicles.

2.5.6 Information

Within statistical mechanics, the entropy of a system at equilibrium can be recast in terms of the variety of microscopic states available to the system:

$$S \equiv k \ln \Omega,$$

where Ω is the number of states in which the system can be found when at equilibrium, and k is the Boltzmann constant, $1.38x10^{-16}J/K$. Consequently, entropy has been interpreted as a measure of macro-level disorder, formalised as Shannon entropy [108] defined as:

$$S = -\sum p_i \log p_i,$$

where i ranges over the possible states of the system and p_i is the probability of finding the system in state i.

As such, it is possible to reinterpret the thermodynamic work cycle in information theoretic terms [52, 78]. We have seen that the difference between doing work on a system and merely heating it up is the difference between how informed we are about the organisation of the system's energy. The potential gradient that must be established within a system before useful work can be extracted from it is thus also an informational property. Given that we are interested in computational systems that consume electricity and also process information, there is scope for the equivalences between information, energy and entropy to be useful, but also confusing.

2.5.7 Thermodynamics Beyond Physics

The application of thermodynamics is not limited only to physical systems [52]. Ever since Alfred Lotka (1922) began writing about energy flows as the basis for natural selection, there has been a thermodynamic paradigm in evolutionary theory. Lotka observed that selection will favour those organisms that, in pulling resources into their own service, also increase the energy throughputs of their ecosystems [128]. What all organisms have in common is that they operate and evolve at some distance from thermodynamic equilibrium. By doing so they maintain the integrity of their organisational structures by irreversibly degrading free energy through informed kinetic pathways acquired through evolution. From this perspective, succession can be considered as the process by which an ecosystem moves away from thermodynamic equilibrium with its environment [56]. By developing this account, the principles of variation and natural selection can be given a sound thermodynamic basis. The principle of variation derives from two sources: the entropic drive to generate configurational randomness and the quantum indeterminacy about where that randomness will occur. Natural selection follows from competition among alternative patterns of energy utilisation [127].

One consequence of this perspective is an increasing appreciation that organisms can be viewed as more sophisticated 'engines' than the physical systems described so far [116]. According to [55], for instance, life or its physical manifestation can be described in terms of an autonomous agent. This agent is a collectively autocatalytic system performing one or more thermodynamic work cycles that: (1) measures useful displacements from

equilibrium from which work can be extracted; (2) discovers devices that couple to those energy sources such that work can be extracted; and (3) applies work to develop and maintain the constraints that enable the further extraction of work.

2.5.8 Thermodynamic Account of Self-organisation in Computational Systems

Whilst much is still to be understood in relation to the role the thermodynamics plays at producing order and life in particular [54], initial models how it may be applied to study and understand the self-organising properties of decentralised and information-driven systems have already been provided [24, 29].

These models suggest that the concepts such as equilibrium, constraint and work, which were initially derived from the study of thermodynamics can be generalised and applied to computational, multi-component infrastructures represented as multi-agent systems.

More specifically, as suggested by Gambhir and Guerin in [24, 29] the equilibrium within an information-driven system can be associated with the behavioural degree of feedom of a software agent. Here, each agent may be characterised by its behavioural *repertoire*, the set of actions that are currently available to it. During each decision-cycle, an agent is required to select one action from the set of available ones and, by executing it, act upon its environment. The behaviour of the agent will exhibit the highest degree of uniformity when selection of any action is equally probable during each decision cycle, and the agent behaves randomly. Since the degree of uniformity of the whole population can be measured as the average over individual agent states, a multi-agent system can be said to be at equilibrium when all agent decisions are made at random.

Given this, the emergence of constraint that would influence agent to favor only certain actions from the whole *repertoire* would indicate agent's (or system's) displacement from equilibrium and thus more organised state from which useful work can be extracted.

Relying on this intuitive interpretation of equilibrium, in what follows we provide two examples of such software system architectures [90, 24], the functioning of which is interpreted from the thermodynamical viewpoint outlined above. In each case, the local decision-making of individual system elements is achieved through the creation and destruction of gradients achieved through organised flow of information.

2.5.8.1 Entropy in a Two-agent System

A thermodynamic account of self-organisation within a multi-agent system is presented by Parunak and Brueckner in [90]. The authors consider a simple coordination problem between two agents who desire to be together, one a mobile walker, the other in a fixed

location. Both agents are embedded within a spatial environment with neither knowing the location of the other. The coordination problem for the walker is to locate the other agent and move towards it. An intelligent observer capable to seeing the state of both agents could send instructions to direct the movement of the walker. However, in this model Parunak and Brueckner investigate stigmergic coordination inspired by organisation in insect colonies. For this purpose, the stationary agent deposits pheromone molecules at its location. Initially, the walker is unable to sense any molecules and performs unguided movements. However, once pheromone molecules diffuse through the environment and are detected by the walker, it follows the gradient formed by them, thus reaching the target. We can understand how self-organised system behaviour emerges from the random processes of pheromone molecule diffusion on two levels: a macro-level at which coordinated behaviour of the walker agent arises; and a micro-level represented by a random motion of pheromone molecules that diffuse through the environment. An analysis of system organisation at both levels based on Shannon's entropy reveals that an increase in the micro-level entropy (as pheromone molecules diffuse to occupy an increasing number of locations) is accompanied with a decrease in entropy at the macro-level (as the movement of the walker is increasingly informed by the pheromone gradient).

This simple example illustrates not only how 'intelligent' behaviour emerges from a simple, entropy increasing processes, but also that the resulting self-organisation does not defy the second law of thermodynamics since the price paid for the entropy reduction at the macro system level is the increase in entropy generated by the random process that produces and maintains the gradient.

2.5.8.2 A Full Population Model

A continuation of Parunak and Brueckner's work is presented by Gambhir *et al.* in [24]. Here, the authors apply a computational model of an ant foraging system to demonstrate how complex organisation of interacting agents can be explained in terms of ideas from far-from-equilibrium thermodynamics. Their analysis of this classic example of self-organisation distinguishes three distinct modes of system behaviour: structure formation, structure maintenance and structure decay. During structure formation, some members of a population of agents diffusing over the environment discover a food source and establish a pheromone distribution instructing other agents to organise their activities into a foraging trail. By maintaining this structure, the population achieves reliable transport of food to the nest. Once the food source becomes depleted, the structure begins to decay and the agents return to their initial disorganised state.

To interpret how the system is displaced from equilibrium and how work is extracted from these conditions, the authors evoke ideas of unconstrained and constrained transfers of energy that are responsible for thermodynamical organisation and work extraction.

Within a computational system, unconstrained flow of heat is considered as a diffusive, entropy producing process of agents performing random walks. By contrast, constrained transfer of energy, in the form of interactions with an organised pheromone distribution, is interpreted as work done on agents, constraining their behavioural degrees of freedom (i.e., agent movements are directed to climb the pheromone intensity gradient, as in the case of the walker agent discussed above). The insights drawn from this model are similar to those arrived at by Parunak and Brueckner [90]. An initial increase in entropy, during which agents explore the state space, enables the formation of organisation, imposing constraints on agent behaviour through interaction with the pheromone field. To measure construction and destruction of constraints in this self-organising system, Shannon's entropy is applied. The measure of *useful work* done by the system is represented by the number of pieces of food taken from the food-source to nest over a run.

2.6 Discussion

At the end of the previous chapter we were interested in understanding why do natural systems tend to prefer organised states over the ones that exhibited disorder. To answer this question we hinted that self-organisation is a process that arises as a consequence of certain physical laws. These laws as well as their account in achieving self-organisation were presented in this chapter.

Surprisingly, and against our initial expectations, the role of these laws (the second law of thermodynamics in particular) in achieving self-organisation is, at the first glance, dubious as they describe the end-directed progression of the universe towards more disordered state [56].

However, as these studies also show, it is this very *destructive* and ubiquitous nature that allows spontaneous ordering to take place. In here, self-organisation is not considered as an end-directed goal that aims to produce as much order within the universe as it is possible, but is merely a side effect of the opposite processes that tend to produce disorder. How can this puzzle be understood?

Even the most classical self-organisation experiment (Benard cells) that we have described in Section 2.5.4 shows that, given the natural tendency of the system to move into more disorganised and low energy state, it will choose the internal configuration that allows it to degrade the supplied into it energy at the fastest rate. This can be achieved once the system configures itself to fulfill that goal and this is when the self-organisation takes place. In this case, the self-organised state would be sustained as long as the system is fed with energy that displaces it further from equilibrium and thus increases the tendency of the system to move back into the low energy state.

If we now consider living organisms, we can interpret their functioning under the same lines. The difference would be that now we are dealing with organisms that exhibit intentional dynamics [117, 116] which allows them to identify and couple to the sources of energy located within their environment, such that their organisation could be sustained and not employed only for energy degradation but also for work extraction that preserves the survival of the organism [55].

What are the implications of the above provided thermodynamic account of self-organisation within the context of computational system engineering? In the reviewed in this chapter self-organising computational models, we have described how information disparity drives self-organisation in a population of software agents and that random behaviour is an integral part of the maintenance of information flows that allow such a population to organise effectively. This contrasts starkly with the (sometimes implicit) assumption present in the multi-agent system community that software agents share complete knowledge of the system, and make decisions as a result of joint deliberation, or at the behest of a central executive charged with deducing optimal behaviour. This approach is analogous to relying on a kind of maxwell demon to control a computational ecosystem. The demon knows the position and state of every element in the system and is able to impose/remove constraints that allow the system to do useful work. However, thermodynamic considerations imply that, even if such a demon could be implemented, it would be extremely costly.

The interpretation provided here should not be considered exclusive. While thermodynamics and self-organisation have been the object of extensive research, there are still open questions with respect to the application of these ideas to systems that are far from equilibrium but capable of maintaining steady state [56]. In such cases, considerations of thermodynamical systems at, close to, or moving towards their equilibrium state are insufficient, making far-from-equilibrium thermodynamics an open and active area of study with direct implications for engineering open computational ecosystems.

More importantly, if we aim to engineer self-organising IT systems, we must understand the underlying thermodynamic principles of natural self-organisation, and, in particular, how to apply these principles in the context of open IT systems. For this purpose, in the next chapter we outline the existing work that has been done in the area of decentralised system engineering with explicit focus on means through which self-organising system response was obtained. Based on this, we contrast the presented approaches with discussed in this chapter thermodynamic principles of self-organisation.

Chapter 3

Computational Complex Systems

In the previous chapter we introduced some general properties of complex systems together with the means they employ for sustaining stable behaviour and organisation. In this chapter we will focus on the application of such systems to the control problems relevant to IT systems. In particular, we will introduce in detail the aims of Autonomic Computing, outline multi-agent system models suitable for realising this computation and, finally, review the existing work aiming to apply natural self-organisation and complexity studies to engineering artificial systems.

To this end, the chapter is organised in the following manner. In Section 3.1 we outline autonomic computing challenges. Section 3.2 introduces agent-based computing and multi-agent system models. Following this, Section 3.3 reviews three different artificial complex system models within which self-organisation was studied and applied for achieving useful function from the system. We conclude with discussion in Section 3.4.

3.1 Autonomic Computing

The success of Autonomic Computing, according to IBM [58], will deliver control techniques that present remarkable ability in:

- **homeostasis**: carrying out self-regulatory functions across a wide range of external conditions, always maintaining a steady internal system state called *homeostasis*;
- **automaticity**: doing all this without any conscious recognition or effort from the system; and
- **holism**: self-governing the whole system, not just parts.

Each of these properties points to significant characteristics of self-regulatory mechanisms in natural systems that so far have been oversimplified, discarded or not sufficiently well understood, whilst considering control mechanisms for software systems. In particular, self-regulation is viewed here as a response of a system to a number of wide external conditions, directed towards maintenance of a steady internal state. This implies open and dynamic systems tasked to self-manage their activity in response to some external pressures.

However, self-regulation is not a *conscious* effect of some centralised authority continuously inspecting the state of every element and the existence of, for example, component failure applying recovery plan. Rather, global self-management arises as a product of the interactions of individual elements and their local responses to sensed changes in the same principled way as the social intelligence of an ant colony arises largely from the interactions among individual ants [121]. This suggests that self-regulation is an effect of some self-organising processes the system is relying on and willingly exploiting for its own benefit.

Finally, just as an increase in heart rate without a corresponding adjustment to breathing and blood pressure would be disastrous, bringing autonomic capabilities to storage systems would certainly be an improvement, but if the computing systems that mine the data in those storage repositories become next to impossible to manage, that partial automation will not yield much overall benefit [42]. Autonomic computing is thus a *holistic* vision that will enable the whole of computing to deliver much more automation than the sum of its individually self-managed parts. This leads to an important consequence — it does not suffice to produce autonomic elements such as particular infrastructure components in order to engineer self-regulating systems. System-level engineering is required.

3.1.1 Aims of Autonomic Computing

Although the realisation of fully autonomic computing will take many years and will require substantial understanding of natural system functioning, the properties of modern IT systems allow the identification of four key domains within which self-management will be required [58]:

- self-configuration — configuration of components and systems follows high-level policies, and the rest of the system adjusts automatically and seamlessly;
- self-optimisation — components and systems continually seek and achieve nonlinear tuning parameters and opportunities to improve their own performance and efficiency;

- self-healing — the system automatically detects, diagnoses and repairs localised software and hardware problems; and
- self-protection — the system automatically defends against malicious and cascading failures. It uses early warnings to anticipate and prevent system-wide failures.

3.1.2 Autonomic System Architecture

Autonomic computing envisions self-managing software infrastructures as computational ecologies [26] comprising myriads of autonomic elements, each adjusting their functioning according to the perceived state of the environment and the response of other autonomic elements [42, 9].

Whilst autonomic systems will utilise distributed computing and service oriented architectures [67], system components, including services, will be represented as autonomous *loci* of control. Consequently, the systems self-management will be a product of the interactions between and decisions of individual elements, with the objective of monitoring managed processes (eg., services) and their external environments, and executing behaviour capable of preserving expected behaviour.

The behaviour of each autonomic element will be driven by goals that its designer has embedded in it, by other elements that have authority over it, or by subcontracts to peer elements with its tacit or explicit consent. However, to achieve flexibility and dynamism, it is assumed that more sophisticated techniques will be utilised to express the goals and objectives of autonomic elements [58, 71]. Thus, hard-coded behaviours will give way to high-level objectives such as 'maximise this utility function' or 'find reliable provider'. Similarly, hard-wired connections between system components will be replaced with dynamic and flexible interaction patterns involving less direct specifications of an element's partners — from specification by physical address to specification by name and finally to specification by function, with the partners identity being resolved only when it is needed.

Whilst all this will be provided on top of pre-existing service oriented architectures (SOA), autonomic features will augment these models with features unavailable previously. First, as service providers, autonomic elements will not offer services to every requesting consumer, as would typical web services or objects in an object-oriented environment. They will provide a service only if providing it is consistent with their goals. Second, as consumers, autonomic elements will autonomously and proactively issue requests to other elements to carry out their objectives.

Because the complexity of system management will be devolved into autonomic elements, they will involve more complexity than simple objects. Thus, autonomic elements will have complex life cycles, continually carrying on multiple threads of activity, and continually sensing and responding to the environment in which they are situated.

Autonomy, proactivity, and goal-directed interactivity with their environment are distinguishing characteristics of software agents [53, 88]. Viewing autonomic elements as agents and autonomic systems as multi-agent systems makes it clear that agent-oriented architectural concepts will be critically important whilst engineering self-managing systems.

3.1.3 Autonomic Computing Challenges

With autonomic computing being at the forefront of a paradigm shift of modern computing [134], the difficulties involved in its achievement have already been identified as nontrivial and interdisciplinary. In particular, understanding of following properties is considered crucial for progress in engineering self-managing software systems [58]:

- *Emergent behaviour.* Interactions between independent and autonomic computing elemenets will produce qualitatively new behaviour on the global system level [8, 6]. Controlling, and designing this emergent behavior in autonomic systems is a challenge at the heart of autonomic computing [58, 93].
- *Robustness.* Autonomic systems cannot guarantee to operate without system element malfunctions. Consequently, a theory of robustness for autonomic systems, including definitions and analyses of robustness, diversity, redundancy, and optimality, and their relationship to one another is required [22, 115].
- *Adaptation.* Actions performed in parallel by myriads of autonomic elements stress the effects of learning conducted individually by each component to preserve its local goals. Furthermore, the learning and decision-making of each element will affect the functioning of others with no guarantee that the system will converge to the optimum [2]. Thus what is needed are design principles and learning mechanisms that cope well in described environments [94, 133]. With respect to this, machine learning by a single agent in static environments is well studied, but learning in multi-agent systems is a challenging but relatively unexplored problem, with virtually no major principles and only a handful of empirical results.

3.2 Multi-agent Systems

When addressing complex computational problems, multi-agent systems offer more convenient alternatives to centralised systems that often struggle with a lack of scalability and flexibility in response to unexpected conditions. From this perspective, the traditional way of conceiving software, in terms of functional entities interacting with each other in a client-server fashion, is substituted by a perspective in which software is modeled and designed in terms of autonomous software entities (agents), situated in an

environment and capable of proactive actions toward their individual goal achievement [86, 84].

3.2.1 General architecture of multi-agent systems

Multi-agent systems can be described using two different levels of granularity (that also relate to different focuses of research within the multi-agent system community), as follows [134]:

- micro-level (focuses on individual agents, their architectures and corresponding decision-making mechanisms); and
- macro-level (agent societies, organisations and teams) at which the multi-agent system displays global behaviour, whose conformance to the application requirements, for instance, determines its success as a whole.

Another important aspect of multi-agent systems is represented by the environment, considered to be an integral part of such systems [57, 81, 89] that both influences and is influenced by the actions of agents.

3.2.2 Decentralised Control and Autonomy

The ability of agents to make autonomous decisions enables them to control their own actions to a higher degree and to exhibit much more autonomy than the components of traditional distributed systems [86]. As a result, the multi-agent system approach offers the capability to represent the system-level objectives (global goals) in a distributed manner, where individual agents are responsible for the achievement of sub-goals and therefore the overall system objective results from the actions of all agents.

The flexibility of the system is therefore achieved through decentralised system control and autonomy at a lower level, represented by the individual agents. Such architectures offer much greater ability to quickly counteract unexpected situations by enabling agents to autonomously respond to detected problems (i.e. failures), whereas decentralised global control minimises the chance of a serious system failure in case of partial system damage [91, 88, 134].

Although this approach offers valuable characteristics for building large scale and complex distributed systems, the reliable and predictable functioning of multi-agent systems depends strongly on the ability to organise and coordinate the actions of individual agents.

3.2.3 Coordination Mechanisms

The introduction of multiple *loci* of control, resulting in the overall decentralisation of system control, if not properly influenced, may lead to a decreased performance of a system or its unpredictable functioning. Such undesired behaviour, in general, arises because agents are concerned only with the achievement of their individual goals, yet the system objective (defined at the global level) results from the actions of all agents. As a result, multi-agent systems need to be endowed with coordination mechanisms ensuring that agents interact in a manner that permits their activities to be developed and integrated into the overall solution. In turn, this can be represented as a process of efficient management of the dependencies (conflicts) arising between agent activities [68], leading to organised behaviour of the system. To enable such coherent behaviour we distinguish four general approaches, as follows.

3.2.3.1 Controlling the Degree of Interaction Between Agents

The most effective and restrictive way to coordinate a group of agents is to control the degree of interaction between them by exerting strict control over each agent and each interaction [132, 133], so that agents can interact in predictable and required ways at each level. Such techniques for controlling the degree of interaction impose a relatively static organisational structure on agents [19], where their autonomy and flexibility are reduced to provide controlled interactions. This approach works well for very simple and deterministic environments, where the size of the multi-agent system is very small and the environment is static. In such scenarios agents are able to coordinate efficiently and cooperate through contracts, rational planning and collective decisions in order to achieve the requested goal [86, 15].

3.2.3.2 Computational Organisations

Following and extending the previous strategy, many approaches model multi-agent systems as organisations of interacting agents [86, 130], where agents are endowed with cooperation and negotiation capabilities that imitate human interactions, and are able to form computational organisations [123]. The shift toward computational organisations shows the tendency to increase the autonomy of individual agents and to control their actions through organisational roles and prohibitions, making the coordination problem distributed among a number of agents.

3.2.3.3 Coordination Through Environment

Coordination techniques relying on the organisational metaphor or on control over agent interactions are mostly achieved through mentalistic attitudes of agents and explicit exchange of messages (seen as communication acts). This often results in a small number of complex software agents, gifted with reasoning and planning capabilities.

Another important approach to the coordination problem tends to exploit interactions between agents and the explicitly modeled environment [83]. This is either achieved through *coordination artifacts* [82], representing shared information repositories (eg. shared blackboards), situated within an environment, and further shared and exploited by a collective of agents for achieving a social objective or by relying on *stigmergy*. Inspired by natural biological systems [27, 91] stigmergy that is indirect interaction through active environments in which agents deposit information, thus influencing and directing the behaviour of other agents. In this case, the role of the environment is to limit the information dissemination to relevant agents (those local to the origin of the information) and to allowing the state or irrelevant information to leave the system. This group of coordination mechanisms is the most promising for large scale systems since it acknowledges the existence of an uncertain environment and exploits the full potential of interacting and situated agents both in offensive and defensive ways, by enabling autonomous entities to stimulate the system dynamics through interaction with the environment, thus making it more adaptive and responsive to unpredictable conditions.

3.2.4 Multi-agent Systems in Practice

Despite the architectural advantages of multi-agent systems over centralised systems, their potential and practical application has been, so far, limited to small scale and fairly static environments. This is due to the extensive focus on scenarios relying on the deterministic conditions of multi-agent system functioning, which limits the application of existing coordination mechanisms in real-world dynamic and unpredictable conditions. As a result, in practice most research within the multi-agent systems community is influenced by traditional engineering, making multi-agent systems evolutionary rather than revolutionary in terms of the models and approaches used during their design. This can be represented by the following issues that need to be considered while building effective multi-agent systems.

3.2.4.1 Scalability

The very first problem is scalability. Increasing the number of agents affects the dynamics of the system and results in a decreased ability of individual agents to obtain global and up to date information about the system, thus localising their perception.

This removes the possibility of efficient centralised control and requires the application of decentralised coordination mechanisms [10, 123]. As a result, it has been noted that increasing the number of interacting agents within multi-agent systems has a large impact on the effectiveness of coordination techniques, making most of them inefficient and unscalable [19].

3.2.4.2 Dynamism

Multi-agent systems approaches often neglect the existence of an environment, or wrap it as a system component, considering it as a deterministic and fully controllable *inner* part of the system [89]. An observed shift toward more uncertain environments [134], where agents are required to operate in dynamic conditions (for example sensor networks or grid environments), shows that environments play an important role in the overall system functioning and can no longer be neglected or simplified. As a result, approaches that try to wrap the environment into a system component may become limited, due to the unpredictability and instability they bring, which can distort the system [89].

3.2.4.3 Top-down Control and Autonomy

The increasing requirement to deploy agents within uncertain open environments, and the fact that the number of interacting components (agents and services) may vary and be impossible to control, requires a focus on dynamic aspects of the system [113]. Unfortunately, many classical approaches emphasise static models of the system, focusing on entities, states and events, where to make agents act predictably, their autonomy is limited and often traded off against control of the whole system using a top-down approach [91], requiring any desired system-level behaviour to be explicitly represented in lower-level components [92]. This can significantly decrease the flexibility of multi-agent systems that require higher autonomy in order to adapt to unexpected changes [33].

3.3 Computational Complex Systems Review

Provided the architecture of multi-agent systems and its decentralised nature resembling many natural systems, let us now focus on examples of computational systems the operation of which was inspired by the behaviour of natural self-organising systems.

3.3.1 Cellular Automata

3.3.1.1 Architecture

A Cellular Automata (CA) is an architecture consisting of a number of components (cells) situated on a lattice on which the cells form a row of neighbouring elements. Here, we focus on the most commonly used CA architecture, where a row has only one dimension (the system components form a one-dimensional array) and the cells can only be in one of two states (0 or 1). It is also assumed that the rightmost cell of the lattice has the leftmost cell as a neighbour, thus making the whole row have the form of a torus.

The functioning of the system is performed through interactions between cells, where it is assumed that only the neighouring cells can interact. The locality of interactions is defined through the *radius* (r), which identifies the neighbourhood size of each cell. For example, radius equal to 3 defines the number of neighbouring cells on each side of the cell.

3.3.1.2 Computation in CA

The computation in CA is achieved through a cell's state transitions from one state to another, briefly described below. The row of cells starts out with some initial configuration of local states and, at each time step, the states of all cells in the row are synchronously updated, according to defined *rules*. These rules are specified by the system designer and are the driving mechanisms which influence the particular evolution of all the system cell's states. For example, the rule may state that if most of the neighbouring cells are in state 0, then the cell should also change to the same state. Because neighbouring cells can be found in a number of different state configurations, each CA rule is specified by a rule table with an entry for each local configuration, outlining the transition of that local configuration. For a fixed neighbourhood size (r), the number of entries in the rule table is finite (2^{2r+1} entries in rule table for neighbourhoods of radius r) and the space of rules is finite ($2^{2^{2r+1}}$ rules). The locality of cells causes the state transitions in CA cells to be affected not only by their current state, but also by the state of neighbouring cells. This feature often leads to the emergence of complex global patterns represented by all the system cell states, which result from the simple and local rules.

3.3.1.3 Mechanisms for Designing CA to Perform Computation

Although CA are suitable for studying the behaviour of natural systems, the application of CA as a parallel computing system suffers from the lack of a programming paradigm

or a general approach to constructing local rules that would enable global information processing to emerge and yield expected results.

Work addressing this problem can be found in [85], where an adaptive technique for programming CA is described. As a solution to this control problem, Packard proposes to apply genetic algorithms responsible for evolving CA rules towards the most fit ones, where the fitness is measured by the CA's ability to accomplish the following classification task. Given the initial configuration of the CA, the goal of the system is to decide whether the density of $1s$ in an initial configuration is greater than 0.5 or less than 0.5. Evolutionary mechanisms involving crossover and point mutation operating on CA cell rules lead to observation of two distinct types of rules being evolved with distinct global behaviour: *active* and *inactive*. The population of inactive rules makes the CA incapable to of transmitting information over the whole system, since they create static regions (within a system) that contain local cell state modifications forever. Very active rules, on the other hand, tend to communicate too much information to do successful computation. For these chaotic rules, rapid state transitions propagate over the entire system, where a given region is influenced equally by a large number of regions in the past. Given these characteristics, Packard observes that the rules that tend to be chosen by the adaptive process tend to lie near the region that marks the boundary between chaotic (active) rules and non-chaotic (inactive) rules. The observed behaviour is explained by the necessity of the system to be in a dynamic state in order to maintain efficient information flow, allowing the computation to take place between a number of independent and localised elements.

3.3.2 Artificial Neural Networks

Whereas in the previously described CA model we were explicitly focusing on the locality of interactions between neighbouring system elements, the model of neural networks that we describe in this section introduces another important characteristics of decentralised systems, that is networks of interactions.

This computing paradigm takes its inspiration from the functioning of neural networks, which are the main constructs of human and animal brains. As the name suggests, neural networks consist of simple and highly interconnected neurons, interacting with each other through the exchange of electrical impulses. The aim in designing artificial neural networks (ANNs) is to understand and exploit the functioning of underlying mechanisms which render these artifacts superior to any other existing computational machines, where their main characteristics are robustness to damage and real-time reactiveness to a continuously changing stream of sensory input.

3.3.2.1 Architecture

Artificial neural network models are often represented as networks of interconnected elements, which simulate the functioning of a biological neuron. Interactions between these artificial neurons (henceforth called neurons) are facilitated through connections, where each such connection has an associated strength (*weight*) and acts as an input for the neuron. It is assumed that each neuron can have more than one input and that exchanged information is represented as a simple scalar message. Because the functionality of a neural network emerges from the interactions between neurons, the functioning of an individual neuron can be described as follows. Upon receiving a signal from its inputs, the neuron performs a weighted sum of its inputs (based on the strengths of the connections representing the inputs), and then fires a binary signal if the total input exceeds a certain level. This signal may be received by other connected neurons and propagated further, eventually forming an output of the neural network processing task. Based on this, during ANN execution (usage), the input variable values are placed in the neurons that serve as data input units, which propagate signals to other neurons. When the entire network has been executed, the values received at the output layer values act as the output of the entire network.

3.3.2.2 Computation in Artificial Neural Networks

As in biological neural networks, the goal of ANNs is to map a range of input values into the output values, which represent a solution to a given problem. Before such a network will be able to effectively perform that task, it has to be trained, through a process of learning the strengths of connections between the neurons. There exist different learning strategies, where the learning process may take place even while the network is functioning. ANNs have been applied successfully to a number of prediction problems that are difficult for traditional computing architectures such as:

- classification, where the objective is to determine to which of a number of discrete classes a given input case belongs (eg. signature recognition); or
- regression, where the objective is to predict the value of a (usually) continuous variable (eg. tomorrow's stock price, the fuel consumption of a car, next year's profits).

Due to their recurrent interconnectedness and a lack of central control, neural networks are capable of exhibiting non-linear behaviour, where their information-processing is performed in dynamic conditions. Such a computation without stable states is a computing paradigm different from Turing's and introduces a number of interesting characteristics.

3.3.2.3 Mechanisms for Achieving Computation Relying on Artificial Neural Networks

Work described in [104] focuses on this aspect and investigates the computational capabilities of the non-linear dynamical system, represented by the neural network. Within this context, computation is defined as the mapping between received inputs and outputs in the classification domain problem.

The presented model consists of a neural network composed of 256 neurons with 33000 connections, which are then used to form a specific topology, with highly recurrent connections. The system dynamics and non-linear behaviour is controlled by the following parameters: the number of neurons; the number of incoming connections per neuron; the variance of the zero-centered Gaussian distribution from which the weights for the incoming connections are drawn; and the external input signal driving each neuron. Based on these control parameters, a network is assumed to exhibit chaotic dynamics if arbitrary small differences in a (initial) network state $x(0)$ are highly amplified and do not vanish at the current state $x(t)$. A totally ordered network, on the other hand, forgets immediately about the (initial) network state $x(0)$ and the current network state $x(t)$ is determined to a large extent by the current input $u(t)$.

Given this, the network performance is evaluated against the classification problem. Schurmann *et al.* observe that by pushing the network into dynamic, far from equilibrium state, the neural network performs the best when its dynamics is between order and chaos.

The same observation is reported by Bertschinger and Natschlager in [4], focusing on the relationship between the dynamical behavior of a system also represented by neural network, and its computational capabilities. To define this relationship, the authors calculate the critical boundary in the parameter space where the transition from ordered to chaotic dynamics takes place, and further show that only near the critical boundary can such networks perform complex classification tasks.

3.3.3 Swarming

So far we have presented two distinct examples of decentralised models based on which self-organising properties of natural systems can be studied, that is Cellular Automata (CA) and Artificial Neural Networks (ANNs). Although these models show how computation can be realised through local interactions, their application to autonomic computing is not straightforward. Whilst the first model suffers from too broad generality of its architecture to be directly applied to autonomic computing the latter one, on the other hand, is too neural-networks domain specific to be easily operationalised within the context of IT systems management.

In this section we consider the third example of self-organising system models that is inspired by the functioning of insect societies such as ant or termite colonies, fireflies, bees, bacteria or slime molds [112]. Models of these decentralised architectures are often described as *swarming systems*, where swarming relates to a 'useful self-organisation of multiple entities through local interactions' [94].

Although individual organisms within these systems may be considered as very simple and locally interacting entities, many biologists studying these systems [105] suggest that they possess collective information processing abilities the understanding of which, as we will show in the remainder of this section, has a potential to provide decentralised means of control suitable for autonomic systems [91, 134].

3.3.3.1 Decentralised Data Clustering

Work exploiting the swarming system characteristics is presented in [93], where Parunak *et al.* focus on the problem of data clustering based on its similarity defined through a similarity function. Most existing data clustering algorithms suffer from centralisation, where data structure and similarity function are centralised, and the requirement to preserve both the population of items being clustered and the similarity function to remain constant during clustering.

To overcome these limitations, Parunak *et al.* provide a decentralised data clustering algorithm, influenced by the behaviour of ants performing the same operation in their habitat. The natural algorithm works as follows. As ants wander about, they pick up objects with a probability u and drop them with a probability d. The probability u decreases, and d increases, with the objects' similarity to nearby objects. As objects move from regions where they are dissimilar to their surroundings to regions where they are similar, homogeneous clusters form.

In the adaptation of the ants clustering algorithm, the objects to be clustered are represented as active *content agents* (representing paragraphs of text) that are able to change their location by moving into different places. Because initially content agents are assigned randomly to a fixed set of logical *places* (distributed over multiple processors), the system's objective is to maximise the similarity among the content agents that occupy a given place. During information clustering, parallel decision-making conducted by simple agents is exploited. From an individual system element point of view, each content agent relies only on local information, where it compares its information content with the information content of a random sample of agents from the same and other places. Based on that, it probabilistically decides to change its place, where the probability increases with the increase in similarity that the move would provide. As a result of this decentralised decision-making process, conducted by a large number of simple agents, homogeneous places form, where information in each place represents similar concepts.

Experiments conducted on a simulation model show that a system composed of a large number of such active components reaches a high average place homogeneity very quickly, where its convergence speed has an exponential fit. Another interesting observation is that increasing the number of agents (and thus data to cluster) does not affect the exponential characteristics of the algorithm.

3.3.3.2 Analysis of Self-organisation in Swarming Systems

Similar observations are supported by work investigating self-organising properties of swarming systems, represented by an artificial model of a food foraging ants colony [29]. Based on this model, Guerin and Kunkle observed that the self-organisation process continuously stimulates the system to undergo transitions from structure (of foraging trails) formation, its maintenance, and finally decay. The ability of the system to dynamically discover new foraging trails and thus leap into optimal system configurations is driven by the availability of the food within the environment explored by the ants and the stigmergic coordination [121] ants utilise for collective food location and transportation. In achieving this flexible system adaptation, local information propagation through stigmergic mechanisms plays a crucial role, and is achieved by depositing digital pheromones (imitating chemical substances) into the environment by foraging ants that successfully located the food source. These pheromones attract other foragers and thus mark the route from the nest to the nearest food source, the strength of which is proportional to the amount of ants attracted by it and thus source of the food. As pheromones evaporate over a specific time period, routes that are not reinforced by ants decay and the foraging trail disappears. This naturally inspired coordination mechanism couples the activities of individual agents catalysing self-reinforcing and closed flows of information between collocated components. Although ants do not have an internal model of a foraging trail, and react to attracted pheromones, their collective activity results in ordered structures maintained by their local decisions.

Given this description, Gambhir *et al.* observe that too strong information propagation to the environment (stimulated by the deposited pheromone strength) locks agents into pathological tight loops, where agents wander in circles, attracted by each other's pheromones that, due to high strength, prevents evaporation before being reinforced. In this configuration, the system is totally inefficient at the foraging task due to never being able to transport collected food back to the nest. As a solution to this problem, Gambhir *et al.* limit the pheromone strength, such that it evaporates once the food is being exploited at the current location and before other agents are attracted by it.

3.3.3.3 Decentralised Graph Colouring

The dependence between information exchange between system components and performance of a large-scale agent system is also observed in other swarming architectures. In [12] Brueckner and Parunak investigate the graph colouring problem by applying a decentralised multi-agent system in which a fixed topology of networked agents represents graph nodes. The distributed graph-colouring model represents a class of agent systems in which the agents are only able to interact locally based on potentially incomplete or outdated knowledge. Such systems occur for instance in real-world applications that deploy large numbers of agents in a physical environment with limited resources available to the individual agent (e.g., swarming robotics, sensor networks and autonomic systems). In general, the graph colouring concerns colour assignment to individual nodes (out of a fixed set) such that the number of edges that connect nodes of the same colour is minimised. In the model in [12] agents perform local decisions about which colour to switch, based on the exchanged information communicated about the occupied colours from the neighbouring agents and local strategy that aims to minimise conflicting colours. The rate at which individual agents make their decisions (by randomly selecting agents from a population) is controlled by an activation level parameter (AL). Whilst changing the colour, the agent communicates this to the neighbouring nodes. To reflect noisy and time-constrained information exchange, delay of communication between agents is stimulated by a communication latency parameter (CL).

By increasing the frequency at which agents make decisions (AL parameter) above a critical value, Brueckner and Parunak observe that the emerging dynamics of the decision processes includes a robust phase change in system performance, defining three regions in which the system performs 1) 'better' than random', 2) 'worse' than random and 3) 'asymptotic' to random. For the first and last configuration, the population is able to solve a complex problem, but in the case of worse than random region, the system is in a thrashing mode that prevents the agents from finding and maintaining a good solution. As a solution to this problem, Brueckner and Parunak derive local metrics that allow agents to detect that the system is thrashing and by stimulating the rate of decision-making (AL) to suppress inefficient decision-making of individual agents contributing towards this pathological behaviour.

3.3.3.4 Resource Management Through Biologically Inspired Division of Labour

Regulation of labour is fundamental to the organisation of insect societies and is thought to be one of the principal factors in their ecological success [99]. Exhibited by adaptive allocation of individual colony members into a number of different tasks [28], such self-regulatory processes offer a decentralised means of control that, if sufficiently understood, can become an efficient way of constructing artificial self-organising systems.

An example of such a process can be observed in an ant colony [7] where from a population of homogeneous ants, each capable of handling the same number of tasks, a system proliferates into distinct but organised collectives (castes) of elements. Each such collective specialises at carrying out specific tasks such as food foraging, nest building, brood feeding, nest defense, etc. The survival of the colony depends on both the efficient handling of each system task and the adaptive division of resources (ants) into a number of such collectives achieving different tasks. For example, if there is more brood to feed, food foraging needs to be carried more efficiently; if the colony is expanding, more builders are required to carry out the nest expanding; if the colony is under attack, a group of food foraging ants needs to be minimised and recruited for the nest defense. One of the most striking aspects of such a self-regulatory response achieved by division of labour is plasticity, a property achieved through the workers' behavioral flexibility: the ratios of workers performing the different tasks that maintain the colony's viability and reproductive success can vary (i.e., workers switch tasks) in response to internal perturbations or external challenges [7].

Understanding how this flexibility is implemented at the level of individual workers which certainly do not possess any global representation of the colony's needs has been addressed in [120, 7, 73], revealing that self-regulatory properties of insect colonies stem from simple threshold based responses of system elements to perceived demand, in which specialisation of system elements to handle particular tasks arises as a result of a reinforcement process, the principles of which are explained in [120]. Here, Theraulaz *et al.* present a simple stimulus-response threshold model where m tasks within a system are associated with stimuli or demands, the levels of which increase if they are not satisfied (because they are not performed by enough individuals or at high enough rates). Likewise, system elements capable of handling any of these tasks maintain a list of response thresholds, each for a particular task. These thresholds represent the likelihood of reacting to task-associated stimuli where low threshold individuals perform tasks at a lower level of stimulus than high threshold individuals. Given this, within individual workers, performing a given task induces a decrease of the corresponding threshold, and not performing the task induces an increase of the threshold. This reinforcement process leads to the emergence of specialised workers; that is, workers that are more responsive to stimuli associated with particular task requirements from a group of initially identical individuals.

A number of self-organising models exploring labour division based on such threshold reinforcement mechanisms are presented in [94, 16, 110]. In [94] Parunak and Brueckner propose a stigmergic coordination mechanism to preserve energy resources within a power constrained mobile ad-hoc network achieved by activation and deactivation of servers according to the locally perceived demand. In [110] a decentralised system model exploring the process of division of labour in a honey bee colony is presented by Robinson *et al.* Another example of an ant coordination mechanism is introduced by Cicirello *et al.*

in [16] and applied to the distribution of different tasks among a set of servers adapting to the perceived demand.

3.4 Discussion

Irrespective of whether these systems are abstract models of interacting cells, imitations of a swarm of insects or a chemical system, the exploitation of their computational capabilities requires facilitation of both *function* and *control*. Whereas the first relates to the mechanisms through which the system is achieving the required tasks (i.e. performing stock market price prediction or managing distributed resources over a network of distributed mobile nodes), the latter is concerned with the processes controlling the reliable provision of the functionality despite perturbances and varying functioning conditions.

The process through which function and control is facilitated in these models is related to self-organisation of system elements into collectives (otherwise called organisations) [37], characterised by robustness and resilience to perturbations and single component failures [129]. As the ability to robustly provide function and control through a set of interacting autonomic elements is at the heart of autonomic computing, below we characterise the means through which these processes are addressed in decentralised models.

3.4.1 Information Flows

Recent insights into understanding the functioning of decentralised models have revealed that the organisation of localised system elements into collectives handling particular functions and self-regulatory activities is subject to their behavioural stimulation achieved either by direct or indirect information exchange [57, 7, 95]. Such information dissemination, stimulated independently and in parallel by autonomous system elements, gives rise to self-sustaining flows of information across the system that, in turn, are assumed to play the key role in organising the whole system [91].

Although within emerging networks of interacting agents no system element has the ability to control information exchange beyond its locality, it is the information *perception*, *processing* and *propagation* performed by each agent that stimulates the global system functioning and gives rise to function and control. Below we characterise these three activities.

3.4.1.1 Information Perception

Elements acquire new information by interacting with other components as well as with the environment they sense. Such interactions have a certain locality, preventing each

component from obtaining and operating on full knowledge about the current system state. As a consequence, information acquired by individual system elements represents only a local view of the current system state. Here, the locality stems from constraints imposed by the time, cost to transmit information and the system dynamics that causes information to become stale as soon as the system configuration changes [105].

3.4.1.2 Information Processing

Given this limited scope of available knowledge to individual system elements, the mechanisms governing their behaviour do not involve complex reasoning, but exhibit simple and reactive *stimulus-response* properties. For example, in CA, state change rules are designed in such a manner that each perceived state change of a neighbouring rule is mapped to a new state the cell switches, thus propagating this change to other nearby elements. In artificial neural networks (ANNs) each neuron, upon receiving a signal (and based on the current weight and threshold parameter), propagates it further to the interconnected nodes. In a more sophisticated model, represented by swarming systems such as ant colonies, the acquired information acts as a 'potential field' attracting individual elements to perform certain tasks, such as food foraging, brood feeding or nest construction. Again, individual ants do not perform complex decision-making on the perceived information, but rather act as automata with reactive strategies dependent on the information signaled and their location.

3.4.1.3 Information Propagation

Information processing may lead to the modification of perceived information and thus dissemination of filtered or modified knowledge to other system elements. For example, in the case of a swarming system involving ants, the individual ant sensing pheromones will be attracted by it (if they are sufficiently strong) and thus modify its behaviour accordingly. This behavioural modification affects also what information the ant propagates to the environment or other elements in the future. In this case, an ant that has been attracted by a food foraging trail will actively forage and deposit information, encouraging other ants to follow the same process. In ANNs perceived stimuli will be propagated only if they exceed a certain propagation threshold. The effects of this process, analysed from an individual's perspective are understandable. However, when a set of components interacting in parallel is considered, propagation of certain information may lead to the adjustment of the behaviours and willing contribution in this self-reinforcing process of other elements, until all system components (within a certain local system vicinity) form a group of coherently acting components. This feature, often described as *autocatalytic potential* or positive feedback [11, 91] is said to play key role in achieving the system's self-organisation.

3.4.2 Bottom-up Information Flow Regulation

The existence of information flow within the system does not suffice to organise system elements into collectives that provide the required functionality. Both the manner in which information is communicated between localised elements and how this process is being regulated through local decision-making mechanisms have significant impact on the arising information flows and thus the ability of the system to self-organise or drift between ordered or chaotic states. These observations, shared across such decentralised system models are described below.

3.4.2.1 Dynamic Interaction Topologies

Traditional software systems conform to a static model of computation, where it is performed by the flow of information through static and pre-imposed at design time interactions between the system components. Within the reviewed decentralised architectures, on the other hand, computation involves and is inherently related to the modification of their internal structure or organisation, often defined through relationships and interactions between system elements. Because of this, the system responds to the environmental changes by reorganising its structure and thus adapting to new conditions. For example, in the case of CA, computation is performed through the modification of states of individual cells, which lead to the formation of specific global patterns that exhibit information-processing capabilities. The same local behaviour is observed in artificial neural networks, where strengths of associations between simple neurons play crucial role in the ability of the network to perform complex classification tasks. Finally, swarming systems also achieve their global goals through reconfiguration of their organisational structure reflected by the proliferation of ants into collectives that handle different system level tasks.

3.4.2.2 Local Regulatory Mechanisms

As there is no global controller directing the decisions of individual system elements, the control over the arising interaction topologies and thus the manner in which information flows across the system is devolved to individual system elements. For example, Packard [85] showed that it is the role of local rules embedded within individual cellular automata that, by appropriately regulating the information exchange, sustain the organisation of the whole system and prevent its descent into a chaotic state. In this context, he observed that the population of evolved 'active' rules tended to communicate too much information to perform successful computation, leaving the system in rapid state transitions propagating over the entire system, whereas 'inactive' rules were incapable of transferring information over the whole system. The best system efficiency was

achieved with rules that existed near the region that defines the transition from chaotic to non-chaotic ones.

Similar observations are supported by Guerin and Gambhir [24] investigating self-organising properties of a swarming system represented by the model of the food foraging ants colony. Here, the authors show that propagation of strongly biased information to the environment (through digital pheromones the foraging ants deposit) locks agents into pathological tight loops. In this configuration, the system forms organised and circular foraging routes that are totally inefficient at transporting food back to the nest. By limiting the strength of pheromones too much, on the other hand, the system is incapable of forming any structures, since information evaporates before it is reinforced by ants, leaving individual foragers to randomly explore the area. Similarly to the case of the most effective CA's rules residing at the boundary between ordered and chaotic system behaviour, the most effective pheromone strength parameter value in ant-like system is set to reside between two observed behavioural extremes.

3.5 Thermodynamics of Self-organisation

In all the presented models system efficiency is proportional to its ability to achieve order and structure. During this self-organisation process, individual system elements transition from initially disorganised and chaotic interactions into those that are distinguishable and traceable through stable topologies, viewed either as isles of similarly configured cells within CA or collectives of ants forming a foraging trail within a swarming system.

To achieve such organisation, various techniques and local-decision making mechanisms have been applied. For example, Packard applied evolutionary algorithms to 'breed' the most efficient local rules, whereas Guerin and Gunkle explored simple *stimuli-response* model parametrisations to encourage an ant colony model to achieve the food foraging task. As a consequence, although complex system models presented above show interesting self-organising capabilities, they are achieved at the cost of large amount of experimentation and model tinkering. Whilst this is not a main issue addressed within these efforts, the application of such complex systems to preserve control over autonomic computing infrastructures requires a more principled and methodological approach.

In particular, given the fact that complex systems exhibit emergent behaviour that, as we have seen, is the motive force for achieving organised (or chaotic) system response, a careful understanding of such a phenomenon is required. For this purpose, not only do we need to provide adaptive decision-making mechanisms, but more importantly, we need to understand why and under what conditions such mechanisms lead to increased system efficiency. Understanding this bottom-up self-organisation phenomenon will allow not only for construction of artificial self-organising models but, more importantly,

engineering of dependable and reliable autonomic systems.

To advance the current state of research in this direction, we consider a study of thermodynamics and self-organisation as key investigation areas. According to thermodynamics of self-organisation (described in Section 2.5), the increase in system organisation can be understood as the emergence of constraint imposed on individual system elements that arise as a result of energy flow across the system's boundary. If properly regulated, such a flow displaces the system from equilibrium and allows useful work to be extracted from the system. From a thermodynamic viewpoint, work extraction is not a result of intelligent acting of system constituents (since they may be simple molecules) but results from an end-directed behaviour of such elements aiming to dissipate the energy-related gradient and push the system back to equilibrium.

Although the research focusing on thermodynamics of self-organistion within artificial systems is at its infancy, there have been several approaches (presented in Section 2.5.8) that extend thermodynamic analysis to artificial computational models, revealing that the same patterns of behaviour can be observed within artificial systems.

In the remainder of this thesis, we will extend this analysis and interpretation within the context of autonomic computational systems design. In doing so, to approach the three control problems introduced in Section 1.2 through bottom-up control mechanisms, we will employ open multi-agent systems to model decentralised autonomic systems and analyse their response whilst introducing decision-making processes that impose constraint, generate gradient and exploit that gradient for a useful work extraction. Within such systems, rather than considering energy flows, we will focus on the role of information flows across an open system's boundary and between its individual autonomous elements.

As such flows within natural systems are responsible for structure formation and its robust maintenance, we will investigate under what conditions will agents self-organise and, if so, what impact will such organisation have on the efficiency of the system in achieving three of the aforementioned system-level functions. During the experimental analysis we will reveal what processes and decision-making mechanisms are critical at achieving global system stability and organisation and that are consistent with the principles of self-organisation discussed in Section 2.5. Finally, we will extend the thermodynamic interpretation to the class of decentralised multi-agent systems and provide general design principles aiding the design and engineering of such artificial self-organising systems.

Chapter 4

Modelling an Autonomic System

4.1 Introduction

In the previous chapter we introduced the general characteristics of complex natural systems, including their multi-level architecture, emergent behaviour and self-organising features responsible for adaptive and robust behaviour of these systems. Following this, we presented the existing state-of-the art in applying these decentralised models to perform adaptive and efficient computation on behalf of human operators. Finally, we suggested thermodynamics and self-organisation studies as a necessary route to advance the understanding of self-organising properties exhibited by these systems as well as their principled engineering within the context of autonomic computing.

In this chapter we offer an introductory study of applying self-organisation to preserve control over autonomic computational systems. To do so, we present a minimialistic multi-agent system model provided with simple local decision-making mechanisms and confront it with the problem of load-balancing within a controlled resource allocation environment. Based on the experimental results, we then analyse system efficiency and its ability to establish constraint on the behaviour of its constituents that, according to thermodynamics, identifies organised system state in which useful work can be extracted.

Since the main aim of this chapter is to introduce decentralised multi-agent system control into autonomic systems and study its self-organising potential, we focus our attention on the simple thermodynamic analysis of self-organisation observed within the autonomic system model and do not offer any ready-to-use solutions for any particular resource management problem. In the remainder of this thesis, this analysis is extended to a more advanced and realistic autonomic system model, the self-organising properties of which are analysed in a range of more challenging and realistic problems.

To this end, the chapter is organised in the following manner. In Section 4.2 we describe the load-balancing problem that we will approach, relying on the minimialistic autonomic

system model described in Section 4.3. Empirical results, evaluating system performance are presented in Section 4.4, whereas the thermodynamic analysis of system behaviour is conducted in Section 4.5. The chapter concludes with discussion presented in Section 4.6.

4.2 Load-balancing Within a Minimalistic Multi-agent System Model

Achieving load-balancing (defined in Section 1.5) in a decentralised manner represents a non trivial challenge that, if not properly approached, may result in poor system scaleability or an unexpected loss of the system performance. In particular, it has been observed [40] that the introduction of shared and constrained resources into a system composed of a large number of independent and concurrent components, responsible for offering and consuming resources on behalf of system users, may quickly and unexpectedly lead to the emergence of undesirable system behaviour, often described as *resource competition* [94]. In this state, consumer agents end up competing for specific subsets of resources, leaving others under-utilised. This results in poor global resource utilisation reflected by inefficient load-balancing over the set of available resource providers. Furthermore, because there is no centralised control, the system may simply self-reinforce this competitive behaviour leading to a very rapid degradation of system performance [41].

A close investigation of this problem shows that inefficient resource utilisation on the system scale arises from poor decisions made by individual system elements concerning their selection of resources. This presents us with the problem of facilitating local decision-making mechanisms capable of suppressing competitive interactions between resource consuming elements. Where competition cannot be avoided, for instance where demand for resources outstrips supply, such resource allocation mechanisms should effectively distribute service provision across the system, thus preserving fairness. As these mechanisms concern the manner in which resources are selected by consumers, in the remainder of this chapter we will refer to them as resource allocating mechanisms.

In the remainder of this chapter we will approach this problem with a minimialistic multi-agent system model in which agents are required to adaptively balance the consumption of available within the system resources such that inefficient resource competition is avoided. Since the intention of such a design is to investigate how global system stability and efficiency can arise as a result of local interactions and simple decision-making mechanisms, we do not aim to provide a ready-to-use solution to the load-balancing problem but offer a thermodynamic analysis of observed bottom-up system organisation.

4.3 Simulation

4.3.1 Model Design

The autonomic system model that we will explore in this chapter is based on a multi-agent system architecture that comprises:

- a service registry, that serves as an inventory of the resource providers within the system;
- a population of S_N agents representing resource providers (services);
- a population of U_N agents representing resource consumers (consumers).

Services are provided by agents that facilitate access to resources (disk storage, CPU time, etc.)[1]. Each service has a *type* and a *capacity*. For example, $S_1{:}A_{10}$ represents a service provider (identified by the prefix S and a unique instance number, 1) of type A and capacity 10. Such a service is able to simultaneously satisfy a number of resource consumers requesting up to a total of 10 units of resource A.

Consumers consume resources according to a personal workflow defining the type, capacity and order of services required. For example, $U_1{:}W_1$ represents a consumer (identified by the prefix U and a unique instance number, 1) with workflow W_1. Workflows are represented by an *ordered* set of service demands. For example, $W_1 = \{A_{10}, B_5, C_{15}\}$, where the number next to each service type indicates resource capacity required by this particular service type. Here, we assume that consumers may share identical workflows, and that, moreover, different workflows may demand the same service types. As such, consumers may be in competition with one another for the same system resources.

The registry is an agent tasked with maintaining an inventory of system services. When queried by a consumer, it supplies a list including any and all services with a type that matches the service required by any of the consumers workflow components. This list is constructed in set order.

Service Allocation is performed independently by each consumer. For each workflow, the agent first obtains from the registry a list of existing services capable of providing any of the resources required by the workflow[2]. This action takes time T_x and incurs an execution cost, C_x. Repeatedly, services are chosen from this list and their availability determined (each time incurring a query cost, C_q, and consuming time, T_q). Services

[1] We assume that each agent can facilitate access to only one service, and refer to such agents as "services" in order to distinguish them from service consumer agents.

[2] In reality the information obtained from such registries can become stale and unreliable over time since resources are continually leaving and joining the system. Ultimately, we are interested in systems where this unreliability is sufficient to ensure that agents prefer not to make use of it at all.

may be unavailable because they are busy to the extent that they do not have the spare capacity required by the workflow component, or because they are no longer part of the system. Once an available service has been located, the agent attempts to allocate the next component of its workflow. Once all components of the workflow are allocated to services in this way, the agent attempts to execute the workflow using these services. Since services are not locked during the allocation process, it is possible that a consumer agent may allocate a workflow component but find that the service is busy when it attempts to execute it. In such circumstances, the consumer must re-allocate this workflow component. Successfully executing a component also takes time, T_x, and incurs an execution cost, C_x. Should a service fail during execution, the consumer still pays the execution cost, but must also re-allocate the workflow component. If, during any allocation process, a consumer makes n attempts to locate an available service of a particular type, the allocation is deemed to have failed, as is the workflow it is part of. Here, for each workflow component to be allocated, we set n equal to the number of services of the required type returned by the registry. Whether successful or unsuccessful, upon completing a workflow, a consumer agent, U_i is inactive for some randomly determined period drawn from a uniform distribution $[0, \omega_i]$ after which the same workflow allocation process begins again.

Given the costs involved in the allocation process, we can define an optimal level of performance against which to compare system behaviour. The cost to an individual consumer of completing a workflow W_j can be written as

$$C(W_j) = C_x + \sum_{i=0}^{|W_j|} f_i C_q + (g_i + 1)(C_q + C_x)$$

Here, f_i and g_i are, respectively, the number of failed attempts to locate an available resource and the number of failed attempts to execute a located resource, for each workflow component, i. A workflow is achieved with minimal cost where $\forall i, f_i = g_i = 0$, giving

$$C^*(W_j) = C_x + \sum_{i=0}^{|W_j|} (C_q + C_x)$$

For a system of consumers and services involving W_N unique workflows, where the proportion of consumers attempting to complete workflow W_i is p_i, the optimal system cost, C^*, can be written as

$$C^* = \sum_{i=0}^{W_N} C^*(W_i) p_i$$

Time is not represented explicitly within the system, as the simulation takes place in real time, with an independent parallel thread associated with each consumer, each service and the registry. As such, agent activity takes time, which is a limiting physical property of the system. Moreover, system update is truly asynchronous, which avoids the possibility that components might become phase locked, or other artifacts that may result from discrete, synchronous time updating [43, 31].

To realise this, the underlying multi-agent system model is constructed such that every consumer agent is a an independent software thread [3] driven by its own internal control mechanism. As a result, all consumer agents asynchronously and in parallel conduct their resource allocation decisions whilst searching for available resources. During this process the time it takes to allocate is proportional to the number of requests the consumer agent performs until it finds an available service.

Scenarios explored here all conform to the following specification, unless stated otherwise. An equal number of agents repeatedly attempt to execute each of three different workflows for a period of time, T_N. Since we expect simple workflows to be requested more frequently, the maximum period of time for which a consumer will sleep after completing (or failing to complete) a workflow, ω, is workflow-dependent such that more complicated workflows tend to be associated with longer periods of sleep:

$$W_1 = \{A_{10}\},\ \omega_1 = 1\text{s}$$

$$W_2 = \{A_{10}, B_{10}\},\ \omega_2 = 2\text{s}$$

$$W_3 = \{A_{10}, B_{10}, C_{10}\},\ \omega_3 = 3\text{s}$$

For all agents, the following values are assigned to the costs and execution times of service interactions and service executions: $C_x = 20$ units, $T_x = 500$ms, $C_q = 10$, $T_q = 100$ms. Based on the equations derived above, these parameters define minimal costs for a consumer attempting to execute each workflow: $C^*(W_1) = 50$ units, $C^*(W_2) = 80$ units, $C^*(W_3) = 110$ units. Since the proportion of consumers attempting to execute each workflow is the same, the optimal cost for a system can be calculated as $C^* = 80$ units.

4.3.2 Consumer Strategies

Consumers rely on *strategies* to guide their individual behaviour. Here we explore minimally sophisticated strategies separately and in combination.

- **Null strategy** ($\emptyset$): when attempting to allocate a workflow component to a service, the consumer agent proceeds to select services from the list of available resources obtained from the registry, in order.

[3] The system model is written in Java programming language.

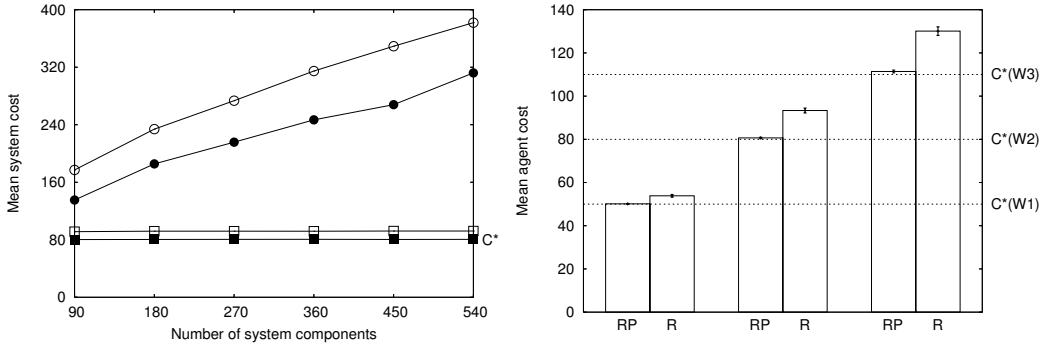

FIGURE 4.1: The impact of system size on global (left) and local (right) system costs. Left: mean system cost for representative runs of four strategies, $\emptyset$ (empty circle), $\mathcal{P}$ (solid circle), $\mathcal{R}$ (empty rectangle) and $\mathcal{RP}$ (solid rectangle), where a dotted line (C^*) corresponds to the optimal cost (in each case consumer demand matches service provision such that $U_N : S_N = 1 : 2$). Right: mean workflow completion costs for representative runs of systems of 540 components ($U_N = 180$, $U_S = 360$) for three workflows, W_1, W_2 and W_3, where dotted lines ($C^*(W1)$, $C^*(W2)$, and $C^*(W3)$) correspond to the optimal costs for each workflow. In each case $T_N = 400$ seconds.

- **Random selection** ($\boldsymbol{\mathcal{R}}$): attempting to allocate a workflow component to a service, the consumer makes random choices from the list of available resources obtained from the service registry.

- **Preferential selection** ($\boldsymbol{\mathcal{P}}$): when attempting to allocate a workflow component to a service, the consumer preferentially returns to the last service employed for that component, if it was executed successfully during the last workflow, otherwise the null strategy is employed.

- **Hybrid strategy** ($\boldsymbol{\mathcal{RP}}$): when attempting to allocate a workflow component to a service, the consumer preferentially returns to the last service employed for that component, if it was executed successfully during the last workflow, otherwise the random selection strategy is employed.

We do not expect these simple strategies to be employed within real autonomic systems. However, the simplicity of the randomising and canalising behaviours that they employ, both separately and in combination, make them good candidates for examination, since these processes are considered by thermodynamics studies to play key roles in achieving system self-organisation. To understand how these processes achieve global system efficiency within the context of the load-balancing problem, below we provide an experimental evaluation that is then followed by a thermodynamic analysis, explicitly focusing on constraint imposing/relaxing features of the proposed mechanisms.

4.4 Results

Here we characterise the behaviour of the minimal simulated system, concentrating on the manner in which system performance scales with four system parameters: *size*, *heterogeneity*, *load* and *reliability*. In each case, we are interested in both the efficiency of the system as a whole, and the efficiency of the consumer agents within it. The former is measured over a specific test period by calculating the average cost per executed workflow. This measure makes sense where consumers never (or rarely) fail to execute a workflow. At the level of individual consumers, we are interested in any advantage that one class of consumers (say those attempting to execute a simple workflow) might have over another. In each case, we are interested in how different consumer agent strategies impact on these measures.

4.4.1 System Size

First, we examine how the system responds to an increase in the number of consumers and services. Figure 4.1(left) illustrates the relationship between mean system cost and user strategy as the number of system components is varied. The results illustrated in this figure were achieved through consecutive model runs, where at each run a number of consumer and provider agents was increased such that the demand-supply ratio remained in balance but the size of the system increased.

For consumers employing the null strategy, system cost increases linearly with system size. Consumers employing a simple preference for remembered services ($\mathcal{P}$) show a slight improvement, but the system cost still scales linearly with system size. The introduction of randomised selection of services $\mathcal{R}$ improves performance in two senses. First, system cost is now unaffected by system size. Second, the level of system efficiency approaches optimal levels (recall that $C^* = 80$), particularly when randomised selection is combined with a preference for remembered services $\mathcal{RP}$.

How does the combination of two *extremely* simple mechanisms result in impressive performance that is robust to increasing system size? When employing the randomised strategy, selections of resources by consumers disperse, avoiding the resource competition that results from the $\emptyset$ strategy. When $\mathcal{R}$ is combined with $\mathcal{P}$, consumers are able to remember and return to services that have been involved in successfully executing a workflow, attenuating dispersal. By discouraging consumers from continuing to randomly choose from the same set of services, preferences reduce the possibility of conflict.

Figure 4.1(right) depicts the mean workflow completion costs for consumers relying on the two most successful strategies for the largest system size depicted in Figure 4.1(left).

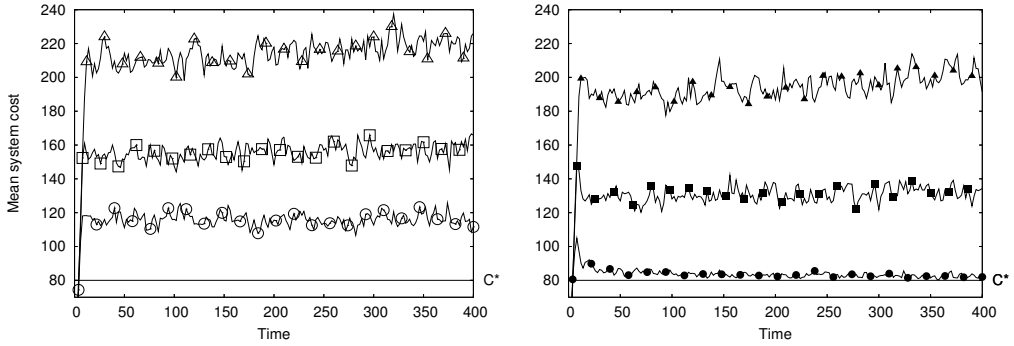

FIGURE 4.2: Relation between mean system cost and system load for agents relying on $\boldsymbol{\mathcal{R}}$ (left) and $\boldsymbol{\mathcal{RP}}$ (right). Three levels of system load are represented: $L = 1$ (circle), $L = 2$ (box), $L = 3$ (rectangle). The dotted line (C^*) corresponds to optimal system cost. In each case, $S_N = 240$, while U_N is varied from 120 through 360.

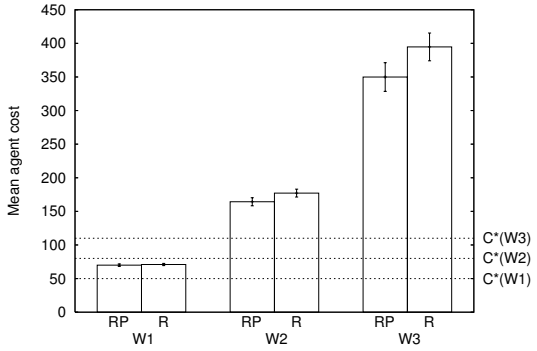

FIGURE 4.3: Mean workflow completion costs for representative runs with $\boldsymbol{\mathcal{R}}$ and $\boldsymbol{\mathcal{RP}}$ under increased system load ($\boldsymbol{L = 3}$). Dotted lines represent optimal costs for each workflow. $U_N = 360$, $S_N = 240$, $T_N = 400$ seconds.

For both strategies, the average departure from optimal performance increases linearly with workflow size. However, this departure is extremely small for the $\mathcal{RP}$ strategy.

4.4.2 System Load

Up to this point, the provision of system resources has matched the demand of system consumers: there exists a potential allocation of services where every workflow component can be executed simultaneously and every system service is fully occupied. In principle, this situation allows an allocation process to exhibit convergent behaviour. Since real utility computing infrastructures must cope with variation in both the level and type of demand for (and provision of) services, demand may sometimes outstrip supply, precluding such an allocation.

To investigate the impact on system behaviour of variation in the balance between supply and demand, we vary the system load, L, defined as the ratio of consumer demand to service provision. For a system where $L = 1$, in principle every workflow component can be simultaneously satisfied by system services. Doubling this load ($L = 2$) ensures

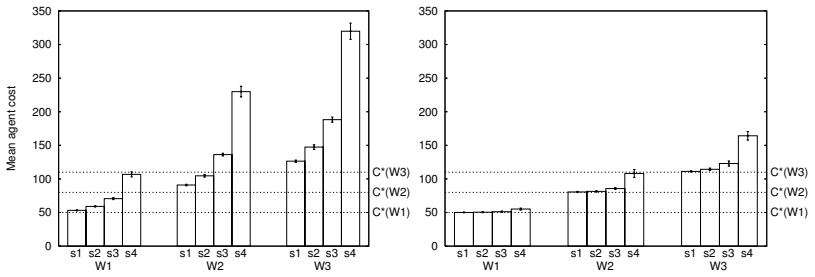

FIGURE 4.4: Mean workflow completion costs for agents relying on $\boldsymbol{\mathcal{R}}$ (left) and $\boldsymbol{\mathcal{RP}}$ (right) where $\boldsymbol{H = 4}$. Within each workflow type, four subclasses are identified in order of increasing capacity requirement $(s1, s2, s3, s4)$. Dotted lines correspond to the optimal cost for each workflow group. $U_N = 180$, $S_N = 360$, $T_N = 400$ seconds.

that only half of the consumers' workflow components can be executed simultaneously. Here, system load is manipulated by holding the number, type and capacity of system services constant, and varying the number of consumers (but not the proportions of different workflows being allocated). Doubling the number of consumers thus doubles system load.

Scenarios are identical to those described in Section 4.3.1 save that there is heterogeneity in the demand for capacity across workflows:

$$W_1 = \{A_{10}\}$$

$$W_2 = \{A_{11}, B_{11}\}$$

$$W_3 = \{A_{12}, B_{12}, C_{12}\}$$

How does the system respond to increasing load? Figure 4.2 illustrates the relationship between mean system cost and load for the two most successful strategies. The ability of $\mathcal{RP}$ to approach optimal performance where supply matches demand $(L = 1)$ is lost for higher system load, and the advantage it enjoys over $\mathcal{R}$ is reduced. Neither strategy can cope with the increased number of "collisions" during resource allocation that result when consumers can no longer utilise preferred services exclusively.

Recall that consumer costs increase linearly with workflow size for systems where supply meets demand (see Figure 4.1 (right)). By contrast, Figure 4.3 demonstrates that mean workflow completion costs accelerate with workflow size for both strategies when system resources fail to match consumer demand $(L > 1)$. The costs of competition for scarce resources are being borne disproportionately by consumers with more workflow components to allocate.

4.4.3 Consumer Heterogeneity

It is highly unrealistic to assume that agents attempting to allocate the same type of resource will also share exactly the same service preferences. For example, in the domain of utility computing, different amounts of CPU processing power, storage size, quality of service, etc., may be required. Hence, for each consumer, only a subset of services of a particular type will be capable of satisfying its particular demands. Since many of these attributes are dynamic properties that may change rapidly and unpredictably, it may be that a centrally maintained registry of services cannot be relied upon to provide the information required by consumers to identify appropriate services.

In order to manipulate the degree of heterogeneity in consumer demand, H, within the model we assign different capacity requirements to consumers and differing capacity provision to services. A consumer will be satisfied by any service of the required type with free capacity that either equals or exceeds its capacity requirements. As such, consumers with high capacity demands must necessarily have at least as difficult an allocation task as consumers with lower capacity demands. In all cases considered here, there exists an allocation of services to consumers where all available service capacity is utilised in executing every workflow component simultaneously (i.e., supply always exactly matches demand), and no service has capacity to simultaneously execute more than one workflow component. We define H as the number of unique levels of service capacity required by the workflows of a consumer population (or, equivalently, the number of unique levels of capacity provided by a population of services). Thus, for $H = 1$, all workflows share the same capacity requirements, whereas for $H = 2$, each workflow (and every service type) is present in a low-capacity and high-capacity variant such that each variant is assigned to an equal number of consumers. The scenarios reported below are otherwise identical to those described in Section 4.3.1, with systems comprising of 180 consumers and 360 services.

For consumers employing the random selection strategy, $\mathcal{R}$, there is a significant increase in cost, and in its variation, as the degree of consumer heterogeneity increases. While $\mathcal{R}$ preserves a uniform distribution of resource requests among a group of services of the same type, and thus effectively minimises the number of conflicts, it is *blind* to the various levels of capacity offered by services. By contrast, the $\mathcal{RP}$ strategy, which combines $\mathcal{R}$ with preferential selection of previously utilised services, delivers a significant improvement in performance. Here, the system converges to a near optimal allocation of services to consumers, effectively matching consumer capacity demands to service capacity provision from a global as well as local perspective. This convergence is achieved, even though there exists a potential for conflict between low- and high-capacity consumers over who gets to utilise high-capacity services.

These observations are confirmed by Figure 4.4 which depicts the mean workflow completion costs for a high degree of consumer heterogeneity ($H = 4$). For each workflow,

four levels of capacity demand are introduced: $\{s1, s2, s3, s4\}$. While high-capacity workflows tend to attract higher allocation costs, irrespective of consumer strategy, the departure from optimal allocation costs is much reduced for $\mathcal{RP}$ strategists. Consumers adopting this hybrid strategy were thus able to form preferences for resources that not only satisfied their own resource demands but, in the case of low-capacity consumers, also contributed to satisfying the demands of their competitors, increasing overall system efficiency.

4.4.4 Service Reliability

A further distinguishing characteristic of utility computing infrastructures is the lack of assurance that existing services will not fail or become unavailable during a consumer's lifetime. To investigate the impact of resource failure, randomly selected services are removed from the system at a constant rate. The scenario is initially identical to that described in Section 4.4.2, with 360 unallocated services in principle exactly matching the demand of 180 consumers. However, after a 40-second period of normal service allocation during which time the system settles to its typical behaviour, services begin to be removed at random at a rate of one per second, until none remain.

Figure 4.5 illustrates the manner in which mean system cost varies over time in such a scenario, both for $\mathcal{R}$ strategists and $\mathcal{RP}$ strategists. Over the majority of the simulated period, the $\mathcal{RP}$ population enjoys an advantage over the $\mathcal{R}$ population in terms of allocative efficiency. However, this advantage decreases over time. For each population, costs rise with increasing service failure at an accelerating rate, until a catastrophe is reached at around 360 seconds. At this point, certain types of resource are no longer present within the system, preventing some workflows from being completed successfully. By 400 seconds, all consumers are paying a cost associated with accessing the registry, but are failing to carry out any allocation activity.

For the scenarios simulated here, over time, as services fail, system load increases. It is instructive to compare the mean system cost that results from service failure to that reported for the same constant system load. Prior to the first resource failure at $t = 40$, system load has been stable at $L = 1$. Subsequently, as resource failure increases system load, it is remarkable that consumers are able to achieve an allocative efficiency equivalent to a system under constant load for load values as high as $L = 12$. Despite the scale of the system and the simplicity of the resource allocation mechanisms, it is evident that the allocative reconfiguration required by increasing load can be achieved smoothly and efficiently.

As noted above, as the number of failed resources (and thus system load) increases, the advantage that $\mathcal{RP}$ has over $\mathcal{R}$ in terms of allocative efficiency diminishes until, under extreme system load, it disappears. This result can be interpreted as indicating that

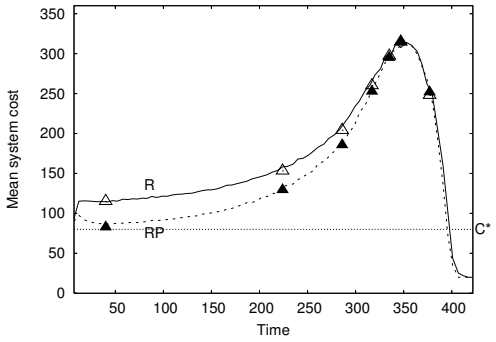

FIGURE 4.5: Relation between mean system cost and degree of resource failure for consumers relying on $\boldsymbol{\mathcal{R}}$ (solid line) and $\boldsymbol{\mathcal{RP}}$ (dotted line). Initially, $U_N = 180$, $S_N = 360$. From the 40^{th} second, one randomly selected resource fails permanently each second. Symbols indicate the mean system cost experienced at an equivalent constant load, calculated over a window 300s $< t <$ 400s, for load values drawn from $\{1, 2, 3, 4, 5, 6, 12\}$.

the role of preferential selection within the $\mathcal{RP}$ strategy also diminishes over time, with behaviour increasingly dominated by random selections at high system load.

4.5 Thermodynamic Interpretation

Neither $\mathcal{R}$ nor $\mathcal{P}$ were able to efficiently and fairly allocate services to resources in all of the scenarios explored here. While $\mathcal{P}$ performed poorly in every test, the relative strength of $\mathcal{R}$ quickly diminished when confronted with a degree of consumer heterogeneity. It is therefore perhaps surprising that a hybrid strategy, $\mathcal{RP}$, combining both simple strategies yields convergent behaviour that is more efficient in every test.

Within scenarios where there is competition for scarce resources, an interesting interplay between the $\mathcal{R}$ and $\mathcal{P}$ elements of the hybrid strategy arises. As Figure 4.5 shows, as long as competition for resources is weak, stable consumer preferences can be established and exploited. These allow the system to converge and exhibit near optimal behaviour. However, once the system load increases, the difference in performance between $\mathcal{R}$ and $\mathcal{RP}$ disappears. This suggests that, within the hybrid population, the proportion of time spent relying on the $\mathcal{P}$ element of the strategy gradually reduces, until agent behaviour is dominated by the $\mathcal{R}$ element.

It is important to note that this gradual change is motivated by increasing pressure within the system. When this pressure is held constant at a particular level, as it was in the system load tests, mean system cost stabilises rapidly, despite individual agent behaviour alternating between the $\mathcal{P}$ and $\mathcal{R}$ strategy elements. Even at high loads, the system does not exhibit oscillatory behaviour during this stabilisation. This suggests that the system is able to adaptively balance the elements of the hybrid strategy in response to particular levels of demand. In some sense, the strategy also ensures that this balance is not achieved at the expense of fairness within the system, since results

show that, on average, every agent with the same workflow spends the same amount of time employing each strategy element.

Since there is no central controller deciding which agent should rely on what strategy element, for a given system pressure, it is interesting to explore by what means the balance between strategy elements is brought about. At any point in time, a system of hybrid consumers can be represented as two interdependent populations of agents, each relying either on random selection or developed preferences. Agents relying on $\mathcal{R}$ are more aggressive, selecting resources randomly and thereby dispersing their activity across all system resources. Agents relying on $\mathcal{P}$, on the other hand, canalise their activity in a specific region of the system resources. Where supply meets or exceeds demand, the former encourages system fairness, while the latter lowers system cost.

Moreover, both populations exert a specific pressure on each other. By "stealing" the preferred resources of conservative $\mathcal{P}$ agents, aggressive $\mathcal{R}$ agents drive $\mathcal{P}$ agents to switch strategy. At the same time, $\mathcal{R}$ agents that successfully allocate resources also switch strategy. In both cases, such switching prevents agents relying upon the same resource for a long time. This ensures that the costs of resource competition are distributed fairly among all agents. Furthermore, as system dynamism increases (with increasing load or heterogeneity, for instance), and the chance of developing useful preferences falls, the proportion of $\mathcal{R}$ agents increases. Likewise, if system dynamism relaxes, the proportion of agents successfully exploiting preferences increases. This coupling between strategy elements drives the system behaviour, and its response to externalities such as load or heterogeneity.

The nature of this coupling resembles certain accounts of self-organisation within natural decentralised systems that we presented in Section 3.5. There, the capability of a thermodynamic system to self-organise and thus perform useful work was proportional to its distance from thermodynamic equilibrium. Only in this state did the constraints limiting the behavioural degrees of freedom of individual system elements form, enabling useful work to be extracted from the system.

In what follows, we conduct a very simple exercise of re-interpreting the aforementioned concepts of *equilibrium*, *constraint* and *work* within the context of our autonomic system model. Having given such an interpretation, we apply it to analyse the behaviour of the model. In doing so, we lay down a basic thermodynamic framework that will be extended in the remainder of this thesis.

4.5.1 Equilibrium, Constraint and Work

4.5.1.1 Equilibrium

In thermodynamics an equilibrium is associated with a state of uniformity, where all differences (eg. temperature, pressure) between interacting systems elements are minimal or non-existent.

As we suggested in Section 2.5.8 one way of interpreting the concept thermodynamic equilibrium within a computational system model can be provided by analysing *behavioural repertoire* of each agent and the degree of freedom each agent holds when choosing actions from such a repertoire. If there exists no bias and the selection of any action is equally probable, the system can be thought of a to be in equilibrium state. The emergence of certain bias for particular subset of actions that agents tend to choose, on the other hand, can be understood as the departure of the system from the state of uniformity represented by the equilibrium.

In the remainder of this section we will elaborate on this analogy in a more detailed manner by introducing thermodynamic concepts of constraint and work to the computational system model and using them to analyse system dynamics and relation to system efficiency.

4.5.1.2 Constraint

Given that we are now able to identify whether a multi-agent system is at equilibrium, let us consider what change indicates that the system has been shifted from such a state. Again, we will interpret this in accordance with thermodynamics.

For the purpose of explanation, assume the existence of an imaginary *force* F that, when applied to the agent, imposes a constraint on the actions it may select from its behavioural repertoire, where the stronger the force, the greater such a constraint becomes.

In physical systems the discussed constraining force can be as crude as mechanical work causing gas compression within the thermodynamic engine, or as sublime as the self-organised columns of interacting molecules of viscose fluid observed in the Benard cells experiment (described in section 2.5.4). The literature review summarised in Section 3.4 suggests that within a computational system the constraining role of the aforementioned 'mechanical' or 'thermal' forces can be facilitated through *information* that is being exchanged between agents and that may either limit or relax the behavioural repertoire of the agent.

4.5.1.3 Work

Both energy and information exchange are the result of local interactions between system components (molecules in a physical system; agents in a computational system) and, under the right conditions, both are 'motive forces' for achieving spontaneous system organisation. However, as we observed when discussing thermodynamics of self-organisation (Section 3.5), it is not the mere injection of energy that allows a thermodynamic system to perform work, but the *organisation* of this energy, which displaces the system from equilibrium.

The same can be said of the distribution of information within a computational system. Here, the capacity of the system to perform useful work is determined by the ability of agents to establish interactions and information exchanges between their peers that impose constraints on their behavioural repertoire. In this state, agents begin to favour interactions that increase their personal utility (eg. their efficiency at allocating tasks) and limit possibly uncoordinated ones that could destabilise the performance of other agents. Consequently, the eventual capacity of the system to perform useful work arises from organised flow of information across the population of interacting agents.

4.5.2 Decentralised Control Interpretation

Having provided a re-description of thermodynamic concepts of equilibrium, constraint and work within a computational system, let us now apply these terms to interpret the process of decentralised control within the autonomic system model presented in this chapter. We base our interpretation on the evaluation of two following hypotheses:

Hypothesis 1:

The increase in computational system efficiency can be understood in terms of thermodynamic displacement from equilibrium and, according to this, is proportional to the increase of constraint imposed on the behaviour of the system elements.

Hypothesis 2:

The decrease in computational system efficiency can be understood in terms of a thermodynamic return to equilibrium and, according to this, is proportional to the decrease of constraint imposed on the behaviour of the system elements.

It is important to note that both hypotheses can be achieved independent of each other. For example, we do not exclude the possibility of achieving high system efficiency in situations where system is in equilibrium or far from equilibrium. Therefore, validation of one of the hypotheses does not necessarily entail the outcome of the other hypothesis.

4.5.2.1 Measures

To identify the system's displacement from equilibrium and the associated emergence of constraint on the behaviour of its elements, a measure of *system constraint* is provided. The efficiency of the system, and thus its capacity to perform useful work is provided through a *system efficiency* measure. Both measures are explained below.

System Constraint (κ)

For each consumer agent we define its *behavioural repertoire* by the set of unique service providers that it may choose to query for its task allocation. The degree of constraint ($\kappa \in [0, 1]$) is defined through the following formula:

$$\kappa = 1 - \frac{N_c}{N}$$

where N represents the total number of providers and thus constitutes the full agent's *behavioural repertoire*, whereas N_c corresponds to the set of the providers from which choice is made (choice set) the value of which is calculated for the particular strategy as follows.

- $\mathcal{R}$ strategy configuration: Since agents relying on this strategy do not have any constraint imposed on the selection of any action from their *behavioural repertoire*, we assume that $N_c = N$.
- $\mathcal{P}$ strategy configuration: For agents that employ only $\mathcal{P}$ strategy and thus either establish preference to a successful provider or continue provider selection in the same, ordered, manner from the top of registry list have a choice set defined by the number of provider queries (q) they have made until they discovered the available provider or failed the allocation. As a consequence, for this configuration $N_c = q$.
- $\mathcal{RP}$ strategy configuration: Agents using this strategy are assumed to have a choice set defined based on the strategy component that the agent was employing during the service allocation. If the agent successfully relied on $\mathcal{P}$ strategy component and thus successfully allocated service employing the previously established preference, its $N_c = 1$ since it achieved its allocation in one query ($q = 1$). However, if the agent failed to rely on established preference and thus conducted unconstrained selection relying on $\mathcal{R}$ strategy component, its choice set is defined as $N_c = N$, implying that it performed a random choice from the whole *behavioural repertoire* of possible actions.

Given the measure of constraint experienced by each consumer separately, the system constraint is defined as an average of individual agent constraints over a sampling period of time (which we define to be a 10 second time interval).

System efficiency (e)

We define system efficiency ($e \in [0, 1]$) by scaling the obtained mean system functioning cost ($cost_{current}$) by the most optimal (minimal) cost achievable for the current model run ($cost_{min}$) . The efficiency (e) is thus defined by the following formula:

$$e = \frac{cost_{min}}{cost_{current}}$$

4.5.2.2 Experiment 1. Influence of System Strategies

In this section we provide results that illustrate how system efficiency (e) and constraint (κ) change when agent strategies are varied. The scenarios explored here are identical to those employed in the scalability tests (see section 4.4.1) in which three different model setups are provided. Each such setup differs only by the configuration of local strategies ($\mathcal{R}$, $\mathcal{RP}$ and $\mathcal{P}$) agents are set to employ. Results showing how the degree of constraint varies for these three setups are illustrated on the left side of Figure 4.6. The relationship between system performance and the degree of system's constraint for each of the three model setups is plotted on the right side of the figure.

As the figures show, the best system efficiency is achieved for the system in which agents rely on the $\mathcal{RP}$ strategy. In this state, the system also achieves the highest degree of constraint, as agents establish preferences to provider agents that are capable to satisfy their requests. In this configuration the system is capable to attain the highest level of constraint, as consumers establish and exploit preferred providers during the system operation. However, as the right figure shows, maintaining such high level of constraint is only possible in optimal functioning conditions. Here both, the efficiency and the level of constraint are presented for $\mathcal{RP}$ model configuration in which service providers are continually (in 4 second time intervals) removed from the system, starting from the $40th$ simulation second. As the demand-supply proportion gradually changes and there are less resources to satisfy consumer requests, we can observe that the initially high level of constraint begins to dissappear in favour of more exploratory selection of resources that is driven by $\mathcal{R}$ strategy component. The gradual decrease in the system constraint level becomes in this situation accompanied by the decreasing system efficiency as it becomes more difficult and thus more costly to discover available providers.

The worst performance can be observed for a system relying on the $\mathcal{P}$ strategy. This is caused by the high competition for a small subset of available system resources, during which consumers attempt to query providers that are already busy. As a result of this, even though they eventually establish preferences to successful providers, in the next allocation round these providers are already employed by other agents. Such competitive behaviour is avoided when agents rely only on the $\mathcal{R}$ strategy.

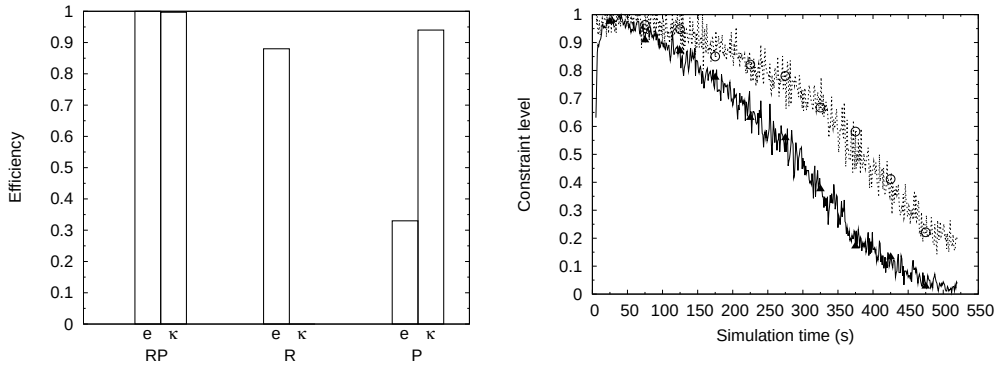

FIGURE 4.6: Left figure illustrates correlation between the level of system constraint ($\boldsymbol{\kappa}$) and its efficiency ($\boldsymbol{e}$) for three model configurations: $\boldsymbol{\mathcal{R}}$ (rectangles), $\boldsymbol{\mathcal{P}}$ (circles) and $\boldsymbol{\mathcal{RP}}$ (triangles). Right figure shows the level of constraint (rectangles) and the system efficiency (circles) for $\boldsymbol{\mathcal{RP}}$ model configuration during system reliability experiments. $\boldsymbol{U_N = 180}$, $\boldsymbol{S_N = 360}$, $\boldsymbol{T_N = 400}$ seconds, $\boldsymbol{H = 1}$.

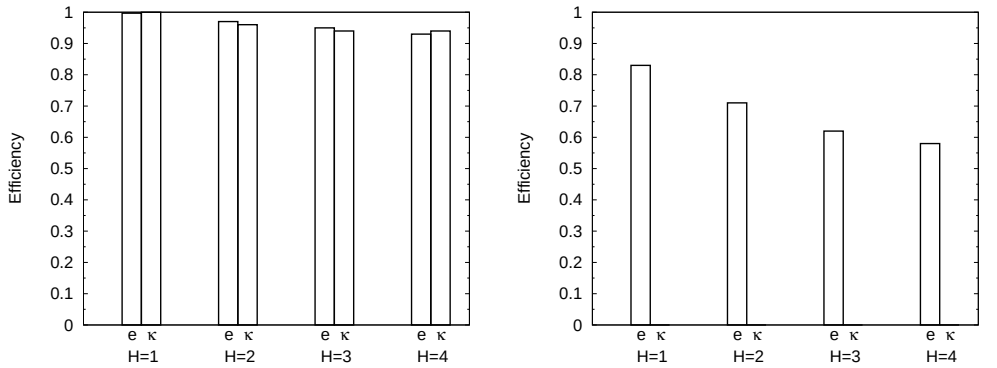

FIGURE 4.7: Relation between system efficiency ($\boldsymbol{e}$) and constraint ($\boldsymbol{\kappa}$) for four different heterogeneity system configurations ($\boldsymbol{H = 1, 2, 3, 4}$) for the system employing $\boldsymbol{\mathcal{RP}}$ strategists (left) and $\boldsymbol{\mathcal{R}}$ strategists (right). In all experiments $\boldsymbol{U_N = 180}$, $\boldsymbol{S_N = 360}$, $\boldsymbol{T_N = 400}$ seconds.

It is interesting to observe that, despite the lack of any learning mechanism allowing agents to return to efficient providers (since agents are not able to establish any preferences), the system achieves a high level of efficiency. Furthermore, this is accompanied by the lack of any constraint imposed on the *behavioural repertoire* of $\mathcal{R}$ strategists. In contrast to this, $\mathcal{P}$ strategists that achieve close to the highest level of constraint achieve the worst performance observed among the three model configurations.

The explanation for these observations is provided in the next sections where we investigate how measures of the efficiency and constraint change when the difficulty of resource allocation is varied through heterogeneity and, finally, discuss the results.

4.5.2.3 Experiment 2. Influence of System Heterogeneity

In Figure 4.7 (left) a relation between system efficiency and constraint for four different system heterogeneity settings ($H = 1, 2, 3, 4$) is illustrated for a system model in which agents employ the $\mathcal{RP}$ strategy. Figure 4.7 (right) illustrates mean efficiencies of models

employing the $\mathcal{RP}$ and $\mathcal{R}$ strategies for the corresponding heterogeneity levels. In both cases, the experimental model setups are identical to those explored in the heterogeneity tests in Section 4.4.3.

Results reporting the system efficiency and constraint for system configuration in which agents employ only the $\mathcal{P}$ strategy are not shown as the performance achieved by such a configuration, due to high competition for resources, is poor in every setting.

The system where consumers employ the $\mathcal{RP}$ strategy (left side of Figure 4.7) outperforms a model setup where agents rely only on the $\mathcal{R}$ strategy (right side of Figure 4.7). A more detailed analysis of the difference in both model performances is reported in Section 4.4.3, including mean workflow completion costs for model setup where heterogeneity is equal 4 ($H = 4$).

Since the only difference between both model setups is the ability of agents relying on $\mathcal{RP}$ strategy to exploit a preference for the last successfully executed provider, it is the ability of system elements to establish a constraint on their selection that achieves higher than the $\mathcal{R}$ strategists performance. As results on the left figure show, the formation of constraint through this simple technique becomes increasingly important as demand for resources becomes more heterogeneous and thus the difficulty of resource allocation demanded from the system.

4.5.2.4 Hypotheses Evaluation

The above experiments confirm that the high system efficiency is predetermined by the system's ability to impose constraint on the behaviour of consumer agents. This, as the heterogeneity results presented in Section 4.5.2.3 illustrate, is especially important in conditions when the difficulty of resource allocation increases and the consumer agent requests can be satisfied only by a subset of available providers.

Moreover, the analysis of results reported in Section 4.5.2.2 suggests that the satisfactory degree of constraint cannot be pre-imposed by the fixed system configuration that either limits agent's behavioural *repertoire* (when consumers are allowed to use the $\mathcal{P}$ strategy only) or removes the possibility to establish a constraint (when agents are allowed to rely on the $\mathcal{R}$ strategy only). Consequently, what is required is the system design that has the capacity to either impose or relax constraints on the behaviour of its elements at run-time and in response to existing resource demand conditions. To this end, for the simple resource allocation scenarios explored by the model, agents with the $\mathcal{RP}$ strategy configuration exhibit such adaptive behaviour.

Important to note is that the degree of constraint imposed on agents is a variable property that is adjusted by the system at run-time through local strategy reconfigurations conducted by individual system agents. In the case when such conditions are mild (eg.

when heterogeneity level is minimal) there may be no need to impose the constraint on the selection of providers. However, as the difficulty of resource allocation grows such that only a subset of providers may satisfy particular consumer agents, the existence of constraint is a necessity for stable and reliable system operation.

4.6 Discussion

In this chapter we have presented a simple model of a decentralised autonomic system tasked to effectively distribute available system resources to users requesting them. To facilitate this, a set of very simple local strategies was provided and their efficiency evaluated for a variety of different resource allocation scenarios and strategy configurations.

To understand why only certain strategy combinations are able to preserve balanced and fair provision of resources, a thermodynamic interpretation of the system response was provided. In this interpretation we introduced a very simple thermodynamic framework, based on which the self-organising properties of the system were discussed.

In what follows, we will focus on more challenging resource management problems involving dynamic load-balancing, adaptive service provisioning and power management. In particular, we will propose fully decentralised autonomic system models that function without a central registry and within which providers are allowed to reconfigure at run-time in order to adjust the service type they offer (or their on-line status) to perceived demand. As such system flexibility will demand more adaptation at the level of autonomous system elements, we will propose novel algorithms achieving the required system functionality.

Finally, having more advanced models of autonomic systems, we will extend our understanding of their design and functioning by analysing a thermodynamic account of their self-organisation.

Chapter 5

The Model

5.1 Introduction

In this section, we propose a framework for a *bottom-up* role and resource allocation mechanism, whereby the adaptation of agents (in response to changes in the environment) is based on stimulus-response based reinforcement mechanisms inspired by behaviours that encourage self-organisation within insect societies [7, 120].

In contrast to the previously introduced simplistic model (in Chapther 4), here we consider a fully decentralised model with no access to central service registries. In the absence of such centralised controllers, the system elements need to preserve a certain degree of autonomy, allowing for local adaptation to occur given perceived changes in the environment. This architectural flexibility is provided through the use of a decentralised multi-agent system architecture.

To this end, in Section 5.2 we outline the general properties of the model, whereas Section 5.3 provides a detailed explanation of its architecture and local-decision making mechanisms employed by autonomous agents. Section 5.4 discusses the experimental setup of such a model that we will apply during its experimental evaluation in the remainder of this thesis. Finally, Section 5.5 concludes the chapter with an explanation of system behaviour and the performance analysis tools that we will use for measuring system efficiency and analysing behaviour in a range of dynamic resource allocation conditions.

5.2 Model Features

In the previous chapter we have considered a very simplistic version of a distributed system constituting a population of resource providing and resource consuming agents

together with a single service registry. Whilst sufficient for our analysis, such model lacked some of the key features that distinguish modern autonomic systems and their dynamic nature of computation. In what follows, we outline these features and, in the remainder of this chapter, explain how they were incorporated into our more realistic autonomic system model.

1. **Distributed architecture**

   In order to model distributed systems, we rely on multi-agent system model. We assume that each system element is represented as an autonomous software agent that either consumes resources (consumer agent) or provides them (provider agent).

2. **Physical constraints**

   Agents provide and consume resources offered by physical machines. As a consequence, both the quantity of offered resources as well as the computational resources used by the agents during their decision-making process are constrained. For this reason, we assume that following constraints should be considered by the model:

   (a) *Limited resource capacity*

   Provision of a service by a provider agent to any consumer agent requesting it can not be guaranteed. The provider may honor the request only if it currently has enough resources and is configured to offer the requested service.

   (b) *Interaction cost*

   Interactions between system components involve certain costs in terms of time and energy. In this thesis we focus on the former type of cost where each interaction between agents consumes time and thus inefficient behaviour may affect the overall performance of the system in terms of jobs completed within a period of time.

   (c) *Provider reconfiguration*

   Provider agents are capable of reconfiguring at run-time to offer a different service type. As this activity in real systems often requires rebooting or erasing of critical data stored on the machine, we impose a timeout during which the provider agent is unable to service incoming requests.

3. **Decentralisation**

   In large scale IT systems it is difficult, impossible or impractical to preserve full access to global information about the system state for individual agents to employ during their decision-making process. We reflect this decentralisation through:

   (a) *Lack of global information about individual and global provider efficiency*

Consumer agents are not provided with central controller or global information repository allowing them to select the currently most efficient providers. Consequently, each agent operates only on a local information discovered through individual experience or information exchange with local peers.

(b) *Lack of global information about individual and global consumer demand*
Similarly, there exists no central authority offering information about the current demand imposed by consumer agents. For this reason, provider agents also rely only on information discovered through interactions with consumer agents to guide their provisioning decisions.

4. **Provider heterogeneity**

   Various services are offered by a number of provider agents with varying levels of resource capacity. As a consequence, the same service type may be offered by several providers, each provider possibly being characterised by a different level of reliability and a different response time.

5. **Consumer heterogeneity**

   IT infrastructures are environments within which requests originating from a diverse group of different users need to be satisfied. To reflect heterogeneity in user requests, we assume that each request may differ in the type of service demanded, the capacity (representing a scalar quantity or amount of resources required) and sought time limit before it has to be allocated.

6. **Dynamism**

   The model represents the operation of an open and dynamic system.

   (a) **Supply — demand fluctuations**
   The ratio between supply and demand does not remain static but varies according to some mechanism intended to approximate real conditions.

   (b) **Changing demand**
   Each request is unique with respect to demanded service type, service capacity and allocation time limit. Furthermore, the overall system demand may change at run-time requiring different service types to be offered by provider agent population.

   (c) **Openness**
   System is not closed: new agents arrive and existing ones are removed.

5.3 Decentralised Autonomic System Model

The challenge of role and resource allocation can be viewed as a market-based, service allocation problem, where there is a (continually changing) demand for services of a given

type, and thus the market responds[1] by changing its supply of such services (or their on-line/off-line status) to satisfy the demand. As stated earlier, a multi-agent system is analogous to an autonomic system, which can be thought of as a collection of computing resources tied together to perform a specific set of functions [58]. These resources may be hosted in a distributed fashion by a number of servers deployed over networked machines, which provide services to each other. The framework is therefore modeled as a multi-agent system comprising a number of provider agents (*providers*), that offer services of a specific type, and consumer agents (*consumers*) which request and utilise the available services to achieve some task. We assume that both service providers and service consumers are agents running on *constrained* hardware components. Depending on the characteristics of the system, interaction between these agents may be limited by power consumption (e.g., sensors), bandwidth consumption, or time-delayed response, all of which may have associated costs if service execution is to take place "on-demand", quickly or by some deadline. In the system presented below, one aspect of such hardware limitation is represented in the form of service capacity, such that each agent may only satisfy service requests for a restricted set of service types, provided to a limited number of consumers simultaneously.

To facilitate the supply of a larger variety of services to consumers, we assume an agent is capable of reconfiguring the service type it provides at run-time. This involves a significant cost in the form of down-time during which various administration tasks may be performed, such as: completing existing service commitments; removing security compromising data from the machine state, or resetting the execution stack; or loading the new software modules representing the new service types. We also consider that providers increase their utility by successfully satisfying service requests, and that consumers increase their utility by successfully consuming services. Thus, to maximise utility, provider agents try to avoid offering services for which there is little demand (thus minimising idle time), and consumer agents try to reduce the time required to provision services (during which their requests are not satisfied) by locating providers that are available and can rapidly satisfy their requests. To locate possible service providers, a decentralised service discovery model is assumed, whereby each agent maintains a limited registry of details regarding service providers in its environment. Consumers can discover new providers through regular dialogue with other peers, which continually evolve and share their awareness of local service availability (see Figure 5.1).

The evolution of the system is therefore driven by a continually reconfiguring network of peers. Both consumer and provider agents co-adapt to each other by exchanging information, and reconfiguring their interactions; i.e., by changing which services are offered (in response to observed changes in service demand), or by changing which providers should be contacted (based on observations of the availability of different

[1] In this context, we refer to the market as a decentralised collection of service providers, that each respond individually based on their perception of changing service demand, rather than a single, atomic, coordinating entity.

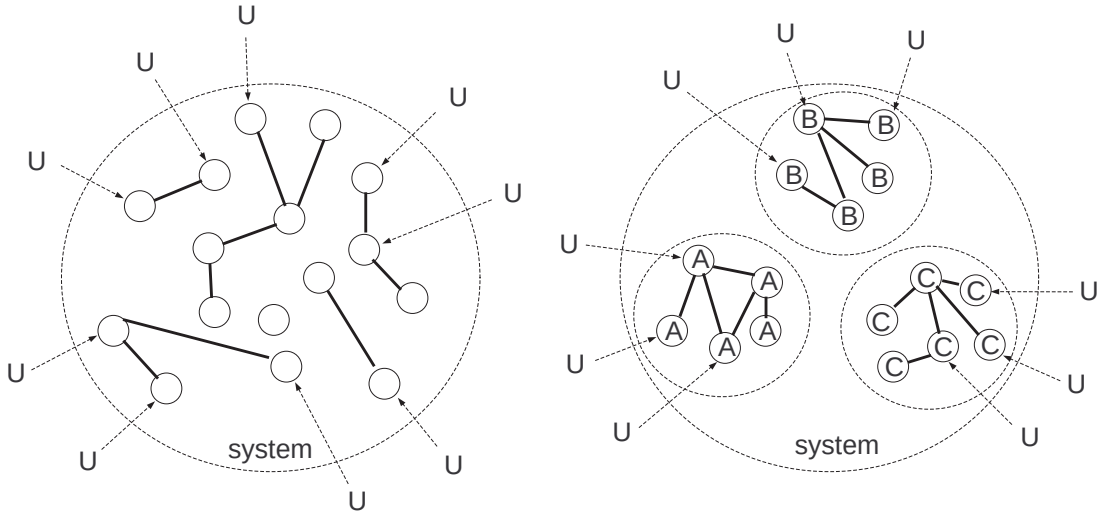

FIGURE 5.1: An overview of the resource management organisation process. Two system states are represented: *disorganised* (on the left) and *organised* (on the right), where users (U) impose a demand for different types of resources (service requirement) on resource providers. The initially inefficient configuration (left), represents the case in which providers have no knowledge of what services are in demand, and consumers don't know which providers offer their desired services. The final, stable organisation (right), in which service demand is satisfied by local supply, emerges from limited information exchange between consumers and providers regarding service availability.

services). These local responses are driven by the decision-making mechanisms (detailed in Sections 5.3.1, 8.2.2 and 5.3.3, and summarised below) and the information that is propagated throughout the topology of agents as a result of their activities.

As providers have no global information regarding service demand, they utilise their own experience (based on the frequency and type of queries they receive from consumers), as well as information of service availability garnered from those consumers they interact with, to determine whether to continue offering a given service type or to *switch* to providing another service type. Consumers discover new providers through a process of social learning, where new information is acquired through "gossiping". When a consumer and provider interact, the consumer may provide details of other providers that it has interacted with together with their efficiency estimates. This information is propagated from the provider agent to other consumers that are co-located with this provider and thus employ it at that time. During this process, consumers learn to which peers to communicate their local information such that their local resources market environment remains stable and offers sufficient amount of resources to satisfy their demand. Consequently, service management and provisioning strategies should emerge from local co-adaptations of individual agents based on observations of previous transactions. Whilst this naturally involves sharing some knowledge, the agents independently modify their individual models of the local environment. Since service availability can fluctuate as a result of several factors, including current demand contention for services, and demand for other service types (resulting in a reconfiguration of service offerings), it is

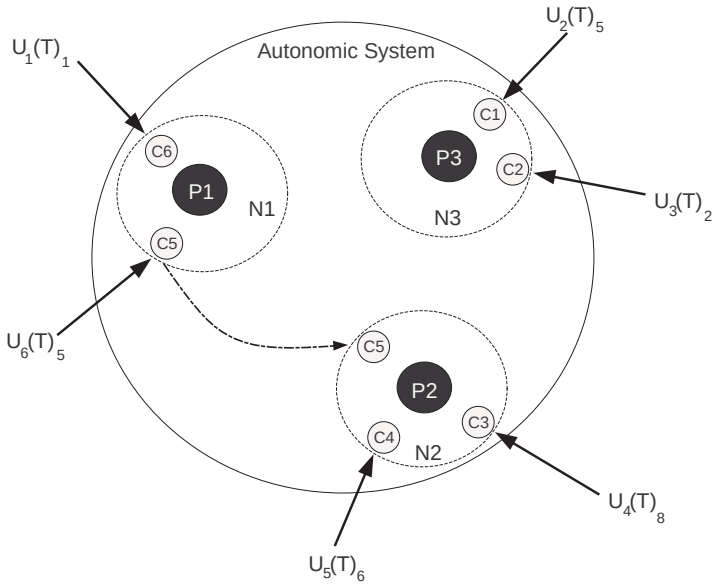

FIGURE 5.2: An example of autonomic system functioning. A number of tasks (T) are issued by infrastructure users (U) to autonomic system computational nodes (N). When a task is intercepted by this node, a new consumer agent (C) is spawned within the system and co-located with the provider managing such node.

necessary that agents maintain an accurate model of the environment by maintaining a continuous flow of pertinent information with their peers.

Whilst this notion of sharing information may initially appear counter intuitive (raising the question as to why a consumer would supply information on available and efficient providers to other resource seeking consumers, thus possibly reducing supply on its own preferred services), it provides a mutually beneficial mechanism whereby consumers can acquire a timely and accurate model of services in the local community, and providers can determine a realistic estimate of the service demand in the same community, and thus (if necessary) *switch* to improve their own utility. The neighbourhood that emerges is dependent on the size of the model that consumers retain of their peers. In addition, the stability of the neighbourhood is also dependent on this model size; the larger the model, the greater the chance of instability, as more providers may *switch* the type of services they offer in response to the perceived change in demand.

The framework makes the assumption that agents will exchange information freely and truthfully and that, moreover, the amount of information passed between agents will be influenced by the degree of *consumer stress* that the consumers experience. This consumer stress (described more formally in Section 5.3.1) reflects the difficulty in locating available providers for a given service, and hence provides an indication as to whether the service supply can sufficiently meet current service demand. Whilst there is, perhaps, the opportunity for deceit in such a system, since the supply of (and demand for) services can fluctuate, maintaining an accurate model of the environment in each agent involves maintaining a continuous flow of information between agents.

A sketch of an autonomic system model is illustrated in Figure 5.2. During the system lifetime, a continuous stream of tasks (T) is issued to a system. These tasks represent service allocation requests and originate from infrastructure users (U) that interactively engage with the system for a limited time period (session). During this time users interactively issue a number of tasks and await a system response. Each task arriving to the system is handled by a consumer agent that is spawned by the system for each new user. During consumer agent initialisation, the agent is provided with the list of provider agents existing within the system and becomes co-located with one of them on a particular node managed by that provider. If there are other consumer agents co-located with the same provider, the newly created consumer becomes automatically co-located with these agents too. For example, as illustrated in Figure 5.2, the arrival of task (T_1) from a user U_1, causes the creation of a new consumer agent $C5$ and its co-location with provider $P1$ managing node $N1$. At the end of this step, consumer $C5$ is also co-located with consumer $C6$ sharing the same provider. After co-location, the consumer becomes updated by the local provider about the existing resource state of the local system environment. Based on this information, the consumer builds its internal model of the local resources environment, and relies on it when conducting allocation. During the allocation process, the consumer agent may decide to employ a different provider for its task allocation than the one it was co-located with initially. If the consumer successfully employs the other provider, it changes its co-location and 'migrates' to the other node[2]. For example, in Figure 5.2, consumer $C5$, that was originally co-located with provider $P1$ eventually migrates to node $N2$. After this, $C5$ is no longer co-located with $P1$ and $C6$, but instead with $P2$, $C3$ and $C4$.

As the number and requirements of consumers cannot be known *a priori*, and as this demand and preference will vary over time, the design goals are threefold:

1. determine which providers should be configured to offer what service types, in order to satisfy current demand;

2. determine which of the providers known to a consumer should be utilised such that the system minimises competition; and

3. determine how should the system be organised such that it is robust to changes in supply and demand for particular service types.

The remainder of this section provides details regarding the models assumed by both the service consumers (Section 5.3.1), and service providers (Section 8.2.2). The mechanisms used to facilitate knowledge exchange (i.e., "gossiping") are presented in Section 5.3.3.

[2]It is important to note that we use terms such as 'co-location' and 'migration' only for descriptive purposes. In reality, all agents may exist on the same machine or be distributed on a set of machines and never change their location.

5.3.1 Model for Service Consumers

Consumers are agents that request and consume services provided by one of the provider agents. An agent may be capable of both offering services to its peers, as well as consuming services offered by its peers; however, for the purposes of this thesis, we consider the model for each behaviour as separate.

The process of service allocation is initialised when a consumer agent receives a service request that is in the form of a task (T) represented as a tuple:

$$T = [S_t, S_c, S_l],$$

where S_t represents the service type demanded by the task, S_c identifies the demanded capacity the service has to offer, and S_l is the maximum time the allocation may take before the task is considered as failed.

The consumer monitors the behaviour of known service providers locally, and uses this knowledge both to provision future service requests, and to share this knowledge when establishing community knowledge. For this purpose, the consumer maintains a local registry, $\mathcal{R}_c$ containing tuples corresponding to services that the agent is aware of. Each tuple is defined as follows:

$$\mathcal{R}_c = \langle \alpha_p, \epsilon, \lambda \rangle$$

where α_p corresponds to an agent that has provided the service of the type required by the consumer at some point in the past, *evaluation* denoted by $\epsilon \in [0.01..\infty]$ corresponds to a score or preference for using provider α_p and *affinity* denoted by $\lambda \in [0.01..\infty]$ reflects the score of preference for communicating local information to agent α_p.

As consumers will not possess complete knowledge about whether a provider is currently available, or even if it is still configured to provide the service of type *type*, they rely on a local learning mechanism which enables them to estimate the possible demand of a given service type. This estimate is based on the periodically exchanged information obtained from the different providers they interact with (Section 5.3.3), the result of which is stored within the relevant epsilon (ϵ) parameter within their local registry $\mathcal{R}_c$.

This registry is also used when provisioning services of a given type. During this process, the consumer issues requests to the providers selected from this registry, until a provider is found which can satisfy the request. The order in which providers are selected is proportional to their ϵ score and is achieved through a probabilistic roulette wheel selection mechanism. To guarantee that any provider will be queried at most once during the

same allocation cycle, the providers that were already selected by the roulette wheel mechanism (with no success) are not considered in subsequent selections.

Each request takes some finite time (T_q), and the provider will respond either to confirm that it will satisfy the request (i.e., that it is available to provide the desired service *type*) within a specified S_l allocation time limit, or to reject the query; either because it currently does not provide that service type, it is unavailable (i.e., it is currently satisfying another query, and does not have sufficient resource to simultaneously honour an additional request without compromising current commitments) or because it is not able to provide the service within S_l time limit (i.e., it is overutilised and this affects its service provisioning time).

Each provider may offer a limited number of service instances of a given type simultaneously (depending on the resource capacity it currently has); therefore, provided that it has enough resources to satisfy another allocation query, a new request can be honoured. If a provider is capable of honouring the request, the service is executed. The execution takes some finite time T_e during which the amount of resources demanded by the allocated task are consumed from provider. Once the allocation finishes, the resources occupied by the task are released.

Typically, at the beginning and end of every allocation cycle, the agent exchanges local knowledge with known providers (described in Section 5.3.3). The knowledge exchange is assumed to take some finite time, T_i (irrespective of the number of providers involved), and corresponds to the process of sharing information about local service demand (and availability), and thus evolving a localised community structure. After the allocation of the current task, the agent inspects its local task queue and if there is another task to be allocated, it initiates another allocation cycle. If the task allocation fails, it is assumed that the consumer will reattempt to allocate the same task before it finally fails and removes it from its local queue.

The time intervals between which new tasks will be dispatched to a consumer agent is determined probabilistically using a Poisson distribution, with the mean ω. If the consumer is already allocating a task, the newly arriving one will be added to its local queue.

The consumer periodically updates the ordered set $\mathcal{R}_c$ to reflect its experience in allocating tasks, and to minimise the number of future rejected queries. If a request was successfully satisfied, then the tuple r_p corresponding to the provider α_p which provided the service *type* is modified, such that ϵ is incremented in the following manner:

$$\epsilon_{\alpha_p} \rightarrow \epsilon_{\alpha_p} + \frac{S_l}{S_l^*}$$

where S_l^* is the time it took an agent to finalise service execution with help of provider

Algorithm 1 Consumer task allocation algorithm

Require: a consumer α_c with a need for service *type* of capacity *capacity*, and the set $\mathcal{P}$, which contains all the known providers that appear in $\mathcal{R}_c$

Ensure: a service of type *type* is provisioned, and $\mathcal{R}_c$ is updated through the exchange of information

1: $\mathcal{R}'_c :=$ exchangeRegistryWithProvider $(\mathcal{R}_c, \alpha_p)$
2: $\mathcal{R}_c :=$ mergeRegistry $(\mathcal{R}_c, \mathcal{R}'_c)$
3: **for** $(q := 0$ to $f_{max})$ **do**
4: $f_q := 0$
5: $r_p := \mathcal{R}_c[q \ modulo \ |\mathcal{R}_c|]$
6: **if** typeOfTuple $(r_p) = type$ **then**
7: $\alpha_p :=$ providerOfTuple (r_p)
8: $response :=$ sendRequest $(\alpha_p, \langle \alpha_c, type, capacity, c_s \rangle)$
9: **if** $response = accept$ **then**
10: **break**
11: **else if** $response = reject$ **then**
12: increment (f_q)
13: **end if**
14: **end if**
15: **end for**
16: **if** exec $(r_p) = success$ **then**
17: $\mathcal{R}_c := (\mathcal{R}_c / r_p) \cup \langle \alpha_p, type, \epsilon - \delta_\epsilon \rangle$ $\{Increment \ \epsilon \ of \ successfully \ executed \ provider \ r_p\}$
18: **end if**
19: **for all** $r_p \in \mathcal{R}_c$ **do**
20: $\mathcal{R}_c := (\mathcal{R}_c / r_p) \cup \langle \alpha_p, type, \epsilon \times \delta_{decay} \rangle$ $\{Update \ \epsilon \ of \ r_p\}$
21: **end for**
22:
23: **for all** $\alpha_p \in \mathcal{P}_c$ **do**
24: $\mathcal{R}'_c :=$ exchangeRegistryWithProvider $(\mathcal{R}_c, \alpha_p)$
25: $\mathcal{R}_c :=$ mergeRegistry $(\mathcal{R}_c, \mathcal{R}'_c)$
26: **end for**
27: awaitNewTask ()

α_p, and S_l is the overall task allocation time limit defined by the task.

To ensure that this model of provider availability does not become stale, a decay function is used to adjust the ϵ and λ parameters for all tuples in $\mathcal{R}_c$, by applying a *decay coefficient* δ_{decay}[3].

A consumer agent maintains a stress parameter ($\Omega_c \in [0, 1]$) that represents an agent's local estimate of how effectively it acts within the system given the information it has about available resources. The stress value is updated by the consumer at the end of

[3]The decay coefficient used in this model has the value $\delta_{decay} = 0.9$; this value was determined empirically.

each allocation cycle through the following formula:

$$\Omega_c \rightarrow \frac{q}{N}$$

where q is the number of provider queries conducted by the consumer agent during the current allocation round and N is the number of provider estimates (ϵ) (kept by the consumer agent within its local $\mathcal{R}_c$ registry) that are greater than $0.1\epsilon_{max}$ (ϵ_{max} being the maximum estimate value kept within the consumer's registry).

The value of the stress is proportional to the number of queries (q) and inversely proportional to the number of attractive provider estimates (N) kept within its local registry. If the agent conducts more queries than N ($q > N$) and thus has run out of informed decisions, the stress reaches the maximum value equal to unity ($\Omega_c = 1$). In order to maintain gradual stress change over the agent operation, each consumer agent remembers four recent stress estimates (calculated in four recent allocations) and calculates average of these as its current stress level. Stress calculated in this manner influences the manner in which an agent acquires information communicated by other peers.

The algorithm used by a consumer agent is represented in Algorithm 1. In line 1-2, the registry is updated through the information exchange with the provider the consumer agent is currently co-located with. The consumer then, using the roulette wheel selection mechanism, traverses its personal registry list, searching for providers that can satisfy its service request, until one is found (lines 3-15). Once a provider is found and its service provision is successful, the consumer updates its ϵ rating for this provider (lines 16-18). Prior to exchanging information with the selected subset of providers (lines 23-26), the ratings of all the providers kept within consumer's registry are decremented using the decay coefficient (lines 19-21). Finally, the agent awaits a new task arrival into its queue (line 27) or selects the one already awaiting allocation.

5.3.2 Model for Service Providers

Providers model the local demand for services to determine which services they should offer. However, within resource-bounded environments, providers may only offer a limited number of services at any time, despite posessing the *capability* of offering several types of services; due to limitations in physical resources (e.g. memory size, processor capacity, etc), or based on security issues. Business sectors (such as the E-Business sector) also limit the number of software modules that servers can provide at any time to avoid information leak. The suspension of availability of one service type and introduction of another can have an implicit cost, as this reconfiguration typically takes some time during which the agent cannot perform any further service execution, and thus will not obtain any utility increase. We therefore assume that each provider agent α_p can only offer one service type at any time, but has the capability of offering several other

service types (subject to reconfiguration). The set:

$$Capability = \bigcup_{\forall \alpha_p \in MAS} Capability_{\alpha_p}$$

contains the union of all service types available from all service providers in the multi-agent system (MAS), whereas $Capability_{\alpha_p}$ corresponds to the set of services that α_p is capable of offering.

Thus, to determine which service type α_p should offer, it maintains a model of current, local service demand, and determines which services to offer from that model. To achieve this, the provider maintains a registry $\mathcal{R}_p$ containing tuples corresponding to service types ($S_t \in Capability_{\alpha_p}$) the agent is able to offer. Each tuple is defined as follows:

$$r_t = \langle S_t, s, \theta \rangle$$

where S_t corresponds to a unique service type, $s \in [0.01..\infty]$ corresponds to a stimulus or preference for selecting S_t to offer and $\theta \in [\theta_{min}, \theta_{max}]$ represents a response threshold associated with this service type the role of which is explained in the remainder of this section.

Providers receive requests from consumers in the following form:

$$req_i = \langle \alpha_c, S_t, S_c, S_l \rangle$$

whereby α_c corresponds to the consumer which submitted the request, $S_t \in Capability$ corresponds to the type of service the consumer requested, S_c represents the capacity of resources required by the task, and S_l indicates the maximum time limit the service provision is allowed to take.

Using the $\mathcal{R}_p$ registry, the provider models the current service type demand based on its local interactions with consumers. Each such interaction is considered by the provider as a signal reinforcing provision of a specific service type S_t. During such an interaction, the provider updates the preference for that service type S_t within its registry according to the following formula:

$$s \rightarrow s + \psi,$$

where s is the stimulus value associated with S_t service type and ψ is a parameter defining the increase of the estimate value.

The provider periodically (in one second intervals) consults the $\mathcal{R}_p$ registry to determine whether or not to reconfigure its offered service. As there is no global view of current

service demand, each provider infers which service type to offer based on local demand observed from previously received requests. If the *type* of service that the provider decided to offer is the same as the service that is currently being offered, then no action is taken. Otherwise, the provider performs a *switch* operation, whereby the provider changes the type of service it can provide. Whilst this switching process has no explicit economic cost, it has an implicit cost as the process takes a finite T_s time, during which no other service can be provided.

To decide which service type to offer, the provider employs a probabilistic selection mechanism, according to which the probability p of offering a unique service type (S_t) from its $\mathcal{R}_p$ registry is determined based on the following equation:

$$p = \frac{s^2}{s^2 + \theta^2}$$

where s is the current stimulus value associated with the service type (S_t) kept within provider's $\mathcal{R}_p$ registry and θ represents the response threshold (that we explain in detail below) associated with the same S_t registry tuple.

The above applied formula is derived from the study of natural self-organising mechanisms governing the division of labour within insect societies (ant colonies) and describes a model of probabilistic response of an individual to perceived task stimulus (eg. food foraging or brood feeding) [7, 120]. In our autonomic system model, the obtained p probabilities correspond to the choice of provisioning a unique service type. Based on the calculated probabilities for each service type, every decision-cycle (every one second), the provider probabilistically (using roulette wheel section) selects the service type that will be offered.

Given the above formula for determining p probabilities, it is clear that the value of the obtained probability for each unique service type is dependent not only on the stimulus (s) corresponding to how many consumer agents requested such a service type but also to the response threshold (θ). Within natural self-organising systems such as insect societies, this parameter identifies the persistence of an individual to continue performing the same task that it was carrying out beforehand. In this context, it has been observed that such persistence is proportional to the time spent performing one particular task where the longer such an individual was carrying one particular activity (eg. food foraging in ant colony) the more resilient it was to switching to conduct a different activity. Within studies focusing on self-organisation in insect societies [7, 120], such reluctance (otherwise called specialisation) of an individual to conduct an other task than the current one is regulated through response threshold update functions that work in the following manner. For the task that is being currently carried out by an individual, its θ parameter value is decreased over time, whereas for other tasks that are not being currently performed, their θ values increase. Persistence facilitated in this manner allows for a smooth division of labour to occur within a collective of system

individuals that was initially unspecialised and unbiased to perform any task.

In our autonomic system model we extend the original threshold update functions such that the values of θ parameters are not only augmented through some static constants (as has been done for the insect societies models presented in [7, 120]) but also depend on the current provider agent utilisation level (U). This is done in the following manner. Each tuple within the provider registry is updated during every provider's decision cycle in the following manner:

$$\theta \rightarrow \theta - 2 \times (U + 5),$$

if the provider is currently offering the given service type and

$$\theta \rightarrow \theta + U + 5$$

otherwise, where U denotes the current provider utilisation level. Throughout the experimental evaluation we have identified that the best provider adaptation is achieved in conditions where $10 \leq \theta \leq 60$ and where the value of the constant used for the threshold update is equal to 5 (as presented in the above functions).

To ensure that the model maintained for current service demand does not become stale, a decay function is used to adjust the s parameter for all tuples in the $\mathcal{R}_p$ registry, using the decay coefficient Δ_{decay}.

A provider agent maintains a stress parameter ($\Omega_p \in [0, 1]$), the value of which is proportional to the frustration it experiences whilst deciding which service type to offer. If the agent reconfigures its resource provision on a frequent basis, its stress level is considered to be high. However, if the agent is persistent at offering a specific service type, and thus reconfigures rarely, its stress level is low.

To capture these intuitive relations in the form of provider stress (Ω_p), a measure of entropy from statistical mechanics is applied. This is facilitated in the following way. Each decision cycle (every second) during which the stress value is calculated, the provider normalises all stimuli (s) estimates contained in all N tuples within its $\mathcal{R}_p$ registry and applies these to calculate Shannon's entropy with the use of the following formula:

$$\Omega_p = \frac{-\sum_{n=1}^{N} s_n \log s_n}{\log N},$$

The obtained provider stress Ω_p is a real number kept in $\Omega_p \in [0, 1]$. If $\Omega_p \approx 0$, the stress level is low since there exists a strong stimulus, reflected by high stimulus (s) value for one specific service type and weak stimuli for other service types. However, when $\Omega_p \approx 1$, the stress level is high since the stimuli for all service types have similar values and thus the probability of selecting any service type is equal.

Algorithm 2 Provider service provisioning algorithm

Require: a provider α_p currently offering service of type $type_p$ with available capacity $capacity_p$, and the set $\mathcal{S}$, which contains tuples, r_t of all the known and unique service types, S_{type}, it may offer.

Ensure: an accurate to the local demand service *type* is offered

1: req_i := receiveRequest() {*where* $req_i = \langle type, capacity, timelimit \rangle$} {*service of certain type (type), capacity (capacity) and allocation time limit (timelimit) is required* }
2: **if** $(type_p \neq type \wedge (capacity_p < capacity)$ **then**
3: sendResonse(req_i,$REJECT$)
4: **else**
5: sendResonse(req_i,$ACCEPT$)
6: executeService(req_i)
7: $r_t := \langle S_{type}, s, \theta \rangle$ {*based on* req_i}
8: **if** $r_t \subseteq \mathcal{S}$ **then**
9: $r_t := \langle S_{type}, \max(1, s+1), \theta \rangle$ {*update stimulus for requested by consumer agent* S_{type}}
10: **else**
11: $r'_t := \{\langle S_{type}, 1, \theta \rangle\}$
12: $\mathcal{S} := (\mathcal{S} / \ r_t) \cup \ r'_t$ {*Update the Registry* $\mathcal{S}$}
13: **end if**
14: update response threshold (θ) for each $r_t \in S$
15: calculate probabilities for all $r_t \in S$
16: perform roulette wheel selection of a single $r_t \in S$
17: **if** $\mathcal{S}[0] \neq type_p$ **then**
18: PerformSwitch()
19: **end if**
20: **for all** $r_t \in \mathcal{S}$ **do**
21: $r_t := (\mathcal{S}_{type}, s \times \Delta_{decay}, \theta \rangle$ {*Decay stimulus value (s) of every* r_t *tuple*}
22: **end for**
23: **end if**

During service provisioning, both the time it takes to provision a particular service (T_e), as well as the time it takes to respond to a service provision query (T_q) is a dynamic property that is determined by the current provider's utilisation level (U). Since each provider is distinguished by its maximum resource capacity, its utilisation can be reflected as a fraction of capacity that is already used (employed by consumer agents). Given this, the execution time (T_e) is defined by the following formula:

$$T_e = T_e^{min} + U^2 \times T_e^{norm}$$

where T_e^{min} is the minimal service execution time ($T_e^{min} = T_e^{norm}/4$) and T_e^{norm} defines provider normal response time whilst being under utilisation equal to 1. The time it takes for a provider to respond to a service provision query (T_q), in turn, is derived from the following formula:

$$T_q = T_q^{min} + U^2 \times T_q^{min},$$

where T_q^{min} is a minimal response time incurred during the query.

In this model a provider will accept service provision queries if its utilisation (after provisioning of the requested service) will remain in $0 \leq U \leq 2$. This suggests that consumers may actually demand more capacity than the provider may offer. This is possible, since once the utilisation exceeds unity, computational resources (such as CPU, connections to database, etc.) that were previously dedicated to a single service provision now become shared between different service provisions. Although this enables the provider to satisfy more demand, it also causes the quality of the offered service to drop down in the form of provision time.

The algorithm used by a provider agent is represented in Algorithm 2. On receiving a service request (line 1), the provider verifies that it is currently able to provide the service (in terms of service type, capacity and task allocation time limit) before going on to execute the service[4]. Once the service execution (or reject) step has been completed, the provider updates its internal registry by creating a tuple, r_t, based on the request (line 7). If a set of records for the requested type exists (i.e., $r_t \subseteq \mathcal{S}$ in line 8), then either the new tuple is added to the set (line 12), or if a similar tuple exists (based on the service type (S_{type}) parameter value held in it), it is updated (line 9). The provider then updates response thresholds for all sets of tuples $\mathcal{S}$ and calculates probabilities for each such tuple based on the *stimuli-response* mechanism presented in this section (lines 14-15). This is followed by a roulette wheel based selection of a service type to offer (line 16) and potential switch operation if the service type selected for provision is not currently offered (lines 17-19). Finally, the provider decrements stimulus values for all tuples stored within its local registry (lines 20-22).

5.3.3 Information Exchange Mechanisms

To facilitate the migration of knowledge regarding the availability of services and current service demand, consumers share knowledge before revising their respective models. During the information exchange process, provider agents are used as information brokers through which the communicated information is passed to consumer agents co-located with them. In what follows, we explain how information is communicated by system peers and what local decision-making mechanisms are implemented for regulating this process in a decentralised manner.

Figure 5.3 shows the sequential steps involved in information exchange during a single

[4]The provider does not verify whether or not it can offer the desired service, if it is in the process of reconfiguring or *switching*. In this case, all requests are rejected until any currently executed services have been completed, and the provider has successfully changed its current service offering.

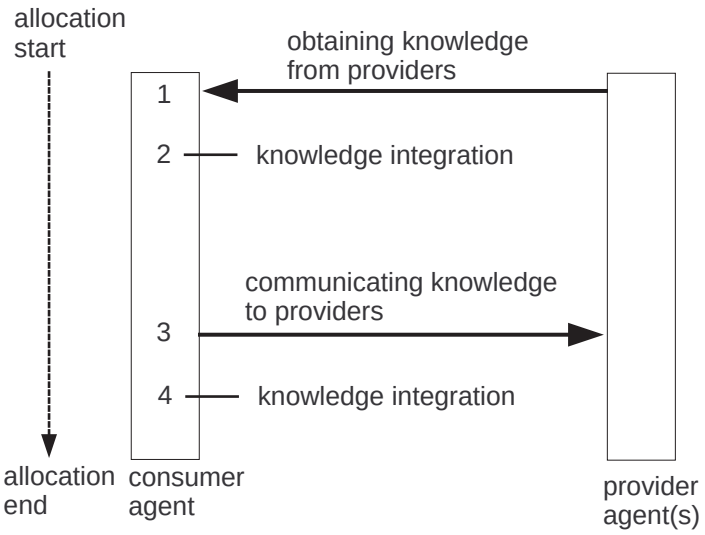

FIGURE 5.3: The diagram showing sequential steps performed by a consumer agent during information exchange activities. Actions are ordered according to their occurrence within the scope of a single allocation cycle and are repeated in the same order in the following allocations.

consumer allocation cycle. During each allocation cycle, a consumer agent performs two information exchanges (arrows show the direction in which the information flows), each followed by knowledge integration. The first information exchange act takes place when the agent obtains a new task to be allocated (indicated by number 1). At this stage, the agent might have been waiting for the task for a long period of time, and its local model might have become outdated. Therefore, before pursuing the task, it obtains knowledge from the provider it is co-located with. The second information exchange process (indicated by number 2) is conducted once the resource allocation cycle of a particular task completes. Since, throughout the task allocation period, the agent updates its internal model based on the outcome of interaction with providers, it communicates knowledge to a subset of selected providers. The choice of which providers the information should be communicated to in this process is made based on an *affinity algorithm* described in the remainder of this section.

Apart from communicating information, during each allocation cycle, the consumer performs two knowledge integration activities. The first (denoted in Figure 5.3 as step 2) takes place when an agent obtains information from a provider co-located with it. The knowledge integration takes place once the resource allocation cycle of a particular task completes (represented in figure as 4). Since, throughout the task allocation period, the agent might have communicated with a number of foreign registries from other agents, it will use that information (stored in a temporary list) to update its internal model.

Below we outline in detail the mechanisms that allow consumers to obtain, communicate and integrate knowledge.

5.3.3.1 Communicating knowledge to providers

At the end of every allocation cycle, consumer agents gossip with other agents. Knowledge communication is illustrated in Figure 5.4. A consumer $C5$ gossips information to the selected provider ($P2$). The information that is received by $P2$ is represented in the form of a tuple:

$$I = \langle R_c, \Omega_c, \alpha_p, \Omega_p \rangle$$

where R_c is a list of registry tuples originating from consumer $C5$, $\Omega_c \in [0, 1]$ represents consumer $C5$ stress level (defining how well it performs), α_p is the provider agent $C5$ is co-located with ($P1$) and $\Omega_p \in [0, 1]$ represents $P1's$ stress level (described in Section 8.2.2).

The communicated information (I) is then propagated by provider $P2$ to all consumers ($C1, C2, C3$) that are co-located with it. Apart from I, $C5$ also signals to $P2$ its currently demanded service type (S_t) that is being used by the latter to update its internal demand model.

During information communication, the consumer gossips the content of its local registry only to a subset of selected provider agents that have $affinity$ estimates (kept within consumer's $\mathcal{R}_c$ registry) higher than the default value (0.01) and greater than $0.1\lambda_{max}$, where λ_{max} is the highest *affinity* value the consumer has within its memory. The affinity estimates are updated by an affinity algorithm during the *tuple integration* process described in the remainder of this section.

5.3.3.2 Obtaining Knowledge from Providers

When obtaining knowledge from a provider, the queried provider propagates that query to consumers that are co-located with it and sends the obtained list of tuples $\langle I \rangle$ back to the consumer agent requesting it. The consumer also signals the provider about its current service type demand (S_t), allowing the latter to update its demand model.

During information exchange conducted by both mechanisms, only a subset of tuples contained within the consumer's local registry is communicated, according to the following rule. The consumer selects the tuples that have an evaluation score greater than $0.1\epsilon_{max}$, where ϵ_{max} is the highest evaluation score the consumer has within its memory. By doing this, low scored tuples are not communicated.

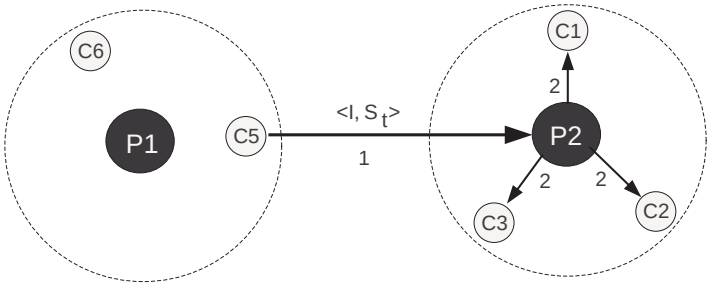

FIGURE 5.4: Information exchange between consumer agents. Two providers ($P1$ and $P2$) are represented with co-located with them consumers: $(C5, C6) \in P1$ and $(C1, C2, C3) \in P2$. Consumer $C5$ decides to communicate information to provider $P2$. During the communication act, the agent sends its personal registry content and the provider name it is co-located with in the form of a tuple $I = \langle R_c, \alpha_p, \Omega \rangle$. Provider $P2$ propagates received in this form information to co-located with it consumers $(C1, C2, C3)$ (step 2). During the communication act, the consumer additionally signals to the provider service type (S_t) it is currently interested in allocation. This information is used by the provider to update its local demand model.

5.3.3.3 Tuple Integration

During the tuple integration process the consumer updates its internal model based on a subset of tuples selected from the list $\langle I \rangle$ that were obtained during information exchange.

As consumer agents are bounded, the processing of any inflowing information consumes an agent's computational resources (such as CPU, time or memory space) and there is a limit on how much information the agent is allowed to process before its task allocation efficiency becomes compromised. In this model, such a limitation is represented by the upper bound (defined by L parameter) on the number of information tuples the consumer will consider during tuple integration procedure. If, at the beginning of tuple integration, the size of a list $\langle I \rangle$ is greater than L, the consumer will randomly (using uniform distribution) remove tuples from this list, until its size matches L.

Since there are two knowledge integration activities during each task allocation cycle (before and after allocation process as presented in Figure 5.3), it is assumed that the limit of maximum foreign information registries that can be accepted during each procedure is equal to $L/2$.

5.3.3.4 Information Merge for an Agent with a Default Knowledge Model

If the consumer agent is a newly created agent, experiencing the first information merging process and thus has a default internal model (ϵ and λ values for all tuples originating from its internal registry are set to default 0.01 values), the agent will update the ϵ for each α_p provider contained within its registry according to the following formula:

$$\epsilon_{\alpha_p} \rightarrow \frac{\sum {\epsilon_{\alpha_p}}^*}{n}$$

where ϵ_{α_p} represents an evaluation score for provider α_p the consumer stores within its local memory and $\sum {\epsilon_{\alpha_p}}^*$ is the sum of n evaluation scores for the matching provider obtained from all inspected I tuples selected for the merging process.

The affinity scores are updated according to the following formula:

$$\lambda_{\alpha_p} \rightarrow \frac{\sum {\lambda_{\alpha_p}}^*}{n}$$

where λ_{α_p} represents affinity associated with α_p provider stored within its internal registry tuple and $\sum {\lambda_{\alpha_p}}^*$ is the sum of n $affinity$ scores for the matching α_p provider obtained from all inspected I tuples the agent decided to merge with its registry.

Finally, the stress level (Ω_c) of a newly deployed agent is determined by the mean stress of other consumers exchanging information with it:

$$\Omega_c \rightarrow \frac{\sum \Omega_c^*}{n}$$

where Ω_c represents the stress level of a consumer agent from which communicated information originates and $\sum \Omega_c^*$ is the sum of n consumer stress estimates originating from all I tuples the agent decided to merge with its registry.

5.3.3.5 Information Merge for a Non-default Knowledge Model

If a consumer agent already experienced at least one information merging process (and thus *evaluation* and *affinity* scores have already been modified during this process), it will update the evaluation for each α_p provider contained within its registry according to the following formula:

$$\epsilon_{\alpha_p} \rightarrow \frac{\epsilon_{\alpha_p} + \frac{\sum {\epsilon_{\alpha_p}}^*}{n}}{2}$$

where ϵ_{α_p} represents an evaluation score for provider α_p the consumer stores within its local memory and $\sum {\epsilon_{\alpha_p}}^*$ is the sum of n evaluation scores for the matching provider obtained from all inspected I tuples selected for merging.

The affinity scores are updated according to the following formula:

$$\lambda_{\alpha_p} \rightarrow \lambda_{\alpha_p} + {\lambda_{\alpha_p}}^*$$

where λ_{α_p} defines internal consumer affinity score for a α_p provider and $\lambda_{\alpha_p}{}^*$ defines the affinity value (calculated using an *affinity* algorithm) for α_p provider from which the external registry selected for the introduction originated.

The consumer's stress (Ω_c) is determined by the following formula:

$$\Omega_c \rightarrow \frac{\Omega_c + \frac{\sum \Omega_c^*}{n}}{2}$$

where Ω_c represents a consumer's local stress estimate and $\sum \Omega_c^*$ is the sum of n stress scores originating from information exchanging consumers.

5.3.3.6 Information Outflow Regulatory Mechanism

Consumers reveal information kept within their local registries only to a subset of provider agents with sufficiently high affinity (λ) scores assigned to them by the former agents. Whereas the manner in which consumer employs affinity scores during information outflow is described in Section 5.3.3.1, here an *affinity* algorithm that is responsible for λ scores update is introduced and described in detail.

The update of affinity scores is conducted on the basis of the comparison of foreign information communicated to the agent (I) with the knowledge locally maintained by the consumer. To do so, each consumer agent compares every foreign registry $\mathcal{R}_c^*$ from the tuples selected for processing I tuples ($I = \langle \mathcal{R}_c^*, \alpha_p \rangle$) with its internal registry $\mathcal{R}_c$. The comparison is conducted according to the following algorithm:

1. For each α_p provider located in the internal consumer registry, for which $\epsilon > 0.01$, a matching α_p provider is located in the foreign $\mathcal{R}_c^*$ registry. If the evaluation score of the provider within the foreign tuple is also greater than the minimal value ($\epsilon^* > 0.01$), the affinity of the pair of tuples (λ_{α_p}) is calculated with the use of following formula:

$$\lambda_{\alpha_p} = 1 - |\frac{\epsilon_{\alpha_p} - \epsilon_{\alpha_p}{}^*}{\epsilon_{\alpha_p} + \epsilon_{\alpha_p}{}^*}|$$

2. Using this algorithm, the affinity between every matching pair of tuples is calculated. The aggregated affinity score:

$$\lambda_{\alpha_p}^* = \sum \lambda_{\alpha_p}$$

   is then used to update affinity score of the tuple representing provider α_p located within consumers local registry:

$$\lambda_{\alpha_p} \rightarrow \lambda_{\alpha_p} + \lambda_{\alpha_p}^*$$

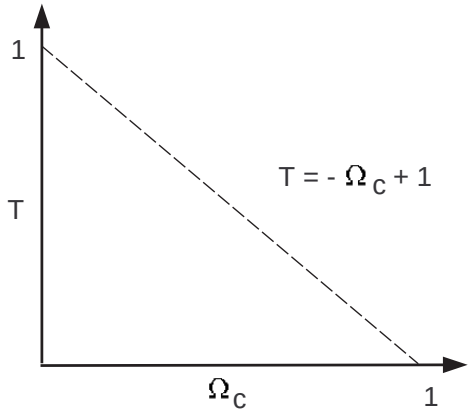

FIGURE 5.5: Function regulating inflow of foreign information to consumer agents. In here the value of threshold T, below which consumer will accept information from the offering it provider is a function of consumer's stress (Ω_c).

As the affinity algorithm involves a comparison between local consumer registry and the registries obtained through information inflow, it is assumed that a consumer agent decides to apply this algorithm only if its internal model is non-default (has already experienced at least one information merging process). In the situation when a consumer agent has been initialised and contains default knowledge, instead of relying on the affinity algorithm to update the affinity scores within its local $\mathcal{R}_c$ registry, the consumer relies on the affinity score information communicated from other agents. This information is then incorporated into its own registry (for details on how information is merged for a default knowledge model see Section 5.3.3.4).

5.3.3.7 Information Inflow Regulatory Mechanism

As the inflow of foreign information to consumers will affect their local knowledge and thus the selection of provider agents during the task allocation process, the consumers are provided with local information inflow regulatory mechanisms that decide which foreign registries become accepted and merged with the one internally held by the agents.

This is facilitated in the following manner. Consumer agents accept information only from providers that have their stress level below $T \in [0, 1]$ threshold ($\Omega_p < T$), where T is determined by each consumer individually through the following function:

$$T = -\Omega_c + 1$$

where Ω_c represents the individual stress level experienced by the consumer agent. The detailed description of the procedure used by agents to calculate their stress level (Ω) is described in Section 5.3.1 (consumers) and Section 5.3.2 (providers).

Figure 5.5 illustrates the relationship between the T threshold value and the level of consumer stress (Ω_c). As shown, the regulatory mechanism imposes constraints on the inflow of foreign information when a consumer's stress increases and relaxes the constraints otherwise. Such behaviour is motivated by the observation that providers which exhibit low stress are those that specialised to offer one service type and thus are employed as information intermediaries by consumer agents interested in that service. Providers with high level of stress, on the other hand, are likely to switch to offer different service type and, as a consequence, are likely to mediate knowledge from consumer agents interested in different service types. Such information is considered as a threat as it will confuse consumers accepting it to interact with providers that possibly offer different types of services than they are after and thus destabilise the resources market. In order to facilitate minimal information flow when this mechanism is in use, it is assumed that a consumer agent always accepts at least one foreign gradient during knowledge integration.

5.4 Experimental Setup

5.4.1 Consumer Turnover Mechanism

One of the most important features of autonomic systems is their ability to adapt to changing conditions without human intervention. The most obvious environmental dynamics such systems will be required to counteract relates to variance in service type demand. Such pressure is triggered on the system by the influx of new infrastructure users demanding various service types that may be offered by the system, and requires efficient reconfiguration of resources market in order to satisfy the demand for particular type of requested services. In our model the change in service type demand is triggered by the introduction of new consumer agents. This process is handled by the *consumer agent turnover* mechanism described below.

During each task arrival to a consumer agent, a random number $r \in [0, 1]$ is drawn from a uniform distribution. If the value of r is greater than the value of $1 - \chi$, the agent to which this task was despatched is removed from the system and replaced by the new one. If, on the other hand, $r < 1 - \chi$, the task is handled by the consumer agent already existing within the system.

Given this simple procedure, the consequences of introducing a new agent are twofold:

1. The newly deployed agent does not possess any information about the availability and performance of providers existing within the system. To obtain this information, it becomes co-located with a randomly chosen provider, under the condition that this provider is currently offering the service type demanded by the task the

consumer has to allocate [5]. Although this does not guarantee that the co-located provider will be available and capable of satisfying the task demand, by offering the same service type the consumer is interested in, the provider may already host other similar consumers that, through gossiping, may provide information about the state of other potentially valuable providers to the new agent.

2. The type of the service that the newly introduced agent is required to allocate is selected according to one of the three demand functions (step, sinusoidal and stochastic) that we discuss in a greater detail in Chapter 7 in which we specifically consider the problem of adaptive service provisioning.

The motivation for demanding each consumer agent to allocate the same service type during its lifetime is made on the basis that tasks submitted by the same infrastructure user correspond to the series of interactions with the same application, eg. email client, instant messenger, photo application and are handled by the consumer agent initialised for this user. However, depending on the operation the user is performing (eg. viewing images, uploading images or processing images), tasks submitted by him may still differ with respect to the required capacity (S_c) or the time limit the user wishes the task to be completed (S_l). For a more detailed model operation overview see Section 5.3.

5.4.2 Service Supply Setup

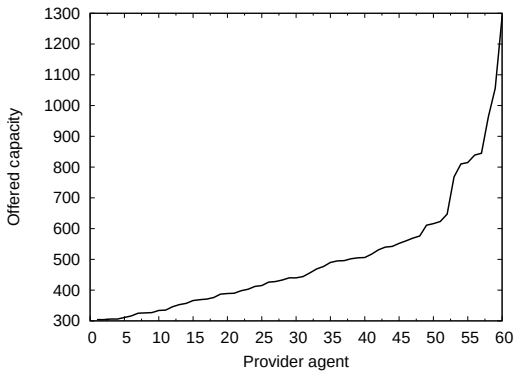

FIGURE 5.6: Capacity levels of deployed within the system providers based on exponential probability distribution.

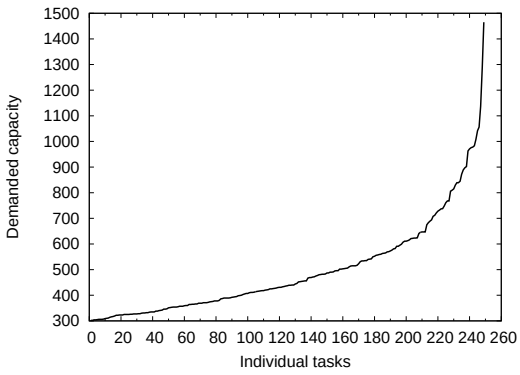

FIGURE 5.7: Capacity levels demanded by 250 tasks introduced to the system. The distribution of task capacities is drawn from an exponential probability distribution.

Resource Capacity Limitation

Providers differ from each other with respect to the resource capacity they offer. The motivation for such uneven resource distribution follows from the observation that within

[5] In cases where no such provider exists, a randomly chosen (using a normal probability distribution) provider from the population of all deployed agents is selected.

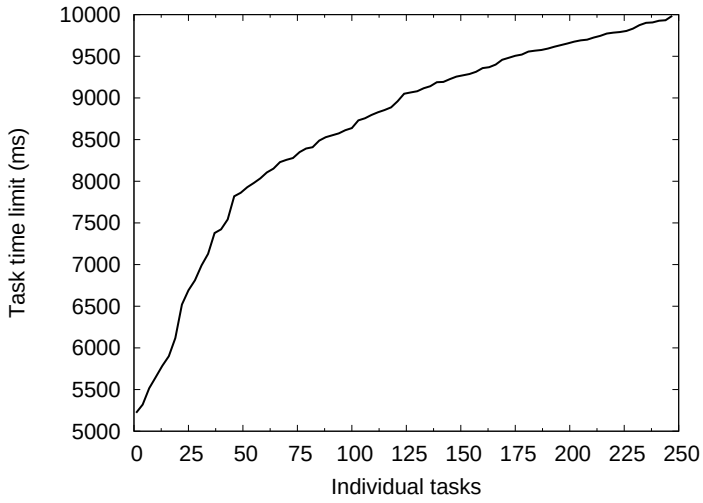

FIGURE 5.8: Distribution of task time allocation deadlines (S_l) drawn from an exponential probability distribution.

real computational systems there will be a large number of nodes offering relatively small resource capacity and a small number of nodes that offer a large resource capacity. Consequently, to model these conditions within our autonomic system model, we assume that the distribution of capacity across providers (more specifically — the computational nodes they manage) is approximated by the exponential probability distribution function. This is achieved as follows. Throughout all experiments conducted in the following chapters, we assume that the total resource capacity offered by the whole system (Σ) remains constant and equal to 29500 resource capacity units. However, as Figure 5.6 shows, such total resource capacity is distributed across individual providers based on the exponential distribution.

5.4.3 Service Demand Setup

To reflect the dynamic nature of resource allocation, tasks that are issued to the system differ from each other with respect to task demanded capacity (S_c), task completion time limit (S_l) and task arrival rate. Below we explain how these values are dynamically determined within our model.

Task demanded capacity (S_c) setup

Figure 5.7 illustrates task capacities that are demanded by 250 consumer agents. Since throughout all experiments we keep this number fixed (but allow consumers to undergo turnover process), the disposition of capacity levels across different tasks, that is illustrated in the figure, reflects conditions in which infrastructure users have different demands with respect to resource capacity required to fulfill their goals. As such non-uniform division of capacity levels is provided on the basis of exponential probability

distribution (the same used for distributing resources across the population of providers, illustrated in Figure 5.6, there is no guarantee that consumers will be able to satisfy their requests throughout their lifetime relying only on a single provider as its available resources may not be sufficient for probabilistically set capacity levels demanded by new tasks. However, it is assumed that the total capacity that is demanded in this manner by the population of consumer agents, on average, is close to the total supply of system resources $\Sigma = 29500$, meaning that there exist enough resources within the system to satisfy the demand.

To match the capacity demanded by all consumer agents to the total system capacity supply, the capacity of each newly injected task to the system is normalised according to the following formula:

$$S_c = S_c^* \frac{C}{P} D,$$

where S_c^* is the capacity drawn from exponential distribution, C is the current number of consumer agents within the system, P is the current number of provider agents within the system and D is the weight ($[0, 1]$) that is used to define the actual capacity demand level (eg. if set to 1 will indicate that the system capacity demand-supply ratio is equal unity).

Task completion time limit (S_l) setup

The task completion time limit is not uniform for every task issued to the system but varies according to an exponential function. Such a distribution is used to model the situation where the proportion of users demanding tasks to be completed in a short time interval is relatively small as compared to the majority of users demanding tasks to be completed at a more affordable (longer) periods of time. An example of 250 task completion time limits drawn from this distribution is illustrated in Figure 5.8.

Task Arrival Rate Setup

The time intervals between which new tasks will be dispatched to a consumer agent are determined probabilistically using Poisson distribution, with the mean defined by ω parameter. If the consumer is already allocating a task, the newly arriving one will be added to its local queue.

5.4.4 Agent Strategies Setup

Depending on the manner in which consumer agents employ (or not) information exchange mechanisms, we consider three general model configurations, the efficiency of which, in achieving desired system functionality, will be evaluated during experiments:

AdaptiveFlow model (henceforth called AF, $FreeFlow$ model (FF) and $NoFlow$ model (NF). Configuration of each of these models is explained below.

1. **AdaptiveFlow** model: A model in which there exists a communication of information between agents and where the transfer of information is regulated by adaptive mechanisms. Here, the amount of foreign information accepted by consumers is regulated through the information inflow regulatory mechanism described in Section 5.3.3.7, whereas the decisions to which other agents the information is gossiped is regulated through the information outflow regulatory mechanism, described in Section 5.3.3.6. In the remainder of this chapter we will refer to this model configuration as the AF model.

2. **FreeFlow** model: A model in which there exists a communication of information between agents but with no regulatory mechanisms deciding which information to accept and to which other peers to communicate it further. In this setup consumers accept all the information inflowing to them and propagate it to a randomly selected subset of known providers. In each allocation round, the number of peers to which the information is communicated is selected on a random basis (using uniform distribution) from a range of $< 0, 30 >$ [6]. In the remainder of this section we will refer to this model configuration as the FF model.

3. **NoFlow** model: A model in which there is no communication of information between system agents. In this setup consumers are allowed to perform their local learning but are not allowed to communicate their locally maintained knowledge to other agents. In the remainder of this section we will refer to this model configuration as the NF model.

On initialisation of each of these models, or new consumer deployment (during consumer agent turnover), consumers are provided with full information about the existence of all deployed within the system providers, but possess no knowledge about their current configuration or availability. Such knowledge is established at run-time by the exchange of information and local learning mechanisms. Similarly, providers (unless stated otherwise) have no preference for providing any service and decide to offer service types based on their local learning mechanisms and locally perceived information about consumer demand. Once the model is initialised, consumers start their allocation within the randomly determined time between the first $50s$ of simulation time. This non-uniform initialisation is applied to avoid any possible activity synchronisation between consumer agents. Once consumers become initialised, the task arrival rate is regulated through the Poisson distribution set by the ω parameter.

[6] The selection of 30 agents as an upper range of maximum number of peers to which information is communicated defined the half of the total population of providers deployed in all conducted experiments.

It is important to state that the purpose of comparing the three above 'flavors' of the same model, each distinguished by the combination of strategies that agents employ during their decision-making, is done strictly on exploratory grounds. This means that we are are interested in investigating the dynamics of a particular decentralised multi-agent system model presented in this chapter and the influence of AF, NF and FF strategy combinations on its behaviuor. In doing so we do not provide any comparison criteria between other existing approaches and multi-agent system models addressing alike resource management problem. We assume that such an engineering approach can be done only when a sufficient understanding of our model dynamics and the role of local decision-making mechanisms responsible for this is obtained.

5.4.5 System Constants and Parameters

The experiments that will be presented in the remainder of this thesis will involve various parameter settings used to evaluate the efficiency of applied decision-making mechanisms under different resource allocation conditions. As this will involve manipulation of various parameters, whilst keeping others unchanged, below we outline which model settings we will consider as constants throughout the experiments (and thus will not refer to them) and which will be used as parameters (hence referred to in experiments in which they are modified).

5.4.5.1 System Constants

Table 5.1 lists the constants that will remain unchanged for all experiments, unless state otherwise.

5.4.5.2 System Parameters

Table 5.2 lists the default model parameter values. In cases when we modify the values of any of these parameters, we will explicitly state this during specific experimental setup.

5.5 System Behaviour Analysis Measures

5.5.1 System Throughput

The number of successfully allocated tasks is computed by measuring both the number of successful and failed allocations each consumer agent experienced. These numbers are measured in $20s$ time intervals. Each time another time window is selected, it starts from

Parameter	Value	Description
C	250	Total number of consumer agents
P	60	Total number of provider agents
Σ	29500	Total amount of offered by the system capacity
L	20	Upper bound on the number of considered foreign registries during tuple integration
D	1.0	Demand level parameter identifying the global demand—supply ratio
U	2	Maximum allowed provider over-utilisation level
ψ	1	The *stimulus* increment coefficient
T_q	250ms	Time taken to query a provider (varies according to provider utilisation level)
T_e	4s	Service execution time (varies according to provider utilisation level)
T_s	4s	Time taken for a provider to perform a *switch* operation
T_o	8s	Time taken for a provider to move into off-line/on-line mode (used for power management functionality described in Chapter 8)

TABLE 5.1: Default model constant values used throughout experiments.

Parameter	Value	Description
Capability	10	Number of unique service types offered by the system
ω	7s	Task arrival rate (mean value of Poisson distribution)
δ_{decay}	0.9	Decay coefficient used to allow stale information to decay within consumer registry
Δ_{decay}	0.7	Decay coefficient used to allow stale information to decay within provider registry

TABLE 5.2: Default model parameter values used throughout experiments.

the middle of the latter sampling period; eg. if the first sampling period was between $0 - 20$ initial simulation seconds, the second sampling period is between $10 - 30$. In each such sampling period, both successes and failures are averaged over the number of all consumer agents; the number of successfully allocated tasks, which defines *system throughput*, is calculated by subtracting the failed tasks from the successfully allocated ones. Thus, the system that experiences more failures than successes is assumed to achieve throughput equal to 0.

5.5.2 Agent Communities

The stability and efficiency of the system is dependent on the establishment of correct interactions amongst system agents (consumers and providers). Given the fact that during each interaction between consumer-provider pairs the information perceived by

the provider is communicated to consumers employing it, consumer-consumer interactions are considered as constituting the main underlying network through which the information about the system state flows across the agents.

Provided this information flow network it is clear that only certain configurations of interactions will allow agents to organise their service provision and consumption. For example, it is unlikely to expect that consumer agents that are interested in allocating different service would benefit from sharing their local knowledge models as they may risk the situation of attracting other (interested in different service type) consumers to engage with the reliable provider they were currently using. As a consequence, the provider might decide to switch and start offering different service type that would further destabilise the performance of the previously employing it consumers.

To identify whether, and under what conditions, consumer agent interactions are able to relax into topological configuration in which information that is communicated to other peers does not destabilise the system performance, we provide measures for extracting community structure from the captured consumer-consumer interactions. In here, we consider a community to be a subset of individuals that have more links to other members of the community than to individuals from the remainder of the network [79]. Given this, we assume that communities consisting primarily of like-minded individuals, represented here by consumers interested in allocating the same service type, may support the emergence of cooperation, by providing an environment in which co-operators are more likely to interact with other co-operators and less likely to be exploited by defectors [30].

Given this interaction network, the purpose of the *community extraction* measure is thus to identify the characteristics of such a network, including:

1. identification of any sub-groups of agents that interact (and thus share their information) more frequently between themselves than any other system agents (such groups of agents are onwards referred to as communities);

2. characteristics of these communities, including:

   (a) composition of the community, identifying whether consumer agents (distinguished by the service type they are interested in allocating) are of the same type and thus how homogeneous with respect to this the community is;

   (b) average community size;

   (c) number of identified communities.

To capture these properties, we rely on the Girvan-Newman community extraction algorithm [79]. The community identification procedure is as follows:

1. During every goal allocation cycle, every consumer agent maintains a list of interactions that it experienced and during which it was communicated with new information. The list comprises the identities of consumer agents from which this information originated [7].

2. Using above collected data, a network is reconstructed identifying the topology of interactions, where nodes represent consumers and edges are drawn between those consumer agents that interacted with each other. The network is constructed on the basis of information collected within $10s$ sampling intervals within which, once an interaction between a given pair of nodes (agents) was identified, the algorithm stops further search of interactions for that pair of nodes [8].

3. A Girvan-Newman community extraction algorithm is applied to the network topologies extracted in subsequent simulation time snapshots.

After community extraction completes, the obtained results are processed in the following manner:

1. as a community that was captured by the algorithm, we consider a group of at least 2 consumer agents;

2. for each captured community, we extract information about its *size* and *composition*, identifying the names of consumers and service types they demanded at this simulation time snapshot;

3. knowing what service types were demanded by community members, we calculate *community homogeneity*. To do so, we identify the most common demanded service type within the community by dividing the number of agents interested in it by the sum of all other demanded service types.

5.5.2.1 Provider constraint measure

To identify constraint imposed on the *behavioural repertoire* of provider agent population during its service type provision, below we introduce the measure of *constraint*.

Each $20s$ time intervals provider's service type demand estimates (s) [9] are normalised

[7] This information is collected only for statistical purposes, where during model operation, consumer agents are not aware of this information and are not exploiting it in order to leverage their allocation efficiency.

[8] The motivation for $10s$ interval length is based on the fact that on average, each allocation cycle lasts for the time defined by task arrival rate (ω) which in all experiments is set to $\omega = 7s$, thus allowing for at least one allocation to take place within the network interaction sampling period.

[9] These estimates are kept in provider agent R_p registry and represent its attraction towards offering a particular service type. Recall that the provider relies on a roulette wheel selection when deciding which service type to offer.

and applied to calculate Shannon's entropy with the use of the following formula:

$$H = \frac{-\sum_{n=1}^{N} s_n \log s_n}{\log N},$$

where s_n is the demand estimate for particular service type and N denotes number of unique service types the provider is capable to offer.

The obtained value is a real number kept in $H \in [0, 1]$. If $H \approx 0$, the entropy is low (and thus constraint high) since there exists high stimulus (s) value for one specific service type and weak stimuli for other service types. However, when $H \approx 1$, the entropy experienced by the agent is high (and thus the constraint low) since the stimuli for all service types have similar values and thus the probability of selecting any service type is equal.

The entropy measurements obtained in this manner are then averaged over the population of all provider agents that have been considered during the given time interval and the overall provider constraint is reflected by the following formula:

$$P_H = 1 - H_{avg},$$

where H_{avg} represents the mean entropy experienced by provider population.

Chapter 6

Load Balancing

6.1 Introduction

In the previous chapter we presented our decentralised autonomic system model that we now apply to facilitate following autonomic system functionality: load-balancing, adaptive service provisioning and power management. Whereas the local decision-making mechanisms facilitating adaptive service provisioning and power management are discussed in the following two chapters, in this chapter we focus our attention on achieving load-balancing.

Load-balancing is one of the most common problems that faces modern service oriented systems and, in general, relates to the efficient allocation of system resources among a group of requesting them consumers [91]. The distributed nature of such systems, in which resources are both offered and consumed by multiple actors makes it natural to approach the load-balancing problem using a multi-agent system model. Here, individual agents are tasked to manage both the balanced provision of system resources as well as their reliable consumption on behalf of the infrastructure users. Achieving the efficient load-balancing using such distributed model relates to the development of local decision-making mechanisms that allow individual consumer agents to discover the best allocation such that all consumer requests are satisfied and neither of provider agents becomes unnecessarily overutilised. However, as has been observed by Hogg and Huberman in [40], realising this optimal match between resources demand and supply is a challenging task, especially if resource allocation is conducted within dynamic demand conditions. In these situations the approaches that stress the adaptation and thus the autonomy of individual agents are favoured over the ones that impose centralised solutions that are less scaleable and responsive to changes [91].

The application of decentralised multi-agent system for achieving efficient load-balancing has been discussed by Sen *et al.* [106], who consider a system of self-interested agents allocating resources on the basis of limited knowledge about the global system state. In

this context, Sen *et al.* investigate the effects of limiting agents access to knowledge about the state of system resources, and the resulting outcome on system resource utilisation.

In [40, 41], Hogg and Huberman examine the effects of local decision making on balanced resource utilisation within a computational ecosystem represented by a population of resource allocating agents. In this work, the authors demonstrate how imperfect information about resource state can lead to chaotic system behaviour and how this can by suppressed through appropriate local decision-making mechanisms. Another strategy, relying on local learning mechanisms designed to preserve energy minimising resource allocation within a mobile ad-hoc network, is presented by Brueckner and Parunak [11]. In achieving local mechanisms that allow the system to reconfigure its resource allocation in a manner that minimises power consumption, Brueckner and Parunak draw their inspiration from self-organising properties of natural systems (insect colonies).

The approach adopted in our work shares the same motivation of understanding how global system stability, in the form of balanced access to resources, can arise when autonomic system elements perform resource allocation independently, i.e., without centralised executive control. However, whereas previous work has investigated how the heterogeneity of the system can lead to stability through either limiting the knowledge possessed by individual agents [106] or by designing local decision procedures that diversify agent behaviour [40], here we focus on identifying the role of local information exchange between system agents as well as conditions under which such information flow can organise agent interactions and lead to efficient load-balancing.

Given this, we hypothesise that the efficiency of such a bottom-up approach depends on the ability of agents to organise into communities. The knowledge that is locally gossiped between the community members not only increases their individual awareness about the availability of provider agents but also allows them to avoid resource competition that could potentially arise if agents were unable to establish such organisations or relied only on their personal and highly limited knowledge.

To this end, the chapter is organised as follows. In Section 6.2 we explain the load-balancing problem and the role that agent communities play in addressing it within a decentralised multi-agent system model, while Section 6.3 presents experimental results, focusing on our model's load-balancing efficiency and the above stated hypothesis evaluation. The analysis of agent communities is then presented in Section 6.4 and their impact on individual agents performance discussed in Section 6.5. Finally, the chapter ends with a discussion in Section 6.6.

6.2 Load-balancing

6.2.1 The Load Balancing Problem

Achieving load-balancing in a decentralised manner is a non-trivial challenge that, if not properly approached, may result in poor system scaleability or an unexpected loss of system performance. In particular, it has been observed [40] that the introduction of shared and constrained resources into a system composed of a large number of independent and concurrent components responsible for offering and consuming resources on behalf of system users may quickly and unexpectedly lead to the emergence of undesirable system behaviour, often described as *resource competition* [94]. In this state, consumer agents end up competing for specific subsets of resources, leaving others under-utilised. This results in poor global resource utilisation reflected by the inefficient load-balancing over the set of available resource providers. Furthermore, because there is no centralised control, the system may simply self-reinforce this competitive behaviour leading to a very rapid degradation of system performance [41].

A close investigation of this problem shows that inefficient resource utilisation on the system scale arises from poor decisions made by individual system elements concerning their selection of resources [12]. In particular, in large scale IT systems it is typical that individual consumer and provider agents are not the same but demand (and offer) different amounts of resources. Consequently, as observed in Chapter 4 during heterogeneity tests, inappropriate allocation of consumer requests to service providers that have less resources (or more than required by the consumer) may either overutilise or underutilise particular system servers. This unbalanced distribution of resources not only decreases profit gained from service allocation (as overutilised servers offer degraded quality of the service or simply reject the allocation queries) but also prevents maximal usage of the system services because resources of the underutilised servers are not used to their full extent. In addition, this inefficient match between resource consumers and suppliers introduces additional infrastructure cost of maintaining surplus of available but unused system resources.

This presents us with the problem of developing local decision-making mechanisms capable of suppressing competitive interactions between resource consuming elements. Where competition cannot be avoided, for instance where demand for resources outstrips supply, such resource allocation mechanisms should effectively distribute service provision across the system, thus preserving *fairness*.

In this thesis we explore decentralised autonomic system models where individual consumers and providers demand (and offer) different types and quantities of resources that need to be completed within a limited period of time. Considering the fact that consumer agents within such model possess only local knowledge about the availability of system providers, but are able to communicate this information to others, it is clear that

allowing them to reveal their personal provider evaluations to others will improve their up-to-date knowledge about the availability of system resources. However, recall that within our model (described in Chapter 5) the only information that is shared across consumer agents is in the form of evaluation scores reflecting the efficiency of particular providers and does not include any additional information what service type this provider is offering [1]. As a consequence, if this limited information is shared among consumers that are interested in allocating different types of services, it may also introduce resource competition as neither of consumer agents possesses the information what service type the provider is currently offering and, as a result of this, will attempt to employ such provider thus consuming time for unsuccessful interaction.

Hypothesis 1:

Given these considerations, we should expect the best system efficiency in conditions when consumer agents organise their information exchange such that it is communicated only within a subset of peers that have the same service type interests. By doing this, the collectively shared information about provider's availability would increase individual agents awareness about system resources availability as well as enhance cooperation on the level of such a group, as every consumer would act in a manner that improves its own as well as other group members efficiency.

Since this approach has the potential to facilitate efficient load-balancing in a bottom-up manner that does not require any central control to be imposed within the system, our aim is to examine the above stated hypothesis. To do so, in what follows we experiment with three different model configurations [2] that is AF (**A**daptive **F**low), NF (**N**o **F**low) and FF (**F**ree **F**low). In the AF model consumer agents employ *affinity* algorithm that allows them to identify like-minded agents (interested in the allocation of the same service type) and share their local provider evaluations only among these peers thus, in principle, allowing for the desirable agent communities to emerge. The consumer agents from the FF model do not employ *affinity* algorithm and, as a result, share their local provider evaluations to a randomly chosen subset of provider agents that are selected from the ones existing within the system. As a consequence, this prevents any stable communities to be formed and introduces a risk of resource competition since consumers interested in different service types may decide to share their local provider evaluations. Finally, in the NF model configuration the consumer agents are configured to rely only on their local knowledge and are not allowed to share it with any other system peers.

In what follows, we evaluate the efficiency of these models in conditions where the system is open and new consumer agents are allowed to enter and consume different types of

[1] We assume that consumers do not share information what service types are offered by particular service providers because maintaining such up-to-date information within a dynamic resource environment would be difficult and could impact on system performance as agents could use stale knowledge

[2] More detailed description of these model configurations can be found in Chapter 5 (Section 5.4.4).

system resources. Under these circumstances we are interested in identifying which of the aforementioned models achieves the best efficiency as well as adaptation to changing conditions.

6.2.2 Load Balancing Performance Measures

Since in this chapter we are only focusing on the load-balancing problem, provider agents are configured to offer only one type of a service that is fixed over their life-time, and are not allowed to reconfigure their provision at run-time. However, in each experiment there exist 10 different sub-sets of provider agents, each offering a unique service type and there are 10 sub-sets of consumers interested in allocation of such unique service types. Given this setup, the main responsibility of consumer agent population is to identify groups of providers that offer service types demanded by them and effectively balance the requests to the appropriate and available provider agents.

The efficiency of load-balancing is evaluated in system configurations in which distribution of resources capacity between individual provider agents is not uniform but varies according to an exponential distribution function described in Section 5.4) [3] and where the system is open and thus new agents are introduced based on a consumer agent turnover mechanism (described in in Section 5.4.1). Finally, all experiments are set up in a way that the total demand imposed by consumer agents is proportional to the total supply of resources offered by provider agents (demand-supply = 1 : 1). The detailed parametrisation of employed models is presented in Section 5.4.5.

Given the above experimental setup, as an indicator of efficient load balancing we will apply three performance measures: system efficiency reflected by its throughput, system organisation identified by the existence of agent communities and finally, the number of rejected requests obtained by consumers during provider querying process. The first two measures are explained in Section 5.5. The purpose and method of applying the last measure is described below.

The rejected requests measure is motivated by the observation that each query in real autonomic systems may introduce additional task allocation costs (eg., consumed bandwidth, energy or time) which, if the tasks are required to be allocated successfully and in an on-demand fashion, needs to be minimised by the infrastructure provider through the application of control mechanisms. In all three discussed model configurations (AF, FF and NF) consumer agents already rely on simple adaptive mechanisms that allow them to continue to search for available providers until the one capable of satisfying the task constraints imposed by the task (service type, maximum allocation time limit and resource capacity) is found. However, although this enables consumers to avoid already

[3] The distribution of capacity for each of the sub-set of provider agents offering unique service types is such that despite the variance in capacity offered by individual provider agents, the total capacities offered by each group of providers are equal.

heavily overutilised providers that cannot satisfy the stringent task specification, we assume that provider query (and provider allocation) consume T_q query (and T_e execution) time [4] and thus inefficient (eg., brute force) search for available resources will eventually result in task failure.

Consequently, model configurations that are able to successfully operate on locally available information about resource state and coordinate their resource selections in a manner that avoids resource access conflicts should be identified by a low number of rejected allocative queries.

The following experimental evaluation section is broken down into two parts. In the first part (Section 6.3) the system throughput for three aforementioned model configurations is evaluated in a range of dynamic resource allocation conditions, whereas in the second part (Section 6.4) we focus on the identification of agent communities and model parametrisations under which they emerged together with their role in achieving high system throughput.

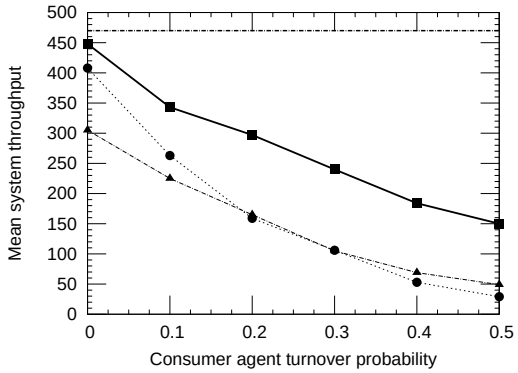

FIGURE 6.1: Figure illustrates mean system throughput as a function of increasing consumer agent turnover probability for three system model configurations: AF model (line with rectangles), NF model (line with circles) and FF model (line with triangles).

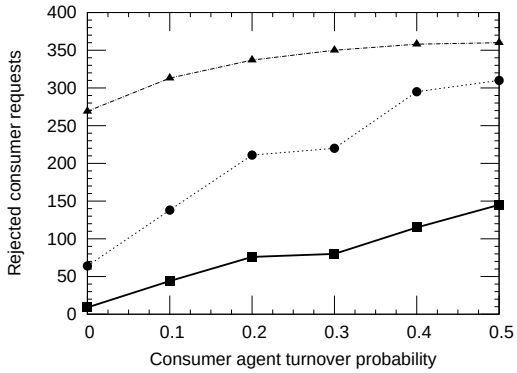

FIGURE 6.2: Number of rejected allocative requests as a function of increasing consumer agent turnover probability for three system model configurations: AF model (line with rectangles), NF model (line with circles) and FF model (line with triangles).

6.3 Consumer Agent Turnover

Figure 6.1 illustrates the mean system throughput for three model configurations, AF (rectangles), FF (triangles) and NF (circles), for increasing resource allocation dynamism reflected by the probability of consumer agent turnover . Here, the horizontal

[4] The duration of provider query (T_q) and execution (T_e) is dependent on the current provider utilisation where it takes more time for a heavily utilised provider to respond to the consumer query. The mechanism controlling T_q response time is described in Section 5.3.2.

dashed line indicates the hypothetical optimal allocative efficiency in conditions in which the system is closed and agents are able to resolve all arising resource access conflicts. As the results suggest, the best performance is achieved by the AF model in which consumers share their local information with other consumers sharing the same service type interests. For this configuration, as Figure 6.2 shows, the system is capable of suppressing almost all rejected resource allocation queries for conditions when consumer agent turnover is set to zero. Considering the NF and FF models, even for the initial closed system conditions (when consumer agent turnover probability is equal zero), there exists an observable amount of rejected consumer requests suggesting more resource access conflicts and thus less efficient load-balancing on the system level. This in turn impacts on system performance for the NF and FF models that, as illustrated in Figure 6.1, is worse than the one achieved by the AF model.

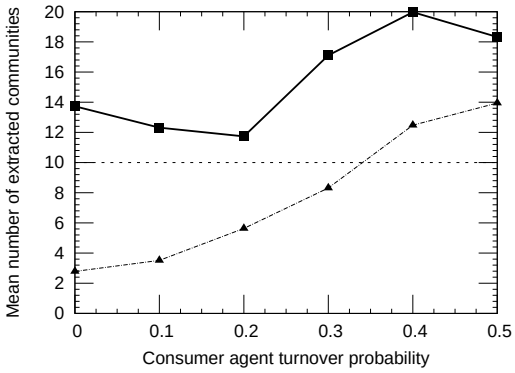

FIGURE 6.3: Mean number of extracted communities as a function of increasing consumer agent turnover probability for two system model configurations: AF model (line with rectangles) and FF model (line with triangles).

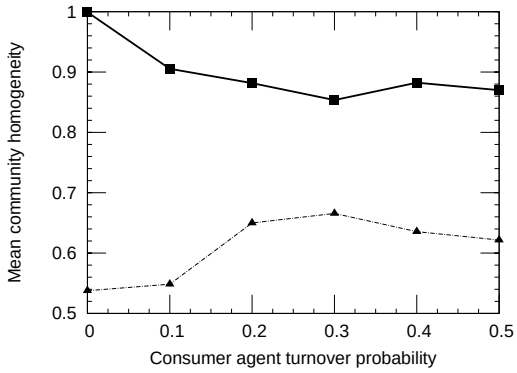

FIGURE 6.4: Mean community homogeneity as a function of increasing consumer agent turnover probability for two system model configurations: AF model (line with rectangles) and FF model (line with triangles).

In conditions in whihc the consumer agent turnover probability is greater than zero, and thus new consumer agents enter the system, the performance of all three models is compromised due to increasing system dynamics. However, in every consumer agent turnover probability setting, the best performance and the lowest amount of rejected consumer requests is achieved by the AF model.

Recall our hypothesis stated at the end of the previous section (Section 6.2.1) that the best system performance is achieved in conditions when agents self-organise into communities within which information is shared among like-minded (with respect to service type interests) agents. Given the results reflecting system throughput and thus efficiency in achieving load-balancing, we observed that two model configurations (AF and FF) that relied on information gossiping achieved orthogonal with respect to each other performances. Whereas the AF model achieved the best performance of all three considered model configurations, the FF model exhibited the worst, even when compared to the

model configuration in which agents did not share their local knowledge (NF). Why the AF model configuration outperformed not only the NF but, more interestingly, the FF model that also employed information gossiping?

6.4 Consumer Agent Communities

To understand why the AF model performance scales much better for the increasing dynamism of resource allocation let us analyse the underlying structure of agent interactions and thus the manner in which locally communicated information influences the consumer agent resource selection process. For this purpose, a network analysis was performed for AF and FF models [5].

The topologies that represent consumer agent interactions showing which consumers shared their local provider evaluations are illustrated for the AF model in the following two figures, Figure 6.5 and Figure 6.7, each showing the results for different consumer turnover probability configuration. The interaction topology for the FF model is shown in Figure 6.6. For both model configurations the networks were constructed by aggregating the consumer interactions that occurred within a 10 seconds period starting at the $600th$ simulation second.

Figure 6.3 presents a distinct number of consumer agent communities that can be viewed as collectives of agents that interact and reveal their personal provider evaluations more frequently among themselves than between agents that are not part of the community. Such communities are not pre-imposed (eg. configured at system design time) but arise spontaneously as a result of local agent learning mechanisms and information exchange. As shown, the number of such unique communities extracted from the AF model (rectangles) resides above a horizontal dotted line that defines the minimal possible number of distinct communities for a properly configured system. For the presented results, this number is equal to 10 as there are ten different subsets of consumers, with each group interested in allocating different service type.

Considering the composition of extracted communities with regard to the similarity of service type interests of the consumer agents, Figure 6.4 shows that communities identified within the AF model (rectangles) exhibit high homogeneity (kept between 0.85 and 1) even for the increasing consumer agent turnover conditions. This means that the majority of members forming each community are agents interested in allocation of the same service type. The interaction network formed by the AF model consumer agents operating in conditions in which consumer turnover probability is zero is presented in Figure 6.5. Here, it is shown that consumers were able to organise into a sufficient number of distinct and homogeneous collectives of interacting peers.

[5] There is no network analysis for the NF model as in this model configuration consumer agents do not communicate any information between each other.

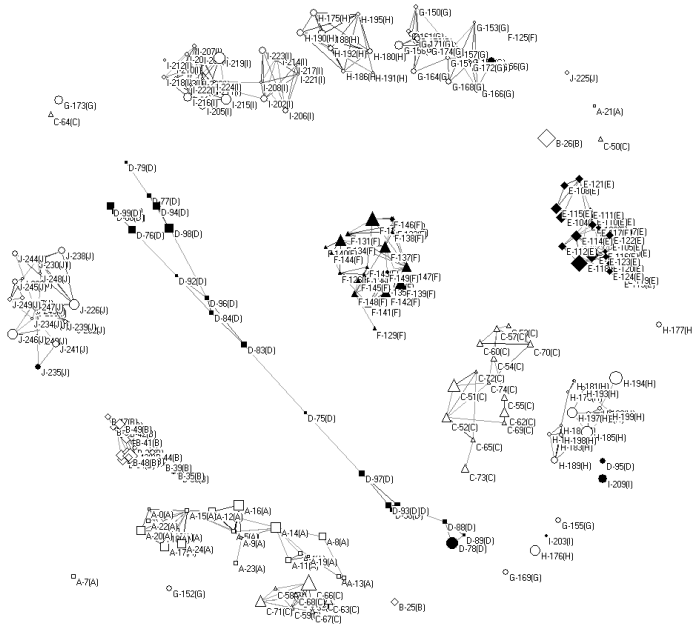

FIGURE 6.5: Correctly organised consumer agent communities extracted from AF model for conditions where consumer turnover probability is equal zero. Edges between nodes represent information exchanges between consumer agents where node shapes correspond to specific (denoted by node label) service type the consumer is required to allocate. The size of each node indicates the amount of resource capacity that is currently demanded by the task.

The same cannot be said about the homogeneity of communities extracted from the FF model (circles in Figure 6.4). Here, the mean community homogeneity varies between 0.53 and 0.66, suggesting that communities are formed out of a mixture of consumers interested in the allocation of different service types. Furthermore, as Figure 6.3 shows, the number of extracted communities for the FF model for the most of consumer agent turnover probability settings ($0.0 - 0.3$) remains below the minimal required level, meaning that consumers are unable to organise into distinct and highly homogeneous communities. This is confirmed by the consumer interaction network presented in Figure 6.6 that illustrates the FF model consumer agent interactions in conditions where consumer turnover probability is zero. As shown here, not only were consumers unable to establish a sufficient number of distinct communities (corresponding to the number of uniquely requested service types, equal to 10 in this model configuration) but also resulted in an organisation characterised by low homogeneity due to interactions between consumers interested in the allocation of different service types.

Finally, the impact of increasing system openness and thus resource allocation dynamism on the proportion of consumer agents forming communities (at least of size two members) is illustrated in Figure 6.8. Here it is shown that for both the AF and FF models the fraction of system agents capable of forming and sustaining communities becomes smaller as the system dynamics increases. Several conditions influence such gradual decay of structures formed by agents. Firstly, the information flow sustaining communi-

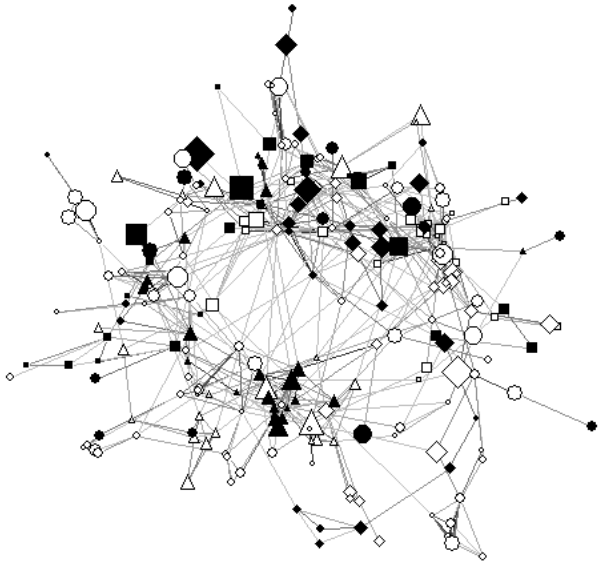

FIGURE 6.6: Disorganised consumer agent communities extracted from FF model for conditions where consumer turnover probability is equal zero. Edges between nodes represent information exchanges between consumer agents where node shapes correspond to specific (denoted by node label) service type the consumer is required to allocate. The size of each node indicates the amount of resource capacity that is currently demanded by the task.

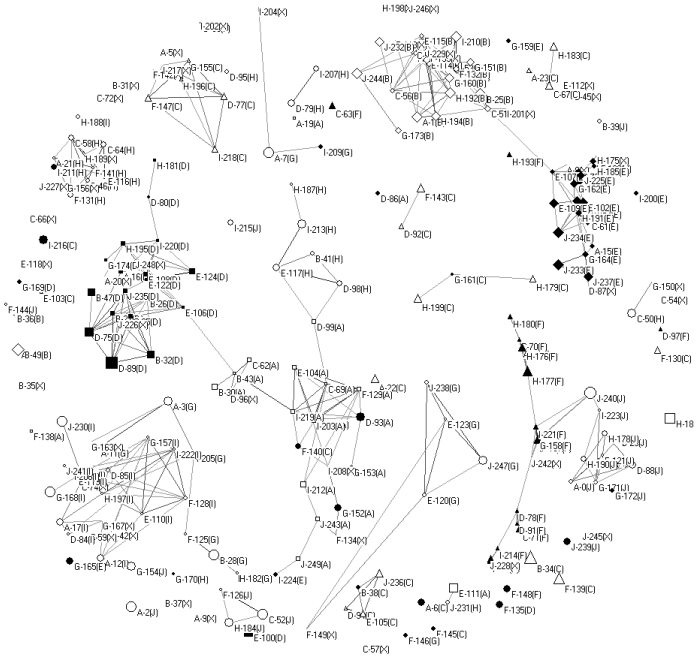

FIGURE 6.7: Correctly organised consumer agent communities extracted from AF model for conditions where consumer turnover probability is equal 0.1. Edges between nodes represent information exchanges between consumer agents where node shapes correspond to specific (denoted by node label) service type the consumer is required to allocate. The size of each node indicates the amount of resource capacity that is currently demanded by the task.

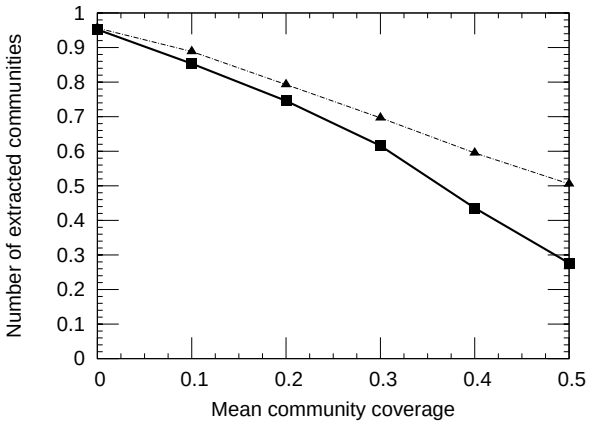

FIGURE 6.8: Mean community coverage as a function of increasing consumer agent turnover probability for two system model configurations: AF model (line with rectangles) and FF model (line with triangles).

ties in the AF model is regulated through the stress indicator perceived individually by consumer agents. As the allocation dynamics increases due to the influx of new agents and consumers become confused about which providers to select, the stress individually perceived by agents reaches a high enough limit to block the information flow between peers. Eventually, large and strong communities proliferate into smaller groups and consumers forming them tend to establish dynamic and transient communities, mostly for the duration of a single allocation. These communities are not identified by our community extraction mechanism as they decay very quickly and therefore overall system community coverage decreases. Figure 6.7 provides a consumer interactions network formed in conditions where consumer agent turnover is equal to 0.1. When compared to the network topology captured in more stable conditions (for the same AF model), where there is no new consumer agents influx (shown in Figure 6.5), it is visible that large communities present in more stable allocation conditions proliferate into smaller communities that are more resilient to instability in more dynamic conditions.

Another factor directly influencing the decay of communities is the influx of new consumer agents. These agents are introduced whenever an existing agent is removed from the system and are initially given no information about system state. Both the removal of consumers with already established local knowledge and the introduction of new ones with no knowledge at all causes a partial loss of information from the system. Although this loss eventually becomes recuperated through the information exchange during the process of which newly introduced consumers acquire information from their peers, the frequent introduction of new agents (and the removal of the existing ones) may cut the communication of information between groups of agents forming the same community and thus disassemble it for a long enough period of time to prevent its re-formation to original state. This is particularly visible within the FF model where agents do not limit the information flow (do not react to stress as within the AF model) but still the overall community coverage drops as the agent turnover increases.

6.5 Impact of Communities on Individual Performance

Up to this point we focused only on the analysis and comparison of two model configurations that employed information exchange (AF and FF) and for this purpose showed properties of networks arising as a result of local information exchanges between consumer agents comprising these models. But what relevance do these information flow topologies have on the behaviour of individual agents and how different it is from the model configuration (NF) in which agents do not employ information exchange?

Consider Figure 6.9 where an illustration of provider evaluation scores maintained by consumer agents within their local registries is shown. These were obtained from the experimental runs discussed in Section 6.3 in which consumer agent turnover probability is equal to 0.0. For the purpose of presentation, each consumer agent registry was organised in descending provider evaluation score order and the obtained values were averaged over the population of all agents. Given this, Figure 6.9 shows the evaluation scores for the first 20 provider agents for three experimental model configurations: AF model (rectangles), FF model (circles) and NF model (triangles).

Based on these results we can see that consumers within the NF model establish a preference and high evaluation score only to a single provider agent as a result of their own personal allocative experience. Although this allows them to consider such a provider in the first allocative request, the unavailability of this agent causes consumers to seek other resources without any additional preference and thus randomly [6] which is inefficient and generates a large number of resource access conflicts observed as rejected queries and poor system performance. Furthermore, the inability of consumers to share their information increases the probability that newly introduced agents that are not satisfied by the provider they are co-located with, will begin a stochastic search for other available providers.

How different from this is the memory state of consumer agents from the AF model? Consider the line with rectangles in Figure 6.9 that illustrates evaluation scores maintained by consumer agents comprising the AF model. Here we can observe that, on average, each consumer has established an attraction towards more than a single provider. Furthermore, the number of valued providers (around six) is a close match to the optimal number of providers that should be of interest for consumers (six) as there are 10 distinct provider and consumer sub-populations, each demanding and offering unique service types within a system comprising 60 provider agents.

Given the NF model consumer evaluation scores characteristics (line with circles in the figure), it is clear that the extended awareness of providers for AF model consumers agents, reflected by more than a single evaluation score within consumer memory, is

[6] As explained in Chapter 5 (Section 5.3.1), consumers rely on roulette wheel selection during the provider selection process.

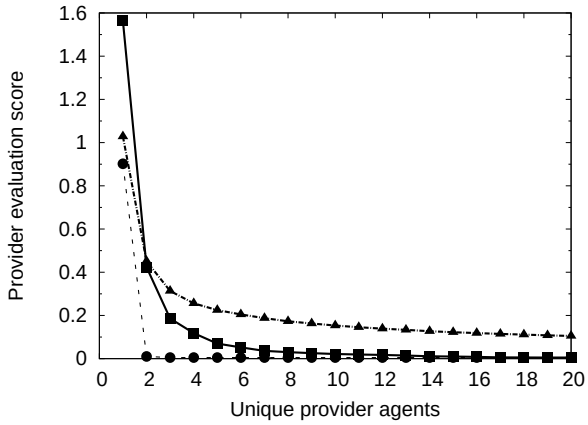

FIGURE 6.9: Provider evaluation scores kept within local registries of consumer agents from: AF model (line with rectangles), FF model (line with triangles) and NF model (line with circles).

provided through the information exchange between agents. As long as this information is being communicated through the collective actions of agents sharing their individual experiences, each community member will be capable of conducting more informed and thus more efficient decisions. In relation to the load-balancing problem, AF model consumers are able to quickly resolve allocative conflicts (if they arise) by conducting the informed selection of another, highly evaluated provider. As the performance results for the AF model show, this allows agents to achieve greater efficiency. Another strength of communities can be observed during the influx of new consumer agents to the system that have not established any preference for provider selection. Within the AF model, once such a new agent becomes co-located with a provider agent, it will be supplied with the community level information from other consumer agents that utilise this provider for resource allocation. This, in turn, will increase its system resources state awareness, avoid random provider search (if the co-located provider cannot satisfy its request) and thus facilitate efficient load-balancing in an open system environment.

The information gossiping mechanism within the FF model was purposefully designed to let consumer agents communicate information to agents that reside outside their communities. By doing so, consumers interested in the allocation of a specific service type (eg. X) encourage consumers that are after other service types (eg. Y) to engage with providers offering the former type of a service. This causes Y consumers to interact with X providers and thus achieve a large number of rejected requests. Within a system where 10 different service types are demanded, consumers sharing their local evaluations across different communities become continually frustrated over which providers to select. This behaviour is illustrated in Figure 6.9 by a line with triangles showing the evaluation scores maintained by consumer agents in the FF model. In contrast to the AF model (lines with rectangles), consumers within the FF model are unable to limit their selection to a valid subset of providers offering service types they demand and thus experience a much higher number of rejected requests due to inefficient load-balancing.

6.6 Conclusions and Summary

In this chapter we analysed the performance of our decentralised autonomic system model tasked to achieve efficient load-balancing in dynamic resource allocation conditions. To do so, we hypothesised that the global system efficiency is dependent on the ability of individual system agents to organise into communities. To examine this hypothesis, we explored the performance of three different model configurations (AF, NF, and FF) that were distinguishable by the usage (or lack) of different local-decision making mechanisms responsible for local information communication.

The results reported in Section 6.3 suggest that the best load-balancing efficiency is achieved by the AF model. It is also the same model within which highly homogeneous and appropriately sized communities have been identified. Considering the poor performance of the FF model accompanied by disorganised (low homogeneity) and insufficiently sized communities or the inefficiency of the NF model where no communities exist at all, the question may be stated: what is the role of agent communities in achieving high system efficiency and load-balancing in particular?

Given the nature of resource allocation within a decentralised system in which access to a global information repository is absent, it is clear that the increase in system efficiency can be achieved when localised agents conduct informed decisions when selecting available resources. Having no central information repositories, the only possible way of achieving this is to learn from others. Although no agent possesses sufficient knowledge to let it facilitate optimal allocation, each individual agent maintains local bits of information about performance of a small group (or even a single) provider(s) that it engaged and employed for allocation at a recent time. By revealing this information to others and encouraging other agents to do the same, individual agents may increase their awareness of the availability of system resources. This enables them, in turn, to make informed decisions that increase their personal as well as the overall system efficiency. This is precisely what we have observed in the AF model where the identified communities represented consumer agent collectives coherently sharing their individual allocative experiences.

Considering the difference in performance between the AF and FF models that both employ information exchange, we observed that only information exchange mechanisms from the AF model allow the system to achieve high efficiency. This implies that not only lack of information flow (NF model) but also inefficient information communication (FF model) may destabilise resource allocation within the decentralised system.

Such insights imply two important things. Firstly, the provision of expected system functionality (load-balancing in this context) in a bottom-up manner requires local decision-making mechanisms that allow specific functional structures such as observed communities to arise out of local interactions and information exchanges between system agents.

And secondly, to sustain a desired system organisation in a form of communities within the dynamic system operating conditions, agents need to employ local mechanisms that regulate the flow of information between their peers.

In the next chapter we will continue our elaboration on the importance of system organisation into communities and their bottom-up organisation and regulation through local information exchange within more complicated system settings. In particular, we will explore the problem of adaptive service provisioning where provider agents are allowed to offer more than a single service type and thus may decide to reconfigure their service provision at run-time and in response to locally perceived demand. Consequently, global system efficiency depends not only on the efficient load-balancing but also on the appropriate adaptation and configuration of provider agents. As in this chapter, the experiments are carried over a number of dynamic resource allocation scenarios, investigating system response and adaptation in changing conditions.

Chapter 7

Adaptive Service Provisioning

7.1 Introduction

In the the previous chapter we focused on the load-balancing problem within autonomic systems. For this purpose a multi-agent system model was presented and local decision-making mechanisms introduced that facilitated efficient distribution of resource requests to resource providers available within the system. Throughout the experiments carried out for different model configurations and dynamic resource allocation conditions we observed that the efficiency of the agents depends on their ability to organise into communities responsible for up-to-date information flow across community members.

In this chapter we continue our analysis of system organisation into communities and its relevance in facilitating more advanced autonomic system functionality that is adaptive service provisioning. To do so, we relax the assumption held in the previous chapter that the proportion of consumer agents requesting various service types remains constant over the simulation time. As a result, provider agents no longer offer a single and pre-configured service type but are allowed to adjust to changing demand conditions at run-time through service type reconfiguration.

In the past, the above described process of adjusting the system to the needs of its users has been performed in the off-line mode by system administrators. Such infrastructure managers were responsible for the reconfiguration of individual server nodes such that the overall system resources supply matched some user demand model, often captured through the analysis of historical demand data the system experienced in the past [58]. Whilst this approach was sufficient for small scale deployments where change in service type demand was highly deterministic and therefore allowed for off-line reconfiguration that closely matched to the on-line demand conditions, existing IT systems no longer guarantee such stable functioning conditions. As a consequence, rather than relying on an off-line server reconfiguration, modern autonomic systems stress the relevance of

on-line adaptation of offered by the system services to the currently perceived demand conditions [59].

To address this adaptive service provisioning problem in a manner that minimises human administrator involvement, much of the current research in the area of autonomic computing focuses on techniques preserving the inter-operation of existing IT systems' software modules, often encapsulating their functions in terms of autonomic managers. As a result, human decision-making becomes gradually replaced by automated responses conducted by autonomic managers that follow certain rules and policies whilst conducting their decisions. In achieving this, techniques such as reinforcement learning [61, 119], optimal control theory [125], and maximisation of expected utility [59] are then exploited in order to balance power-performance tradeoffs, i.e., to achieve efficient allocations of requested jobs at the same time as optimising the power consumption of unused servers.

Two kinds of control architecture tend to be employed: *centralised* and *distributed*; both of which are illustrated schematically in Figure 7.1. Centralised schemes rely on a central executive to co-allocate services, schedule and plan system behaviour, etc. In contrast, distributed control schemes employ distributed protocols and focus on the design of intelligent parallel algorithms for coordinating the behaviour of agents.

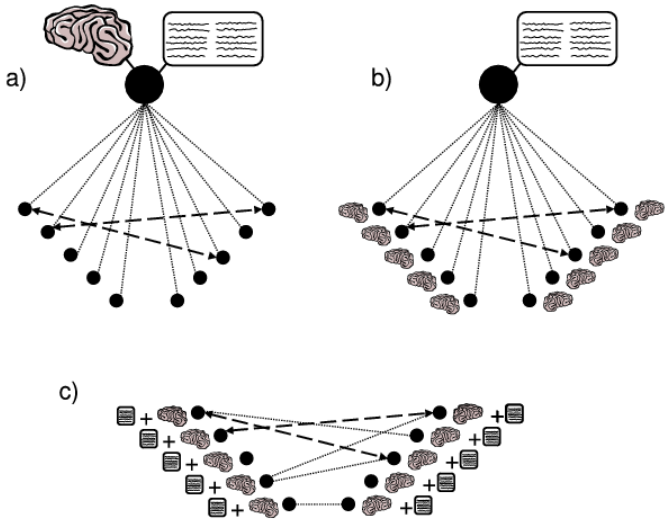

FIGURE 7.1: Three classes of control organisation: a) centralised control, b) distributed control reliant on consensual, up-to-date, global information, and c) fully decentralised control. Service providers and consumers are represented by small circles, central executives or central repositories by large circles. Agents may store information (lozenges) and/or execute co-allocation algorithms (brains). Dotted lines connote information exchange, whereas dashed lines connote the pairing of services and resources achieved by the co-allocation process.

However, whilst these mechanisms are somewhat decentralised, they generally assume that up-to-date information is freely available. Thus, in order to converge on an optimal solution, such schemes require that each agent possesses a substantial amount of global

system information, resulting in the need to perform a large number of interactions in order to maintain awareness of peer goals, actions, etc[1]. Furthermore, with the increasing scale and dynamism involved with operation of modern IT systems, it become apparent that collecting and processing up-to-date information in large scale deployments can become a significant problem due to the time-delays associated with obtaining large amounts of distributed information [60, 26, 41]. As a consequence, systems relying on either centralised or distributed control schemes (Figures 7.1a and 7.1b) are often vulnerable to increasing system scale and/or dynamism. Although [125] proposed a control scheme that is scaleable, Wang *et al.* relied on a 'divide and conquer' approach, where allocation and power management problems are easily decomposable into subproblems that can be solved independently by individual agents. However, as the authors note, this is possible only when the global problem could be decomposed into independent sub-problems with no mutual constraints, thus ensuring that no coordination was required between non-interacting agents. This assumption is unrealistic in service oriented infrastructure deployments, where many workflow processes may compete for the use of multiple services, distributed over a number of servers that thus become interdependent.

In addition to this, many of the systems described above consider small-scale deployments involving small numbers of agents (i.e., between two [61] and sixteen agents [125]), whereas realistic, large-scale, autonomic infrastructures could comprise of hundreds, thousands or possibly millions of such autonomous *loci* of control. Such large scale deployments not only would require control mechanisms that allow them to relax into efficient configuration, but also to achieve it in the robust and timely manner.

To facilitate such adaptive response within the context of adaptive service provisioning we employ a multi-agent system model that we described in Chapter 5. Here, the model scaleability is facilitated by providing decentralised architecture in which agents conduct their decisions based on locally available information and thus do not require coordination protocols or access to global system state. The sufficient level of robustness and adaptation, on the other hand, stems from the increased autonomy and independence of individual provider agents that are allowed to make their own decisions about which service types to offer, based only on their locally perceived information about the system demand.

Given this model, we verify if adaptive service provisioning functionality can arise in a bottom-up manner through local interactions and information exchanges between resource allocating agents. Similar to the work carried in the previous chapter, also in here, we stress the relevance of agent organisation into collectives cooperatively sharing their local information and for this reason continue the evaluation of hypothesis stated in the previous chapter that the efficiency of the system depends on the ability of agents

[1]It is important to recognise the difference between this scheme and a fully decentralised model (Figure 7.1c) in which every agent must make its own decisions based on locally available information that may not be available to its peers.

to organise into communities.

To this end, in Section 7.2 we describe adaptive service provisioning issues raised by autonomic computing. Our experimental model evaluation addressing this problem is presented in the three following Sections 7.3, 7.4 and 7.5. Finally, the chapter ends with discussion in Section 7.6.

7.2 Adaptive Service Provisioning Problem

In the previous section we suggested that one of the possible ways of addressing adaptive service provisioning can be realised through the application of a decentralised multi-agent system model in which individual provider agents are allowed to reconfigure and adjust the offered by them service type based on the locally perceived consumer demand.

Whilst this approach offers flexibility and scaleability that is difficult to achieve within centralised or distributed control schemes, it also introduces additional problem of exerting stable and efficient configuration within a system of autonomously interacting agents that operate only on locally available information.

One way of addressing the problem of controlling group of autonomous agents is to endow them with coordination mechanisms that provide them with sufficient information about the global system state such that coherent behaviour emerges out of their individual but rational decisions. However, Arthur [2] demonstrated in the El Farol Bar problem that instability can emerge within dynamic environments when independent rational agents all have access to the same global information. In the game-theoretic example illustrated by Arthur, each agent wishes to visit the bar if and only if less than 60% of the agent population also wish to visit the bar. Consider a rational agent, A, that decides to visit the bar. A might reasonably assume that every other agent will reach the same decision and choose to visit the bar, in which case A must change her mind and choose not to visit the bar. But she must also reason that every other rational agent would also change their mind in this circumstance. The quandary rests on two symmetries: (i) every agent employs the same decision-making mechanisms; and (ii) every agent reasons on the basis of the same information.

Both these symmetries are typically present in the models of decentralised resource allocation proposed in a number of problem domains, including coalition formation [109], group problem solving [114] and teamwork [97]. A common property of such models is their reliance on distributed protocols and focus on design of intelligent algorithms coordinating the behaviour of agents. However, these mechanisms, whilst decentralised, generally assume up-to-date shared information, and thus in order to converge to an optimal solution, they require a substantial amount of global system information followed by a large number of interactions among system elements to maintain awareness of peer

goals, actions, etc. As a consequence, they are often vulnerable to increasing system scale and/or dynamism [19].

Whereas the above, AI inspired control approaches turn out to be prone to increasing system scale and dynamism, it has been observed that natural decentralised systems, which operation also involves efficient and adaptive management of resources, overcome these vulnerabilities relying only on local interactions and adaptations between system elements [7, 105].

One of the examples of such systems that was extensively studied and well understood are insect societies that show their remarkable abilities at achieving division of labour in a fully decentralised manner [91]. Here, the survival of the whole colony depends on the ability of its constituents to effectively divide their labour, such that the system survival functions are maintained. To do so, a potentially homogeneous population of ants, each capable of handling the same range of tasks, dynamically differentiates itself into a number of distinct but organised collectives or *castes* [120]. Each such collective specialises in carrying out a specific task, such as food foraging, nest building, brood feeding, nest defence, etc. The survival of the colony thus depends on both the efficient handling of each system task and the adaptive division of resources (ants) into a number of such collectives responsible for these different tasks. One of the most striking aspects of such a regulatory response is its *plasticity*, a property achieved through the workers' behavioral flexibility: the ratios of workers performing the different tasks that maintain the colony's viability and reproductive success can vary (i.e., workers switch tasks) in response to internal perturbations or external challenges [7].

Understanding how this run-time flexibility within biological systems is implemented at the level of individual system elements which certainly do not possess any global representation of the colony's needs has been addressed to some extent [120, 7, 73]. According to these studies the self-regulatory colony properties appear to stem from simple threshold-based behaviours where the specialisation of system elements to handle particular tasks arises as a result of reinforcement processes [120].

Despite the advances made in understanding such *stimuli-response* mechanisms, one of the key issues involved in engineering models that exploit them for achieving adaptive resource management remains the difficulty of influencing the 'right' interactions and avoiding those that may frustrate and destabilise the system [91, 11]. Recent studies focusing on this problem suggest that the processes responsible for this are local decision-making mechanisms that perceive, process and propagate locally available information amongst system elements [85, 93, 7, 99].

Results obtained from the analysis of the effects such local information dissemination has on the ability of the system to self-organise point to the relevance of local decision-making mechanisms that regulate information flow and prevent situations in which there exists too little or too much communication between agents [24, 29, 11]. In the former case,

where there is little or no flow of information between the system elements, each must act on the basis of extremely limited information and may tend to make poor decisions that decrease overall system performance. In the latter case, where information can flow *too* freely, or may be globally available, system behavior may become extremely dynamic and unstable (as in the case of Athur's El Farol Bar problem [2]), thus risking the possibility of cached information becoming stale, inappropriate or irrelevant, and consequently destabilising the overall system behaviour. In between these two extremes, there may exist a regime in which information flow amongst system elements may bring about stable and adaptive system response.

Such organised and constrained flow of information is critically relevant in our decentralised model that we employ for achieving adaptive service provisioning. Recall the problem of resource competition that we addressed through agent communities in the last chapter and let us now consider lethal effects it may have on the model configuration discussed in this chapter that we outline below.

As stated in the last chapter, consumer agents may compete for resources either if they possess no information about resources availability and thus choose the random ones or if they share their local provider evaluations among consumers that are interested in the allocation of different service types than they require. Under these circumstances resource competition contributed towards rejected resource queries that negatively affected allocation time and, when exceeded task allocation deadline, caused the current allocation to be failed. However, within the model considered in this chapter, in which providers may decide to reconfigure their service provision at run-time [2], resource competition introduces additional effect that is lethal to the system stability. Here, consumers that mistakenly reveal their local provider evaluation scores to agents interested in the allocation of other service types may now cause the latter, attracted through this communicated information, to pursue these attractive providers and encourage them to reconfigure and offer different service type. This will result in the local instability as the consumer agents that were previously employing this provider will no longer be able to utilise it and thus will be required to identify other one that is available and correctly configured. This, within a model where consumers are unable to organise their interactions and compete for resources may lead to global resource market instability and chaotic system response, where provider agents continually reconfigure and thus are unable to offer any resources.

Hypothesis 2:

We hypothesise that only model configurations in which agents are able to organise into communities are able to secure system resources such that adaptive service provisioning

[2]For this purpose, provider agents rely on *stimulus-response* mechanisms (described in Chapter 5) that are inspired by the division of labour within insect societies.

$\|Capability\alpha_p\|$	Set of service types ($Capability_{\alpha_p}$)	Number of Consumers	Number of Providers
8	$\{A, B, C, D, E, F, G, H\}$	250	60

TABLE 7.1: The general model configuration for adaptive service provisioning scenario where demand changes are triggered through consumer agent turnover mechanism.

emerges within the population of provider agents and thus the system achieves high level of efficiency. This is achieved by agent communities that impose a constraint on the flow of valuable information about providers availability, such that it is disseminated only across community members and thus enhances their individual decision-making. As a result of such regulated information flow, the risk of propagating too much information to other system elements and thus lethal effects of resource competition are limited.

In what follows we will continue our hypothesis examination in the same manner as in the previous chapter. For this purpose, three model configurations (AF, FF and NF) will be employed and their efficiency at achieving adaptive service provisioning investigated in the range of dynamic conditions. As stated earlier in this section, the only difference between these models and the ones explored in the previous chapter is the ability of provider agents to reconfigure at run-time.

7.3 Adaptive Service Provisioning in Open System

We start our experimental evaluation of adaptive service provisioning by applying consumer agent turnover mechanism. For this, the resources market is set in the following manner. Initially all provider agents are not specialised at offering any service type but may configure themselves to offer one service type from the set of eight unique ones ($Capability\alpha_p = 8$). The eventual provider configuration is determined at run-time based on the locally perceived demand imposed by the population of consumer agents. To reflect a heterogeneous demand for different service types, the population of consumer agents is initially split into eight equal size groups. Each such consumer sub-population is interested in allocating a unique service type. Consequently, when the model is run, the optimal system configuration will be achieved if the resource market proliferates into eight subsets of provider agents, each such subset offering a unique service type in accordance to the demand imposed by consumers.

In order to introduce dynamism in service type demand over the simulation time, the demand does not remain constant but experiences perturbations. This is achieved through the consumer turnover mechanism that introduces demand variation in the following manner. Each time a new consumer agent is introduced to the system, the service type it will demand over its life-time is randomly selected from the set of service types ($Capability\alpha_p$) the system is capable of providing. By increasing the probability of

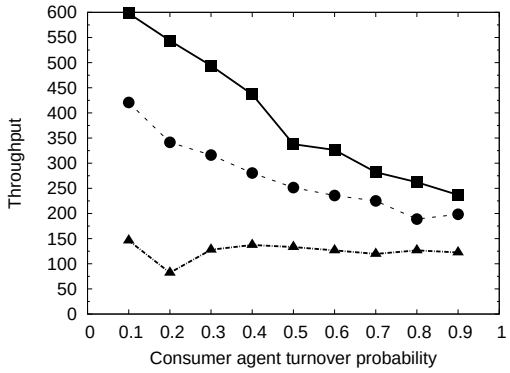

FIGURE 7.2: Mean system throughput as a function of increasing consumer agent turnover. Three model configurations are presented: AF (line with rectangles), NF (line with circles) and FF (line with triangles).

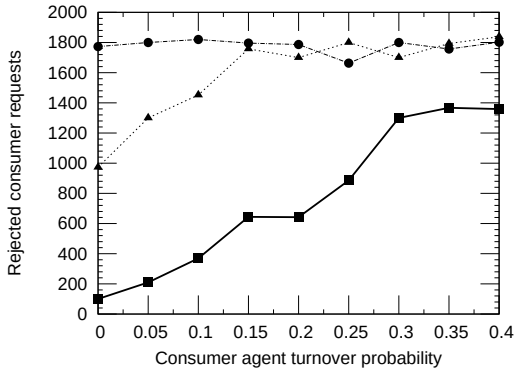

FIGURE 7.3: Mean number of rejected consumer allocation queries for three model configurations: AF (line with rectangles), NF (line with circles) and FF (line with triangles).

consumer agent turnover, regulated by the χ parameter as in Section 5.4.1, the system experiences more perturbations as new consumers are introduced and thus the resource market is required to adaptively respond to dynamically occurring demand changes.

For all experiments, the experiment duration is set to last 3600 simulation seconds and the model configuration is presented in Table 7.1.

7.3.1 Resource Market Adaptation During Consumer Turnover

Figure 7.2 illustrates mean system throughput for three different models in which there exists: adaptive information flow (AF), no information flow (NF) and unconstrained information flow (FF) between system peers. The consumer turnover probability in all experiments remains fixed and is set to the χ parameter value depicted on the X axis. The horizontal dotted line at 695 on the Y axis denotes the best achievable performance for the model in which there is no influx of new agents and where resource provision stabilises such that all tasks are allocated successfully.

The experimental results show that the worst performance is achieved by the FF model that is unable to cope not only with the increasing system dynamics but also with the configuration in which there exists no consumer turnover ($\chi = 0$). This poor performance follows from an unrestricted flow of information among agents. In this configuration, consumer agents communicate their local knowledge about providers' availability to randomly chosen peers and by doing this attract the others towards efficient providers. However, different consumers demand different service types and thus the resource market becomes frustrated and providers continually reconfigure to adjust to disorganised demand imposed by consumers that are confused which provider to select. In this configuration, even for $\chi = 0$, a large number of providers continually reconfigures thus locking

almost half of the system resources and reinforcing the competition for the remaining ones. Consequently, consumers experience more allocative failures than successful task allocations and thus the overall system throughput is close to zero. The inefficiency of the FF model is supported by the measure of rejected consumer queries illustrated in Figure 7.3, showing that the model with unconstrained information flow experiences high (near 1800) job rejects during each sampling period, irrespective of dynamics imposed by the consumer turnover mechanism.

An increase in performance can be observed for the NF model configuration. Here, for conditions in which the system is closed and thus there is no consumer turnover ($\chi = 0$) the system achieves high throughput (around 500 allocations). However, as soon as the turnover probability increases, and thus the system is stressed and pushed from stable allocation conditions, the performance of this model quickly degrades, reaching zero throughput at $\chi = 0.25$. This inefficiency is also reflected in Figure 7.3 showing that the number of rejected queries increases until it reaches 1800 rejected queries that characterise the disorganised and chaotic performance of FF model.

The best response and adaptation to increasing system dynamics is achieved by the AF model. In this configuration the system is capable of achieving close to the optimal (695) allocative throughput in conditions where there is no inflow of new consumer agents. In conditions where turnover is introduced and the probability of new agent arrival increases, the system is still capable of achieving high performance up to a point where $\chi = 0.3$. During this time a decrease in performance is observed but, as compared to the previous two models, it is a graceful degradation that prevents the system from moving into a chaotic and disorganised state. These observations are confirmed by the smooth and gradual increase of rejected allocation queries that is shown in Figure 7.3.

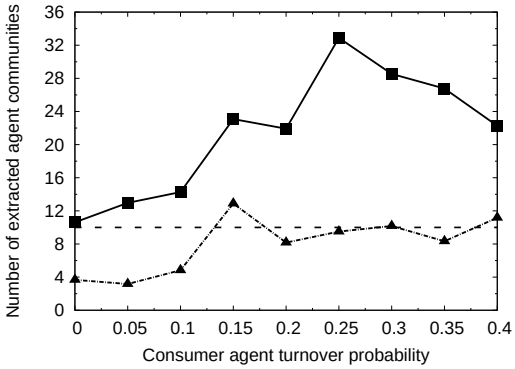

FIGURE 7.4: Mean number of extracted communities as a function of increasing consumer agent turnover probability for two system model configurations: AF model (line with rectangles) and FF model (line with triangles).

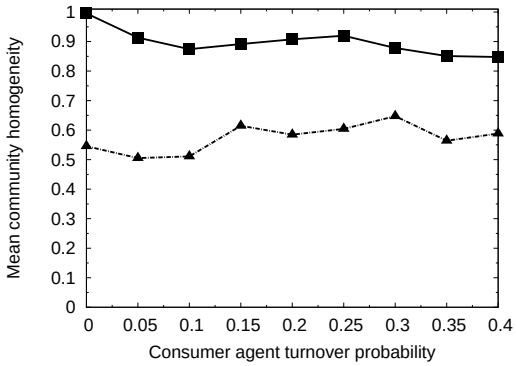

FIGURE 7.5: Mean community homogeneity as a function of increasing consumer agent turnover probability for two system model configurations: AF model (line with rectangles) and FF model (line with triangles).

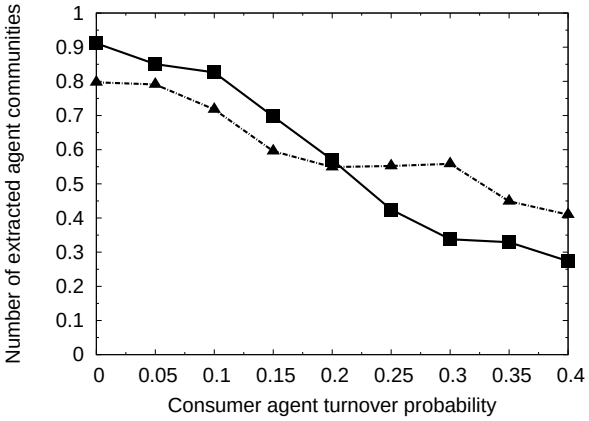

FIGURE 7.6: Mean community coverage as a function of increasing consumer agent turnover probability for two system model configurations: AF model (line with rectangles) and NF model (line with triangles).

7.3.2 Consumer Agent Communities

The regulatory response of the AF model to the increasing dynamics of the resource environment, is supported by the community analysis presented in Figures 7.4 and 7.5. Here, communities of information exchanging agents have been extracted and analysed for the AF and FF models that employ information exchange. The first figure shows that for a low enough consumer agent turnover probability ($\chi < 0.1$) consumer agents form close to the maximum in size (depicted by the dotted line in Figure 7.4) communities that are characterised by high homogeneity (as indicated by homogeneity measure illustrated in Figure 7.5). However, as the dynamics increases ($\chi > 0.1$) the system organises into smaller (but still homogeneous) groups of agents. Such smaller communities arise as a result of reaction of consumers to increasing stress they perceive. To prevent the resource market from moving into an unstable state, they limit the flow of information to a smaller subset of peers. Such smaller collectives turn out to be more resilient to the dynamism and preserve more stability as it becomes much easier for them to collectively maintain their current configuration through local information exchange.

The observed regulatory response for the AF model configuration stems from mechanisms regulating how much and to what peers the local information should be communicated. The community analysis for the FF model that lacks such regulatory mechanisms reveals that not only is the system unable to establish correct size communities, but also the homogeneity of these communities is low (below 0.6 for most of χ values).

Another important community property that changes in response to increasing system dynamism is community coverage provided in Figure 7.6 that illustrates the proportion of consumer agents that do not belong to any community during the community extraction

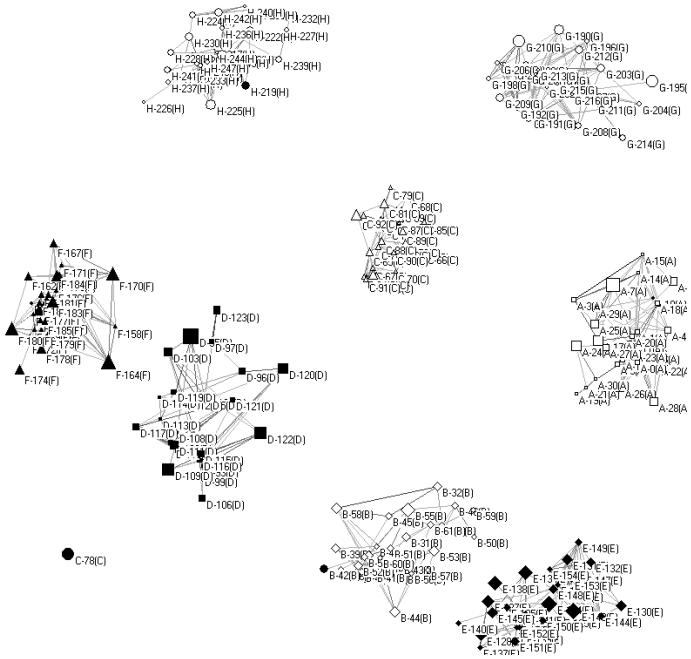

FIGURE 7.7: Correctly organised consumer agent communities extracted from AF model for conditions where consumer turnover probability is equal zero ($\chi = 0$). Edges between nodes represent information exchanges between consumer agents where node shapes correspond to specific (denoted by node label) service type the consumer is required to allocate. The size of each node indicates the amount of resource capacity that is currently demanded by the task.

process. The figure shows that with the increasing consumer agent turnover probability (χ) the fraction of non-community members grows for both the AF and FF models. This is partially due to the fact that there exists a fraction of new agents being continually introduced to the system that are new and thus do not form any community in their initial state and, partially, because certain consumers choose to 'leave' communities and act individually due to the high stress they experience (in the AF model configuration).

Example graphs showing communities for the AF and FF models are provided in Figure 7.7, Figure 7.8 and Figure 7.9.

7.4 Adaptive Service Provisioning with Discrete Environmental Change

In the previous section we made an implicit assumption that the demand imposed by consumers allocating eight unique service types and thus providers being capable of configuring themselves to offer the same number of service types ($|Capability\alpha_p| = 8$) remained unchanged over the simulation time. In these conditions, the dynamics and demand change was influenced only through consumer turnover mechanism.

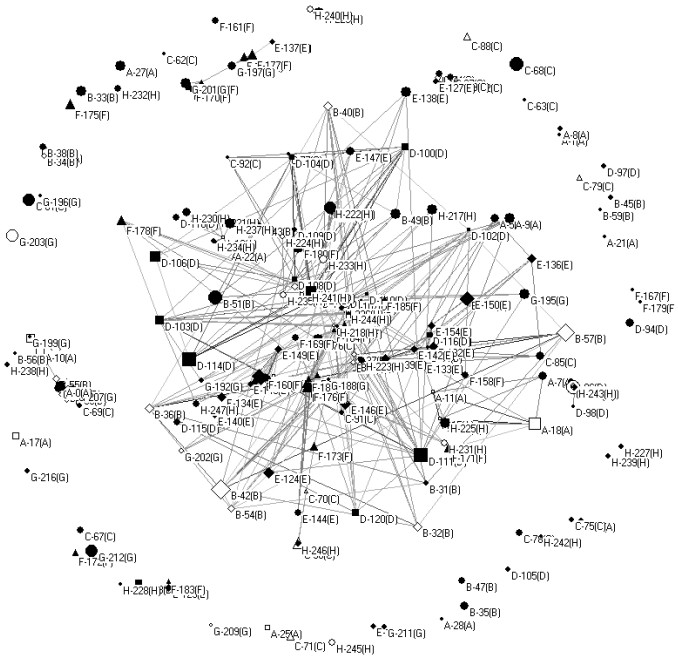

FIGURE 7.8: Disorganised consumer agent communities extracted from FF model for conditions where consumer turnover probability is equal zero ($\chi = 0$). Edges between nodes represent information exchanges between consumer agents where node shapes correspond to specific (denoted by node label) service type the consumer is required to allocate. The size of each node indicates the amount of resource capacity that is currently demanded by the task.

In this section we relax this assumption by introducing a discrete function responsible for service type demand changes during the experiment duration. To do so, we set the experiments such that in specific simulation time intervals (every 2000 simulation seconds) the demand for unique service types imposed by the population of consumer agents changes according to the model setup provided in Table 7.2. Such change causes the consumer population to be divided into sub-populations, each requiring a unique service type during task allocation. The number of such distinguishable consumer sub-population changes over the simulation time according to the function presented in Figure 7.10. Change in demand achieved in this manner triggers the resource market adaptation within the system as providers need to reconfigure and readjust to the new demand conditions.

Apart from this discrete demand change function we also allow for an influx of new consumer agents during the experiment that is regulated through the consumer agent turnover mechanism as in the previous section. Since now each simulation is divided into time intervals within which a different number of unique service types is demanded, the service type for each consumer arriving to the system during the turnover is randomly selected from the $|Capability\alpha_p|$ set that represents the current number of uniquely demanded services within the system. Given this, below we present our experimental

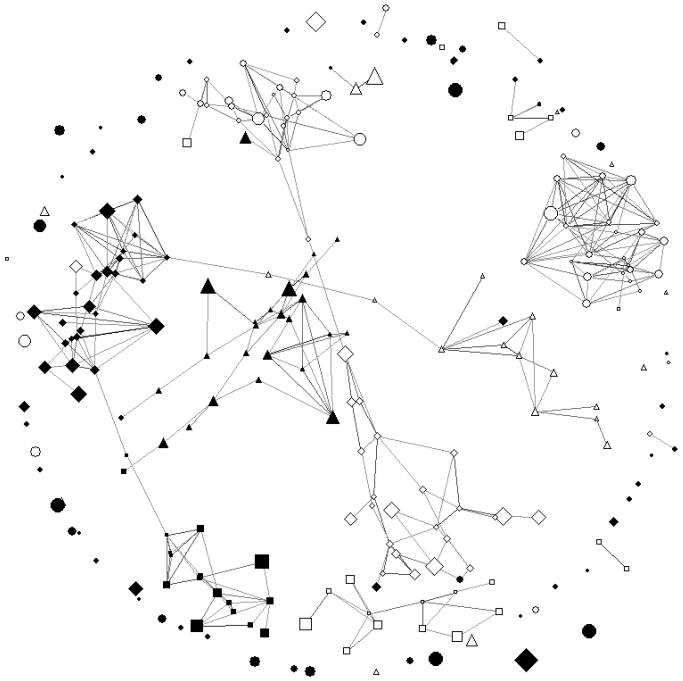

FIGURE 7.9: Correctly organised consumer agent communities extracted from AF model for conditions where consumer turnover probability is equal 0.15 ($\chi = 0.15$). Edges between nodes represent information exchanges between consumer agents where node shapes correspond to specific service type the consumer is required to allocate. The size of each node indicates the amount of resource capacity that is currently demanded by the task.

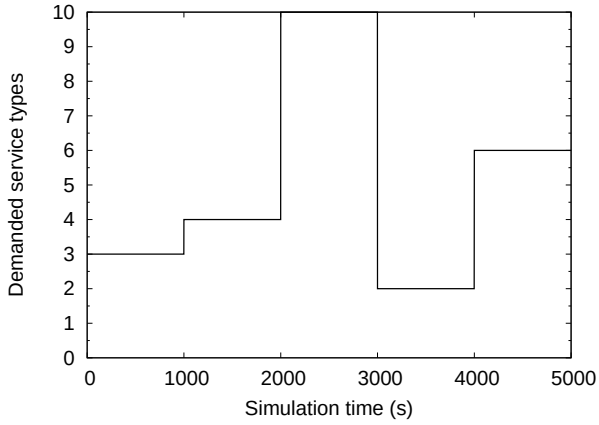

FIGURE 7.10: Figure illustrates step function according to which demand for a number of unique service types (represented on Y axis) undergoes rapid change at simulation periods indicated on X axis.

$\|Capability\alpha_p\|$	Set of service types ($Capability_{\alpha_p}$)	Number of Consumers	Number of Providers
4	$\{A, B, C, D\}$	250	60
8	$\{A, B, C, D, F, G, H, I\}$	250	60
6	$\{A, B, C, E, F, G\}$	250	60

TABLE 7.2: The change in the number of demanded service types ($|Capability\alpha_p|$) for subsequent steps in the step demand function.

evaluation of adaptive service provisioning.

7.4.1 Resource Market Adaptation in Discretely Changing Environment

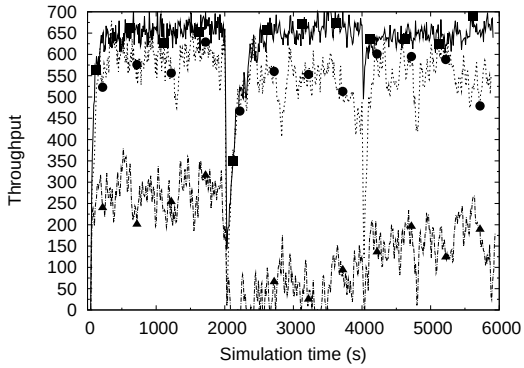

FIGURE 7.11: Mean system throughput as a function of simulation time for AF model (rectangles), NF model (circles) and FF model (triangles). For all models consumer turnover probability is set to zero ($\chi = 0$).

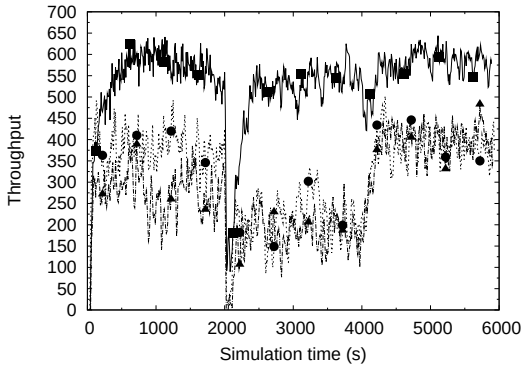

FIGURE 7.12: Mean system throughput as a function of simulation time for AF model (rectangles), NF model (circles) and FF model (triangles). For all models consumer turnover probability is set to 0.1 ($\chi = 0.1$).

The mean system throughput for the AF, NF and FF models in conditions when no consumer turnover takes place ($\chi = 0$) is illustrated in Figure 7.11, whereas Figure 7.4.1 shows the response of the three aforementioned models when the system is open and there is an inflow of new consumer agents controlled through a turnover probability equal to 0.1 ($\chi = 0.1$).

The best throughput in each of these configurations is achieved by the AF model. Despite the fact that in the Figure the system remains open and thus prone to demand deviations influenced by new consumer agent turnover, the system is still capable of reconfiguring and regain stability after each discrete demand reconfiguration period taking place every 2000 simulation seconds. Although consumer agent turnover affects the maximum system throughput (as compared to a closed system), the system is capable of reorganising resource provision and gaining maximum affordable efficiency. During the

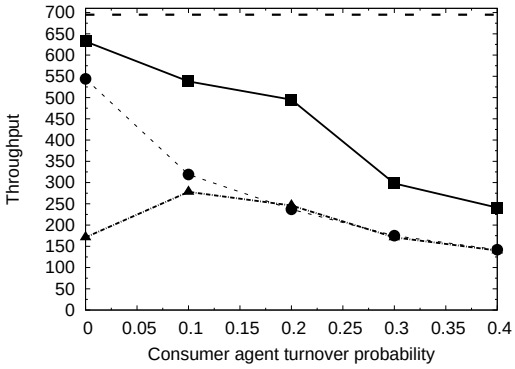

FIGURE 7.13: Mean system throughput as a function of increasing consumer agent turnover. Three model configurations are presented: AF (solid line with rectangles), NF (dotted line with circles) and FF (dashed line with triangles).

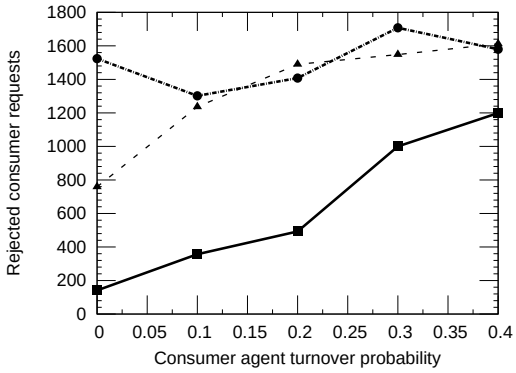

FIGURE 7.14: Mean number of rejected consumer allocation queries for three model configurations: AF (solid line with rectangles), NF (dotted line with circles) and FF (dashed line with triangles).

discrete and rapid demand changes occurring every 2000 simulation seconds, the system performance drops for a short while (reflected in both Figure 7.11 and Figure 7.4.1 as a peak in performance drop) and after this short period of disorganisation moves back to its stable efficiency.

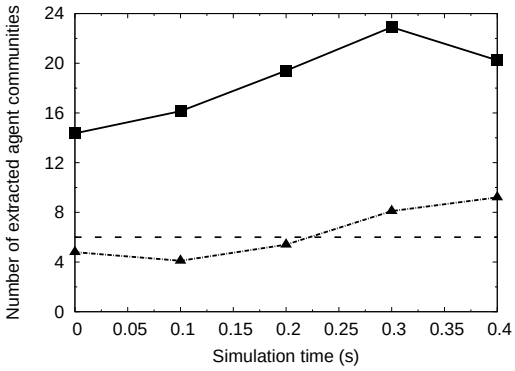

FIGURE 7.15: Mean number of extracted communities as a function of increasing consumer agent turnover probability for two system model configurations: AF model (line with rectangles) and FF model (line with triangles).

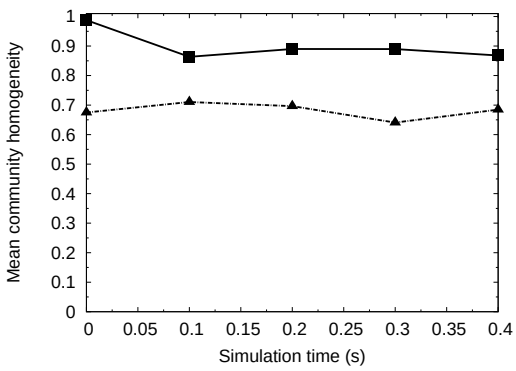

FIGURE 7.16: Mean community homogeneity as a function of increasing consumer agent turnover probability for two system model configurations: AF model (line with rectangles) and FF model (line with triangles).

Such adaptive behaviour does not arise in the NF and FF model configurations. Whereas the NF model achieves reasonably good and stable performance in closed system configuration (Figure 7.11), it fails to do so once the system is open (Figure 7.4.1). The worst performing FF model is unable to achieve stability in neither the open nor closed system configurations and thus is unable to adapt to changing service

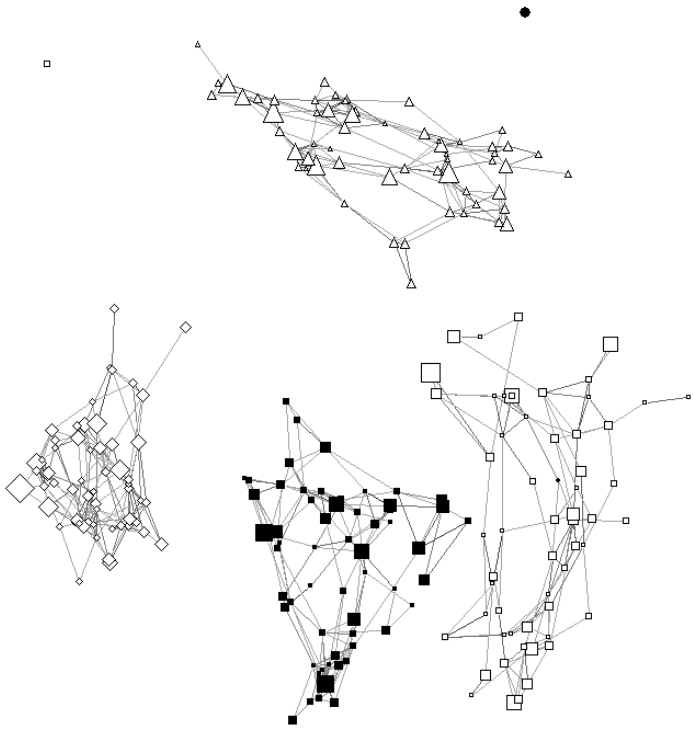

FIGURE 7.17: Correctly organised consumer agent communities extracted from AF model for conditions where consumer turnover probability is equal zero ($\chi = 0$) and there exists demand for four unique service types. Edges between nodes represent information exchanges between consumer agents where node shapes correspond to specific (denoted by node label) service type the consumer is required to allocate. The size of each node indicates the amount of resource capacity that is currently demanded by the task.

demand. In this configuration (the FF model), the slight but counterintuitive improvement in performance for open system configuration stems only from the fact that newly introduced consumer agents are initially co-located with providers that offer the service types they demand. However, the unconstrained flow of information across these agents eventually destabilises the resource market and pushes the system into an inefficient state.

The mean system throughput results achieved by the three models (AF,NF and FF) for different consumer turnover configurations are presented in Figure 7.13. The corresponding accuracy of provider selection conducted by consumer agents during their individual resource allocation is illustrated in Figure 7.14 showing the mean number of task allocation rejects. The dotted horizontal line at the value of 695 that is presented in Figure 7.13 corresponds to the optimal allocation throughput achieved by the closed system, where no reconfiguration was required and thus there was no dynamic demand fluctuation requiring provider agents to reconfigure their provision and consumer agents to track these changes.

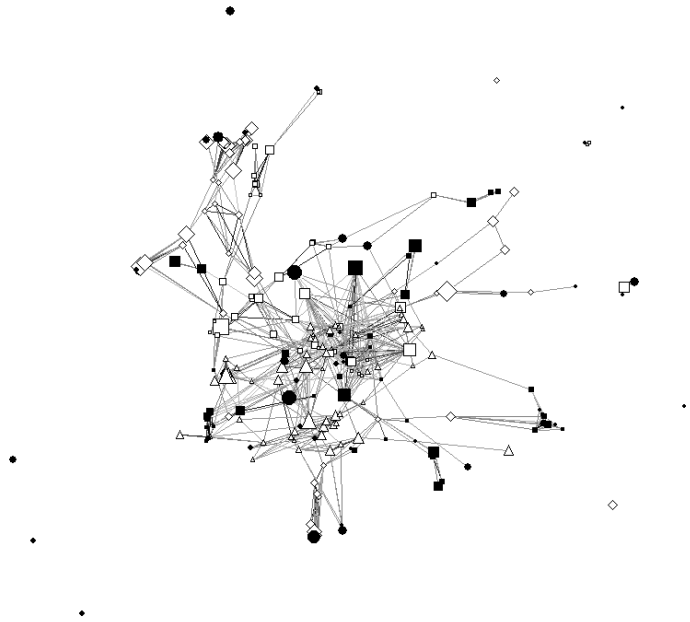

FIGURE 7.18: Disorganised consumer agent communities extracted from FF model for conditions where consumer turnover probability is equal zero ($\chi = 0$) and there exists demand for four unique service types. Edges between nodes represent information exchanges between consumer agents where node shapes correspond to specific (denoted by node label) service type the consumer is required to allocate. The size of each node indicates the amount of resource capacity that is currently demanded by the task.

7.4.2 Consumer Agent Communities

The results reported in the previous section show that the best performance is achieved by the AF model configuration in which agents communicate information. Given the poor performance of the FF model within which agents also communicate information, this implies that only under certain conditions can such information exchange facilitate system adaptation and efficiency. The community analysis presented in Figures 7.15 and 7.16 that shows the mean number of extracted communities (Figure 7.15) and their homogeneity (Figure 7.16) for the AF and FF models supports the hypothesis that organisation of agents into coherent and stable collectives plays a crucial role in achieving system adaptation in dynamic conditions. Here, the inefficiency of the FF model in all experiments is complemented by its inability to establish a sufficient set of communities (with respect to the number of uniquely demanded services, the lower limit of which is presented in the figure as a dotted horizontal line) that in turn results in low homogeneity of extracted communities. In contrast, the AF model, achieving the best response of all models, is characterised by a sufficient number of communities required to fulfill heterogeneous service demand as well as high homogeneity of these communities, implying that agents forming them share knowledge in a manner that supports service provisioning with minimal resource competition and market destabilisation.

Example graphs showing communities for the AF and FF models are provided in Figure

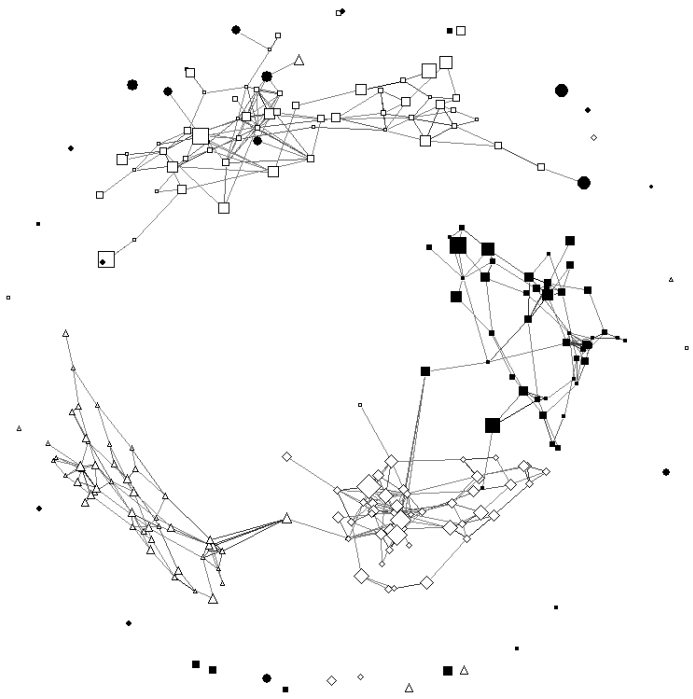

FIGURE 7.19: Correctly organised consumer agent communities extracted from *AF* model for conditions where consumer turnover probability is equal 0.1 ($\chi = 0.1$) and there exists demand for four unique service types. Edges between nodes represent information exchanges between consumer agents where node shapes correspond to specific service type the consumer is required to allocate. The size of each node indicates the amount of resource capacity that is currently demanded by the task.

7.17, Figure 7.18 and Figure 7.19.

7.5 Adaptive Service Provisioning with Continuous Environmental Change

The previous experiments assumed that the demand for different service types was varied only at certain simulation periods, predefined by the discrete demand function. In between rapid demand changes influenced by this function, the resource market was perturbed only through the influx of new consumer agents to the extent regulated by χ. In this section we explore the validity of Hypothesis 2 in conditions where the demand change is not discrete but continuous thus requiring ongoing system adaptation over its life-time.

To achieve this, the change in service type demand is set up with the help of a sinusoidal function that models the gradual turnover of consumer agents and thus the service types they are required to allocate. In all considered in this Section experiments the model is initialised in conditions where consumer population is divided into eight equal subsets, each demanding unique service type, as defined in Table 7.3. Provider agents are, in turn,

$\|Capability\alpha_p\|$	Set of service types ($Capability_{\alpha_p}$)	Number of Consumers	Number of Providers
8	$\{A, B, C, D, E, F, G, H\}$	250	60

TABLE 7.3: The number of demanded service types ($|Capability\alpha_p|$) for the sinusoidal demand function.

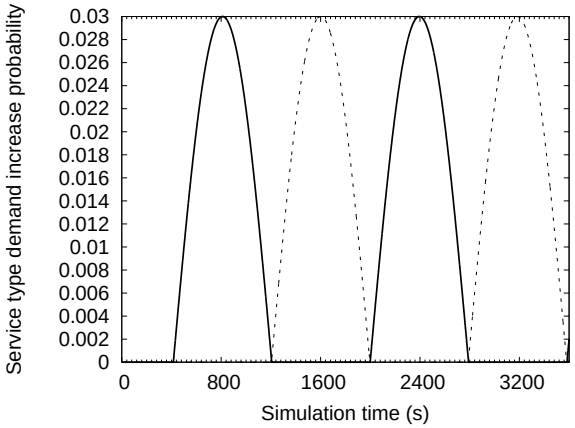

FIGURE 7.20: Figure illustrates sinusoidal function according to which demand for the subset of service types in phase (dotted line) and service types in the anti-phase (solid line) increases proportionately to the probability defined on Y axis. In here, the function period is set to $\Xi = 4.4$ where the maximum probability value the function achieves is $\Theta = 0.03$.

allowed to configure themselves to offer one of the eight services (($|Capability\alpha_p| = 8$).

After the initial 400 simulation seconds, within which the demand for each unique service type remains unchanged and proportional to each other, a sinusoidal function is initialised that gradually and continually influences the change in demand for particular subsets of service types. These subsets are determined by dividing the total number of system service types offered by the into equal sizeed subsets: $d1 = \{A, B, C, D\}$ and $d2 = \{E, F, G, H\}$. Given these two subsets, we assume that the demand intensity for each varies over simulation time according to sinusoidal functions illustrated in Figure 7.20. Here, the solid line defines the probability that consumers allocating services from $d1$ deviate from this configuration and pursue the allocation of randomly selected services from $d2$. Correspondingly, the dotted line in the figure illustrates the opposite situation where consumers become influenced to abandon service type allocation from $d2$ in favour of a randomly selected service from $d1$. Since both sinusoidal functions are in anti-phase, the system is thus allowed to continually and gradually leap between extreme configurations in which all consumers allocate services from $d1$ or $d2$.

The characteristics of this sinusoidal demand function is determined by two parameters: the sinusoidal function period (Ξ) that defines the function phase duration (in Figure 7.20 $\Xi = 2.2$ and lasts 800 simulation seconds) and the maximum probability of consumer service type demand change (Θ) that influences the transition between $d1$ and $d2$ (in the

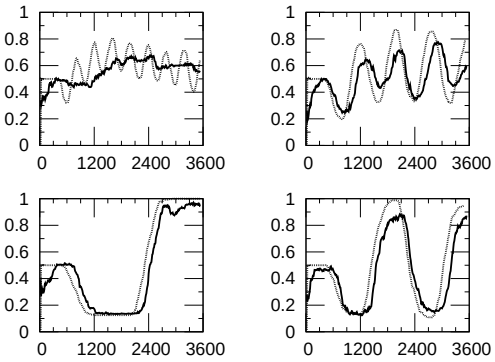

FIGURE 7.21: Demand change for $d1 = \{A, B, C, D\}$ services subset (dotted line) and corresponding supply of resources constituting $s1$ subset for AF model configuration. Figures, in a clockwise direction (starting from a top left one), illustrate demand-supply match for following sinusoidal function periods $\Xi \in \{1.1, 2.2, 4.4, 8.8\}$. In all experiments the maximum probability of consumer changing their service type preferences is equal to 0.03 ($\Theta = 0.03$).

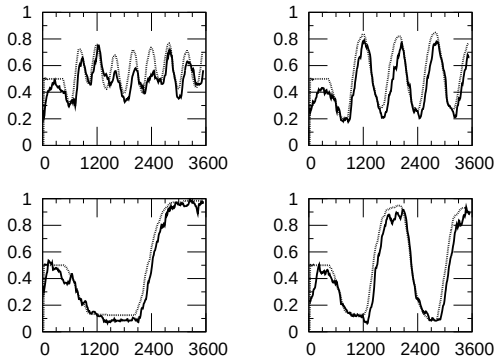

FIGURE 7.22: Demand change for $d1 = \{A, B, C, D\}$ services subset (dotted line) and corresponding supply of resources constituting $s1$ subset for NF model configuration. Figures, in a clockwise direction (starting from a top left one) illustrate demand-supply match for following sinusoidal function periods $\Xi \in \{1.1, 2.2, 4.4, 8.8\}$. In all experiments the maximum probability of consumer changing their service type preferences is equal to 0.03 ($\Theta = 0.03$).

figure this is set to 0.03). Both parameters are varied during experimental evaluation and their impact on system performance investigated.

During sinusoidal demand change it is assumed that consumer agents evaluate their intention to deviate from the current service type demand at each task allocation according to the current Θ probability that changes accordingly to the sinusoidal function. Once the probability is sufficiently high and the agent decides to change its service type preference, it is being removed from the system and substituted by the new agent. This substitution is based on the consumer turnover mechanism discussed in Section 5.4.1.

Apart from the sinusoidal demand oscillations between $d1$ and $d2$, the experiments below also assume the existence of 'noise' provided through a consumer turnover mechanism that introduces demand change fluctuations, the degree of which is controlled through consumer turnover probability (χ parameter).

7.5.1 Resource Market Adaptation in Continuously Changing Environment

The demand change influenced by the sinusoidal function for the AF and NF models is presented in Figure 7.21 (for the AF model) and Figure 7.22 (for the NF model). The results show how the demand change for $d1 = \{A, B, C, D\}$ services subset (dotted line)

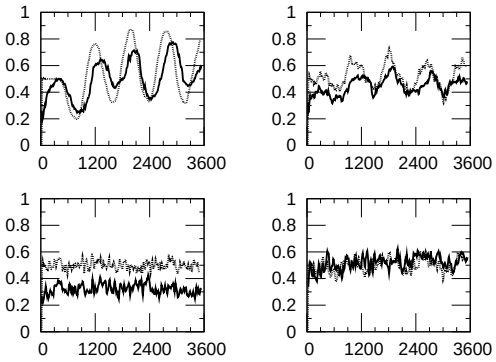

FIGURE 7.23: Demand change for $s1 = \{A, B, C, D\}$ services subset (dotted line) and corresponding supply of resources constituting $s1$ subset for AF model configuration. Figures, in a clockwise direction (starting from a top left one), illustrate demand-supply match for following consumer agent turnover probabilities: $\chi \in \{0.0, 0.1, 0.2, 0.3\}$. In all experiments the the sinusoidal function period Ξ remains set to 2.2 ($\Xi = 2.2$).

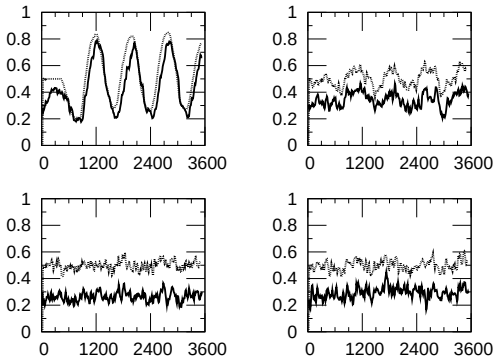

FIGURE 7.24: Demand change for $s1 = \{A, B, C, D\}$ services subset (dotted line) and corresponding supply of resources constituting $s1$ subset for NF model configuration. Figures, in a clockwise direction (starting from a top left one), illustrate demand-supply match for following consumer agent turnover probabilities: $\chi \in \{0.0, 0.1, 0.2, 0.3\}$. In all experiments the the sinusoidal function period Ξ remains set to 2.2 ($\Xi = 2.2$).

is accompanied by the reconfiguration of the system resource market (solid line). In all figures, the demand (and supply) for $d1$ is captured as the proportion of $d1$ demand with respect to the total demand imposed by all system agents.

For the initial conditions (up to 400 simulation seconds) during which the sinusoidal demand is not activated, the demand for both service type subsets $d1$ and $d2$ is equal and thus each subset consumes half of the total system demand as indicated in the figures by the dotted line near 0.5 value on Y axis. However, after the initial 400 simulation seconds the sinusoidal function triggers the demand change, requiring thus the provider agent reconfiguration and adjustment to changing consumer interests.

Whereas the performance of the FF model (not shown in the figures) in every sinusoidal function period is poor and unstable, Figures 7.21 and 7.22 show that both AF (former figure) and NF (latter figure) respond to demand change appropriately in all sinusoidal function period settings. Whereas the results for conditions in which the sinusoidal function period is very short ($\Xi = 1.1$) and lasts only 400 simulation seconds (top left sub-figures for each model) show that the system may not adjust its configuration perfectly as the demand configuration changes sufficiently fast for the system to be able to compensate it optimally, the longer sinusoidal periods provide enough time for smooth and efficient adaptation for both the AF and NF models. It is also interesting to observe that resources market adjustment for the NF model (Figure 7.22) represents a slightly more accurate match to the changing system demand than the results observed by the AF model (Figure 7.21) that shows more resilience to the demand changes, especially

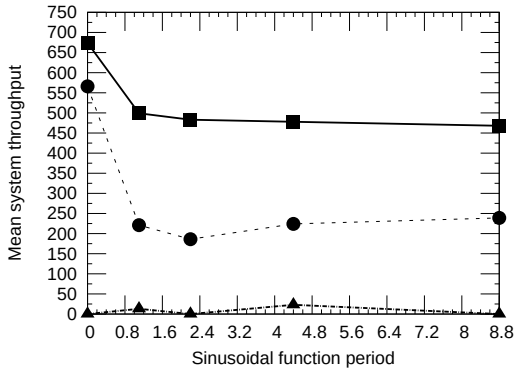

FIGURE 7.25: Mean system throughput as a function of changing sinusoidal demand function period (Ξ) depicted on X axis. Results summarise performance of the three model configurations: AF (solid rectangles), NF (solid circles) and FF (solid triangles). In all experiments $\Theta = 0.03$ and $chi = 0.1$.

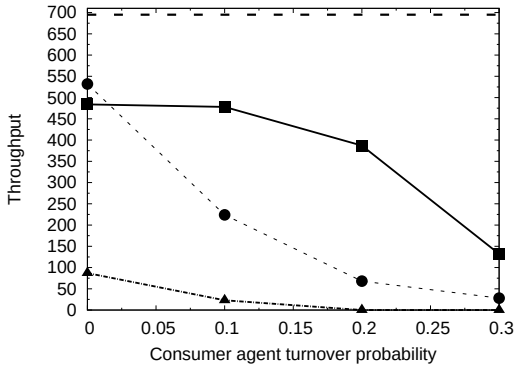

FIGURE 7.26: Mean system throughput as a function of consumer turnover probability (χ). Results summarise performance of the three model configurations: AF (solid rectangles), NF (solid circles) and FF (solid triangles). In all experiments $\Xi = 2.2$ and $\Theta = 0.03$.

for configurations in which the sinusoidal demand period is low and the system has to reconfigure quickly. This slight impact on system adaptation speed for AF model stems from the information flow regulatory mechanisms that prevent the system from moving into an unstable and chaotic state due to perturbations inflicted by the demand change.

The advantage of information flow and its regulation is shown in Figures 7.23 and 7.24 that provide the same comparison between the AF model (Figure 7.23) and NF model (Figure 7.24) but in conditions where the system is open and thus the influx of new consumer agents is allowed during its life-time. In this configuration, the amount of 'noise' inflicted by the introduction of new agents is regulated through consumer turnover probability (χ). For this purpose, each sub-figure (starting from the top left and descending in a clock-wise manner) for particular model performance represents results where χ parameter values are incremented for each corresponding sub-figure according to: $\chi = \{0.0, 0.1, 0.2, 0.3\}$. The performance of the AF and NF models obtained in this manner in an open environment show that the efficiency of smooth adaptation to changing sinusoidal demand becomes negatively affected through the influx of new consumer agents. However, whereas the AF model is capable of matching the supply of resources to the current demand up to the configuration in which $\chi = 0.3$ (bottom-right sub-figure in Figure 7.23), the performance of the NF model degrades much earlier and is observed even for conditions where turnover probability is low when $\chi = 0.1$ (top-right sub-figure in Figure 7.24). For this model, the inability to satisfy the demand stems from resource market frustration that causes a substantial number of providers to continually reconfigure and thus become unable to provide any resources.

Figures 7.25 and 7.26 summarise mean system throughputs achieved by the three dis-

cussed system models in configurations where the sinusoidal function period (Figure 7.25) and consumer turnover probability (Figure 7.26) were varied. The best performance in every configuration was achieved by the AF model capable of sustaining system organisation and degrading gracefully once environmental dynamics perturbs the system. A model configuration (NF) in which agents did not communicate information and thus relied only on their personal and local knowledge was capable of achieving high throughput only when the system was closed and there was no instability introduced through the arrival of new consumer agents. The worst performance was observed for the FF model. In this configuration the resource market was frustrated and unstable in every configuration due to unregulated information flow across agents.

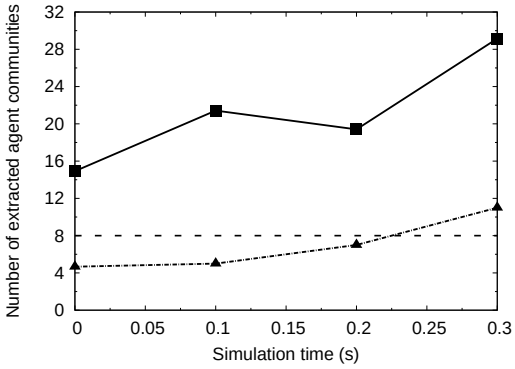

FIGURE 7.27: Mean number of extracted communities as a function of increasing consumer agent turnover probability for two system model configurations: AF model (line with rectangles) and FF model (line with triangles).

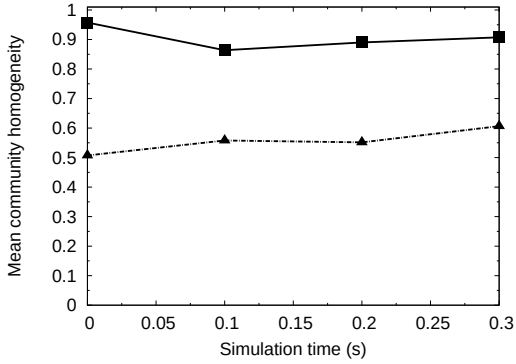

FIGURE 7.28: Mean community homogeneity as a function of increasing consumer agent turnover probability for two system model configurations: AF model (line with rectangles) and FF model (line with triangles).

7.5.2 Consumer Agent Communities

Similarly to load-balancing experiments conducted in the previous chapter, community analysis reveals that the adaptive service provisioning evaluated in this chapter is also dependent on the ability of system agents to organise into communities, each such community supporting in this context the provisioning of different service type. Figures 7.27 and 7.28 outline community characteristics extracted from the AF and FF models. The number of extracted communities for the AF model (Figure 7.27) satisfies the lower limit (denoted in the figure as dotted horizontal line) on the number of unique communities that need to be formed in order to satisfy demand for eight unique service types demanded over the simulation time. This correct configuration is followed by high homogeneity of extracted communities presented in Figure 7.28. Community characteristics for the FF model show that not only it is unable to organise agents into a minimal required number of communities but the communities it forms exhibit low

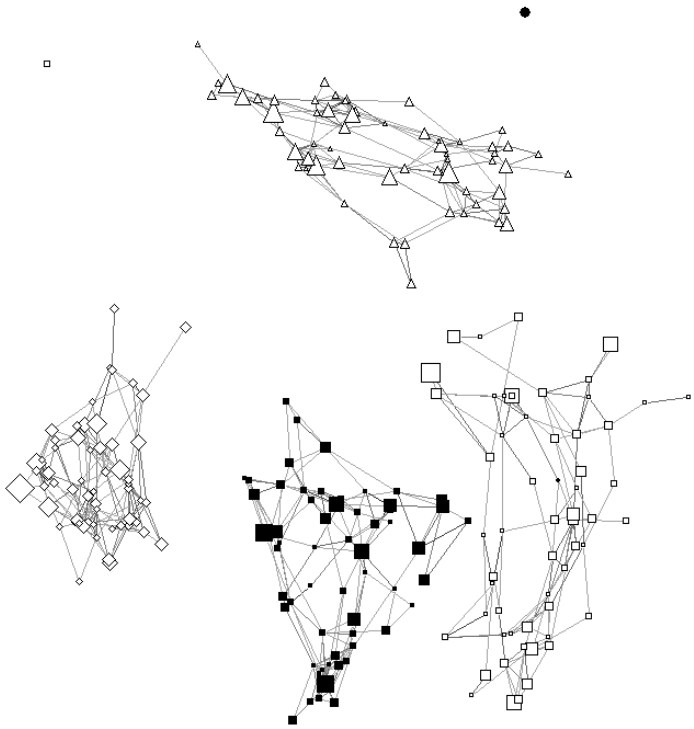

FIGURE 7.29: Correctly organised consumer agent communities extracted from AF model for conditions where consumer turnover probability is equal zero ($\chi = 0$) and there exists demand for four unique service types. Edges between nodes represent information exchanges between consumer agents where node shapes correspond to specific (denoted by node label) service type the consumer is required to allocate. The size of each node indicates the amount of resource capacity that is currently demanded by the task.

level of homogeneity.

Example graphs showing communities for the AF and F models are provided in Figure 7.29, Figure 7.30 and Figure 7.31.

7.6 Conclusions

In this chapter we have continued our evaluation of bottom-up system organisation efficiency within conditions in which the resource market must adaptively reconfigure in response to dynamic demand changes. To achieve this we employed stimulus-response mechanisms inspired by a study of emergent behaviours within biological systems (insect colonies) and evaluated their efficiency at regulating resource markets within a decentralised autonomic system model.

The obtained results confirmed the observations obtained in the previous chapter and supported the hypothesis that bottom-up adaptation is facilitated through the emergence of stable but reconfigurable communities of agents that specialised themselves to

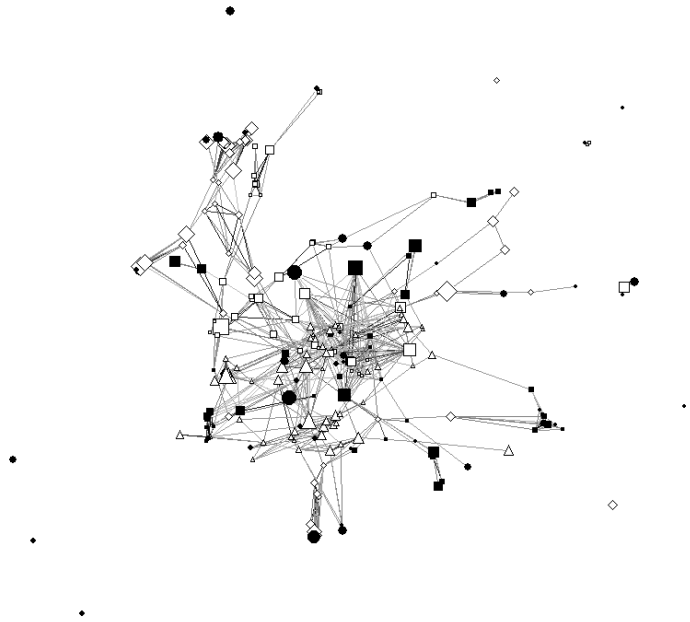

FIGURE 7.30: Disorganised consumer agent communities extracted from FF model for conditions where consumer turnover probability is equal zero ($\chi = 0$) and there exists demand for four unique service types. Edges between nodes represent information exchanges between consumer agents where node shapes correspond to specific (denoted by node label) service type the consumer is required to allocate. The size of each node indicates the amount of resource capacity that is currently demanded by the task.

perform a specific role within the system (eg. provision of a particular service type). Likewise in the results obtained during load-balancing, the role of the observed agent communities is to restrict the flow of information about system resources only to a subset of agents. These agents, in turn, learned to communicate information only to peers forming the same community and thus supported individual agents with up-to-date and community-level information about the system resource state that would not be possible if agents were acting independently, as in the case of the NF model. The advantage of agent organisation into communities is shown in Figure 7.32 illustrating the content of local consumer agent registries for the first set of experiments reported in this chapter (Section 7.3) in which consumer turnover probability equals 0.1 ($\chi = 0.1$). Apart from supporting individual consumers with up-to-date information about provider efficiency, communities also restricted knowledge and thus attraction towards providers that are not of interest for community members (eg. providers that offer other service types). By doing so, the overall resource market was able to proliferate into smaller sub-markets within which providers specialised at reliably offering demanded services and consumers were able to establish attraction to a subset of such providers.

This research highlights a useful property: that of *functional substitutability*; i.e. the ability of a component to change its behaviour in response to changes in the need for particular functionality. This is desirable with many scenarios where there is a risk of failure in one component, such as within robotic rescue scenarios whereby a robot

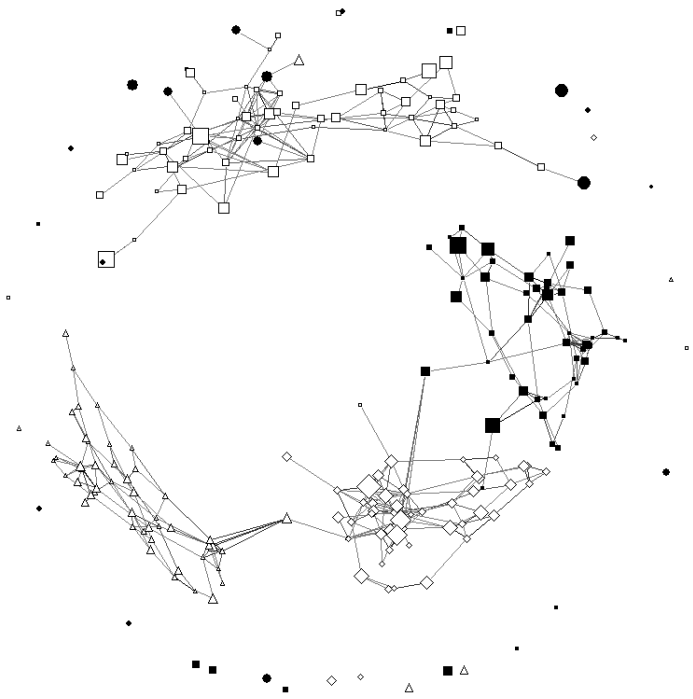

FIGURE 7.31: Correctly organised consumer agent communities extracted from *AF* model for conditions where consumer turnover probability is equal 0.1 ($\chi = 0.1$) and there exists demand for four unique service types. Edges between nodes represent information exchanges between consumer agents where node shapes correspond to specific service type the consumer is required to allocate. The size of each node indicates the amount of resource capacity that is currently demanded by the task.

may be able to perform several different tasks (e.g. sensing, moving, performing actions etc), but only a few of these are needed for any given scenario. However, changes in the environment may necessitate corresponding changes in the roles (at runtime) that these robots perform. Similar behaviours have been observed within biological systems. In natural ant colonies, it has been observed that within what is often described as a homogeneous community of ants sharing the same behavioural repertoire, there arise leaders that strongly stimulate the actions of other ants by their more frequent activity [18]. Such *hyperactive* ants stimulate the workload of the colony in every aspect (food foraging, nest building, brood feeding, etc.). Thus, *leader* and *follower* roles arise under certain conditions (rather than being inherent, programmed behavours). However, a complete understanding of this social stimulation process is still unclear.

In what follows, we extend the model functionality by focusing on the power management problem in which not only are system elements required to perform efficient load-balancing and service provisioning, but they also decide how much resource should be kept on-line (or moved off-line) in order to appropriately match the changing intensity of demand. As such dynamic power management demands more adaptation at the level of autonomous system elements, we extend the model with novel algorithms facilitating this.

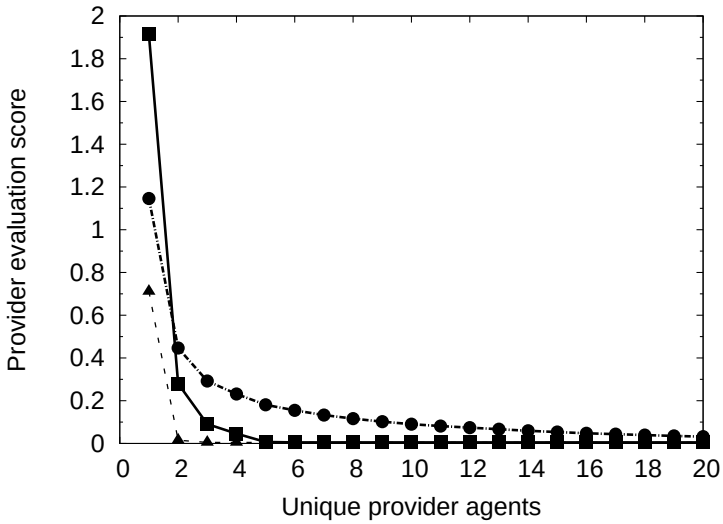

FIGURE 7.32: Provider evaluation scores kept within local registries of consumer agents for: AF model (line with rectangles), FF model (line with circles) and NF model (line with triangles) for conditions in which consumer turnover probability equals 0.1 ($\chi = 0.1$)

Chapter 8

Power Management

8.1 Introduction

Within the autonomic system models explored in the previous two chapters, an implicit assumption was made that demand is often proportional to the total supply of system resources. Under these conditions, the ability of the system to balance the load across available providers and their efficiency to reconfigure service type provision for changing demand was evaluated.

However, within real autonomic systems it is likely that the changing patterns of user activity will lead to changes in the intensity of resource demand. As a result, the demand for system resources will no longer always be close to the maximum supply but will vary over time. Thus, to preserve a high level of efficiency and low maintenance cost, the system needs to appropriately adjust the level of resources supplied to match the current demand.

Since the power consumed by the server delivering some computational resources is proportional to the frequency of the CPU of the machine, Kandasamy *et al.* [76] propose a control mechanism that adjusts the processor speed to the level that is characterised by minimal power consumption but is still capable of delivering the required quality of service. This is achieved with the help of a control mechanism that predicts job request arrival rates for the processor and, based on these predictions, throttles the regulated CPU speed such that expected efficiency (according to predicted workload) is achieved. Experiments with only a single server node imply that optimal solutions for such on-line control problems do not exist, particularly when the task arrival rates are unpredictable and potentially unbounded. As a result, the system designer is required to decide upon an acceptable controller configuration after sufficient experimentation. Although a control mechanism for a single server is offered, the authors do not consider a situation where a cluster of nodes need to coordinate adjustments of their internal CPU speed configurations. This is a major drawback of the approach, as it only focuses

on power management of a single server node and does not consider the impact of its regulation on a large scale, where possibly hundreds of servers offering various services would be deployed.

Whereas the main focus of the work described above is to regulate power consumption at the level of an individual server, Das *et al.* [98] present work aiming to control energy consumption of a group of servers. To preserve efficient power management within a medium-sized server cluster housed in a single IBM BladeCenter chasis, the authors employ a multi-agent system model. Agents are organised to form a hierarchical control topology (a so called 'Unity architecture'), where a group of servers is managed by *application managers* that, in turn, are controlled by a *resource arbiter* agent. Given this model, the performance optimisation of servers is carried on in the following manner. The information about resource needs (given the current workload) is sent by all application managers to a resource arbiter. Based on this information, the resource arbiter agent calculates the optimal allocation and instructs subordinate agents to appropriately configure the state of servers managed by them. To optimise over multiple competing criteria such as application performance and power consumption, two agents are introduced: a performance agent (required to preserve expected level of system efficiency) and Power agent (controlling power consumption and regulating which servers should move into off-line mode). To optimise and align the decisions of both agents, a coordination agent is introduced. This work, in contrast to the previously described approach, focuses on power management within a cluster of servers, rather than a single computational machine. As a result, the more realistic problem of coordinating the configuration of a group of servers is considered and for this purpose multi-agent system architecture is proposed. Furthermore, it is also observed that power management decisions need to be aligned with the system efficiency optimisation task and thus cannot be solved in isolation. However, despite the distributed nature of the proposed control mechanism, Das *et al.* assume that agents within their model have access to the most up-to-date state of all computational system nodes, such that optimal allocation of resources and power management disposition by controller agents can be made. Whereas this is plausible for small scale deployments, relying on overall system state information to make optimal decisions may not be realisable within systems comprising hundreds of such servers. It is thus unlikely that the proposed approach would be easily scaleable and robust to increasing system scale and dynamism.

A remedy to this problem is proposed in [11]. In this work, Brueckner and Parunak propose a system in the form of a mobile ad-hoc network of computational nodes that have limited power and therefore are required to maintain a minimal energy consumption level that still facilitates the level of service required from the system. Similar to the above approach, the authors employ a multi-agent system architecture. However, in contrast to the two previous examples, control over the state of individual nodes is devolved to individual system elements and thus is fully decentralised. As a result,

there are no dedicated controller agents that require access to the state of subordinate nodes. Rather, the individual system nodes adapt and adjust their current configuration relying only on local information, employing for this purpose simple algorithms inspired by the decentralised process of self-organisation within insect societies. As a consequence of avoiding hierarchical distributed control design, the system is shown to be scaleable and robust to changes in dynamism and failures of its individual elements, yet still capable of preserving the expected level of efficiency and power usage. However, whilst offering the required flexibility and bottom-up adaptation, the approach proposed by Brueckner and Parunak does not consider the situation where individual nodes may offer different services and thus the population of interacting agents has to adaptively adjust its provision in dynamic conditions.

For this reason, sympathetic with a decentralised approach described above, in our work we propose a model in which the system is required to efficiently manage power consumption, at the same time preserving adaptive service provisioning. To achieve this, we offer an extension of our decentralised multi-agent system model described in Chapter 5. The extended architecture offers novel decision-making mechanisms that rely on *stimuli-response* principles inspired by the division of labour within natural self-organising systems that allow individual service providers to autonomously decide which ones should be in an active state (on-line) and which should move into a passive mode (off-line) thus saving power.

Using this model, we continue our hypothesis examination carried out in the last two chapters, where we observed that both efficient load-balancing and adaptive service provisioning functionalities depend on the ability of system agents to organise into communities. In this chapter we examine this hypothesis within a more complex setup where the decentralised multi-agent system, on top of the two above functionalities, has to deliver efficient power management.

To this end, in Section 8.2 we describe power management issues raised by autonomic computing and the relevance of agent communities in achieving it within a decentralised system model. Following this, we introduce an extension to our autonomic system model that features local decision-making mechanisms facilitating power management functionality and performance measures that we will rely on during the experimental evaluation. The power management efficiency achieved by our model is then evaluated in the two following Sections, Section 8.4 and Section 8.3, each modeling different demand conditions in which the system has to operate. Finally, the conclusions based on our experimental work are presented in Section 8.5.

8.2 Power Management

8.2.1 Power Management Problem

As observed in [98] the rapidly rising cost and environmental impact of energy consumption in data centres has become a multi-billion dollar concern globally. This is attributable to several alarming trends [46, 62]:

- In 2005, data centre servers accounted for 1.2% of electricity use in the United States, doubling since 2000.
- By 2008, 50% of existing data centres are assumed to have insufficient power and cooling.
- By 2008, power are assumed be the second-highest operating cost in 70% of all data centres
- Data centres are responsible for the emission of tens of millions of metric tons of carbon dioxide emissions annually more than 5% of the total global emissions.

As a result, efficient power management has become a highly desirable, if not critical, characteristic of any large scale IT infrastructure. However, to realise it, a number of challenges need to be met. In particular, power management solutions that either idle the CPU speed of individual data centre servers or turn whole servers into *off-line* mode should not compromise the overall infrastructure performance or negatively affect the availability of demanded resources. Ideally, the system is required to smoothly adjust the proportion of *on-line* and *off-line* servers (or on-line available capacity) in a timely response to changing demand intensity. At the same time, other system features such as load-balancing and adaptive service provisioning should not be affected, thus achieving high system throughput at a low maintenance and energy cost.

Achieving these cross-interest objectives within a dynamic environment is not a trivial task and requires more intervention than human administration can offer. For this reason, to facilitate resource management on top of load-balancing and adaptive service provisioning features, most of the approaches described in the previous section devolve control over these functions to autonomic mechanisms incorporated within a computational system. Such an approach is implemented at various levels of the system architecture and with the application of different control mechanisms.

In this thesis we address this problem on the level of provider agents that are allowed to autonomously decide whether to go into an *on-line* or *off-line* state. Since these decisions are made based only on the locally available information to provider agents, a number of problems may arise during this decentralised decision-making that may affect

not only power management efficiency but also the overall performance of the system. In particular, whilst power management aims to decrease the amount of available on-line system resources, the insufficient number of such on-line resources may impact on the performance of the system as it would not be able to respond to dynamically changing demand. As a consequence, the system is required to continually adjust the level of the resources that are kept in an *on-line* state such that the overall system efficiency as well as power management are not compromised.

More importantly, the system cannot achieve a high level of power management efficiency if it is unable to balance the load across available system providers and appropriately configure resource market to provide services that satisfy consumer demand. As we observed in the last chapter discussing adaptive service provisioning functionality, the inability of the system to reliably maintain this function causes destabilisation of resource market and leads to inefficient provider agent reconfigurations that spent most of their time in an *on-line* state but are unable to offer any resources due to frequent and timely reconfigurations.

Consequently, also in this chapter we assume that power management efficiency is directly related to the ability of system agents to organise into communities and, for this reason, focus on the examination of the following hypothesis:

Hypothesis 3:

We hypothesise that only model configurations in which agents are able to organise into communities are able to sustain a sufficient number of provider agents in an *on-line* state such that efficient power management emerges on the global system scale and the system resources supply matches consumer demand.

In the remainder of this chapter we examine this hypothesis in the same principled manner as we did for load-balancing and adaptive service provisioning presented in the last two chapters. To do so, we evaluate efficiency of power management in dynamic resource allocation conditions using three model configurations, AF, NF and FF, that differ between each other in mechanisms that agents employ for information communication among peers.

8.2.2 Extended Model for Service Providers

Whereas a detailed account of the provider agent architecture was presented in Chapter 5, the remainder of this section provides details regarding the extension of the provider agent model aimed at facilitating power management. It is important to note that, whilst focusing on power management, the mechanisms employed for this are required to function in concert with the previously described load-balancing and adaptive service

provisioning techniques. As a consequence, the system is required to adjust and optimise its functioning over all three competing criteria: power, performance and robustness.

Power management involving shut-down of computational resources is determined by the *stimulus-response* mechanism inspired by the division of labour within insect societies [7, 120] and is analogous to the decision-making algorithms (described in Section 8.2.2) responsible for adaptive service provisioning. However, here, rather than choosing between various service types, the provider is required to decide whether to move into an on-line and off-line state.

To make these decisions the provider uses the locally available information about its current utilisation level (U) and the maximal demand estimate (s_m) representing the highest stimulus (s) value within its internal service type registry ($\mathcal{R}_p$) experienced from interacting consumer agents with it. Based on these values, the agent calculates the probability of staying on-line (p_o). This is done according to the formula below, derived from the self-organising model of an ant society, found in [7, 120] to adequately model the division of labour within that decentralised natural system:

$$p_o = \frac{s_m^2}{s_m^2 + \kappa^2}$$

Analogously to the algorithm governing the adaptive service provisioning described in Chapter 5, here the probability of staying on-line is also dependent on the agent persistence to remain at the current state. Such persistence is represented by the response threshold (κ), the value of which has an impact on the calculated probability p_o.

To align the response threshold with the chracteristics of autonomic system computing, we extended the original response threshold functions (introduced in [7, 120]) by making them responsive to the current provider agent utilisation level (U). For this purpose the response threshold value (κ), associated with a provider on-line state, becomes updated every decision cycle (every simulation second) in the following manner:

$$\kappa \rightarrow \kappa - 2 \times (U + 0.5),$$

if the provider is currently on-line and

$$\kappa \rightarrow \kappa - U + 0.5$$

otherwise. Based on the calculated probability, the provider determines (every decision cycle) the current power management mode in a probabilistic manner by applying roulette wheel selection. The values of κ are allowed to reside in $\langle 0, .., 10 \rangle$ ($0 \leq \kappa \leq 10$), such that whenever the current κ parameter value moves beyond this range, it is set to

its maximum (when being too large) or minimum (when being too low) [1].

If the provider was on-line and decides to move into an off-line state, it will cease to accept new allocation requests, finish the current allocations and after time T_o become inactive. Moving from an off-line state into an on-line one will consume the same T_o time, after which the provider will offer one of the most demanded services, determined by the adaptive service provisioning mechanism.

The algorithm, including power management functionality, used by a provider agent remains identical to the one presented in Section 5.3.2, with one exception. Before deciding which service type to offer, the provider employs presented in this section algorithm to evaluate whether it should go into off-line (or on-line) state.

8.2.3 Power Management Efficiency Analysis Measures

Throughout experiments conducted in this chapter we rely on the default model configuration presented in Chapter 5, and its experimental setup (Section 5.4). If any parameters in this section change this is explicitly stated. Apart from the system throughput and network analysis measures, described in Section 5.5, the power management efficiency analysis is extended by two additional measures: *total on-line resource capacity* and *available on-line resource capacity*. Both are described below.

8.2.3.1 Total On-line Resources Capacity

The total on-line resource capacity defines the total amount of capacity offered by the whole population of provider agents that are in an on-line state. This measure is calculated in the same sampling time intervals as system throughput (described in Section 5.5). For the mean value of total on-line resource capacity, we average the sum of obtained measurements (taken from distinct sampling periods) by the total number of obtained samples within that period.

8.2.3.2 Available On-line Resources Capacity

Whereas the *total* on-line resource capacity defines the total capacity that is on-line, the *available* on-line resource capacity identifies how much on-line resource can be employed for task allocation. The difference between them results from the fact that some of the configured on-line provider agents may be reconfiguring their service provision, thus being unable to offer resources at that time despite being in an on-line state.

[1] The upper and lower limits on the κ parameter values were determined through experimental evaluation. During this process, the best parameter range was selected that allowed for responsive provider on-line and off-line reconfiguration in a wide range of experiments.

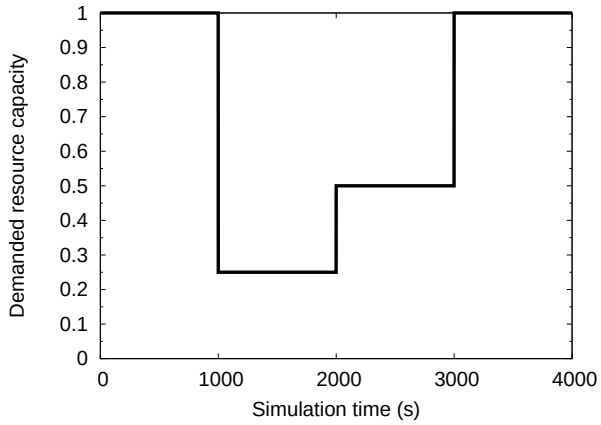

FIGURE 8.1: Figure illustrates the demand intensity step function according to which the demand level (represented on Y axis) for system resources undergoes rapid change at simulation periods indicated on X axis. Here, the value of 1 on the Y axis indicates conditions at which demand-supply proportion is equal and the system operates at its full capacity.

The available on-line resource capacity is calculated in the same sampling time intervals as system throughput. For the mean value of available on-line resource capacity, we divide the sum of obtained measurements (taken from distinct sampling periods) by the total number of obtained samples.

8.3 Power Management with Discrete Environmental Change

In this analysis, the system's ability to preserve adaptive power management was examined over a range of scenarios in which the demand intensity as well as the numbers of service types demanded changed at run-time, thus requiring the system to dynamically adjust to the new conditions. To model the dynamic service demand environment, a step-based function (illustrated in Figure 8.1) was provided.

This function modeled the particular case of an open system where periods of system stability, in which the number of agents, and thus system's composition, remained unchanged, were punctured by a rapid removal or introduction of new consumer agents, requiring the provider population to adapt to changing levels of demand. Similarly to the experiments in the previous chapter, the discrete demand intensity function was coupled with a consumer turnover mechanism that allowed us to regulate the degree to which the system is open, and thus perturbed, through the influx of new consumer agents.

Based on the perceived service demand, providers can choose to offer a single service selected from the set $Capability_{\alpha_p}$, as defined in Table 8.1. Given the variety of different service types that are introduced during demand intensity changes (illustrated in Table 8.1 for each configuration change), apart from power management, the efficiency of

Demanded service types	Set of demanded service types	Demand-supply ratio	Number of Consumers	Number of Providers
8	$\{A, B, C, D, E, F, G, H\}$	1.0	250	60
2	$\{A, B\}$	0.25	62	60
4	$\{A, B, C, D\}$	0.5	125	60
8	$\{A, B, C, D, E, F, G, H\}$	1.0	250	60

TABLE 8.1: The change in the demand intensity (and the number of demanded service types for subsequent steps in the step demand intensity function.

the system is also predetermined by its ability to preserve a balanced usage of on-line configured provider agents as well as their adaptive service type configuration.

Considering these performance criteria, the evaluation of system performance in changing demand intensity conditions is presented below.

8.3.1 On-line Resource Market Adaptation in Discretely Changing Environment

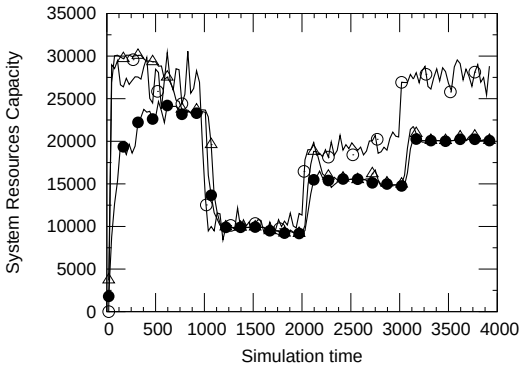

FIGURE 8.2: Figure illustrates the total amount of capacity demanded by consumer agents (empty circles), available on-line capacity offered by provider agents (solid circles) and total on-line capacity (triangles) for the AF model configuration in conditions where system is open and consumer turnover probability equals 0.1 ($\chi = 0.1$)

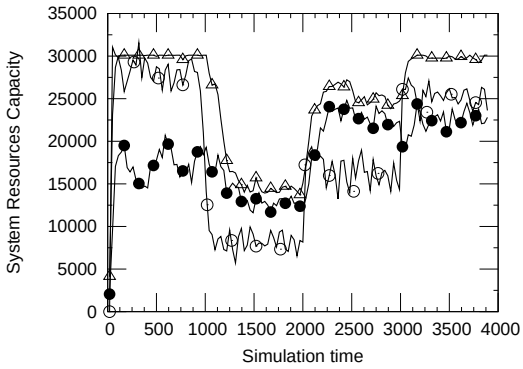

FIGURE 8.3: Figure illustrates the total amount of capacity demanded by consumer agents (empty circles), available on-line capacity offered by provider agents (solid circles) and total on-line capacity (triangles) for the NF model configuration in conditions where system is open and consumer turnover probability equals 0.1 ($\chi = 0.1$).

The power management efficiency for a single run of AF and NF model configurations is illustrated in Figure 8.2 (for the AF model) and Figure 8.3 (for the NF model) [2]. In both figures the power management performance is represented by two measures:

[2]The power management efficiency for the FF model is not represented for this particular case because of its poor adaptation and is presented using comparative analysis in the remainder of this section.

the total on-line capacity offered by the system (triangles) and the amount of available on-line capacity that is ready for consumption (solid circles).

Given the results illustrated in both figures, we distinguish two important conditions that should be considered when analysing and judging the efficiency of power management control. Firstly, in optimal circumstances the total supply of resources provided by the system should match the demand imposed by consumers, and secondly, the supply of on-line resources that are ready for use should be as close to the total amount of available on-line resources as possible. The second condition stems from the fact that whilst provider agents may be in an on-line state, it is not guaranteed that they will be capable of offering resources as they may spend their on-line time continually switching and adapting, thus 'wasting' the resources they could offer otherwise.

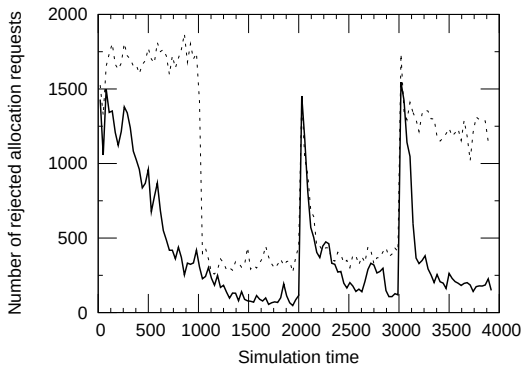

FIGURE 8.4: Mean number of rejected consumer allocation queries as a function of simulation time for two model configurations: AF (solid line) and NF (dotted line). Both models are run in conditions where system is open and consumer turnover probability equals 0.1 ($\chi = 0.1$).

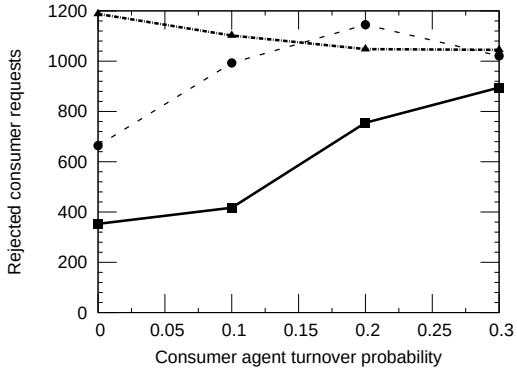

FIGURE 8.5: Mean number of rejected consumer allocation queries for three model configurations: AF (line with rectangles), NF (line with circles) and FF (line with triangles).

The analysis of performance of both models under these assumptions shows that the closest match between the total and available on-line resource capacity is achieved by the AF model. This can be observed in every demand intensity change influenced by the step function that not only requires the resources market reconfiguration during which subsets of providers move into an on-line or off-line state but also are required to reconfigure their specialisation to satisfy the step changing function period of service type demand (defined in Table 8.1). Moreover, this efficient power management adjustment is carried out in open system conditions (turnover probability is equal to 0.1) in which there is a continual influx of new agents that require the resource market to continually readjust and adapt.

Considering the NF model efficiency in adjusting the level of offered resources to the demand intensity (Figure 8.3), a clear disparity between the total amount of offered

capacity (line with triangles) and its available on-line subset (line with solid circles) can be observed. Furthermore, the adjustment of the on-line offered resources is not adequate to the demand imposed by consumer agents (empty circles). This results from the inability of provider agents to adjust and reconfigure to changing service type demand. In conditions when the system is required to operate at full throttle ($0-1000$ simulation time) the resource market is unstable and providers experience a lot of reconfiguration, thus wasting on-line resources. On the other hand, in conditions when the demand intensity (as well as demanded service type variety) is smaller ($1000-3000$ simulation time) the surplus of offered resources to the demand imposed on the system contributes negatively to the power usage.

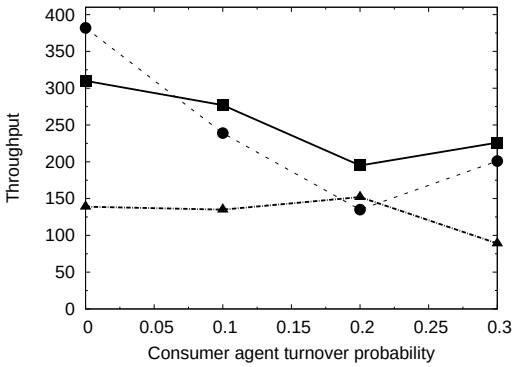

FIGURE 8.6: Mean system throughput as a function of increasing consumer agent turnover. Three model configurations are presented: AF (line with rectangles), NF (line with circles) and FF (line with triangles).

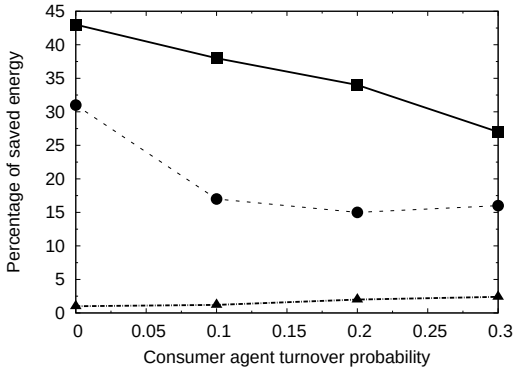

FIGURE 8.7: Power management efficiency for three model configurations: AF (line with rectangles), NF (line with circles) and FF (line with triangles). Lines illustrate percentage of energy that has been saved by provider agents that moved into off-line state.

It is interesting to observe that whilst for the NF model the offered resource capacity is mostly of the time greater than the volume of resources demanded by consumer agents, the supply of resources within the AF model configuration is slightly smaller than the demand, and the system regulates itself towards the least amount of on-line resources.

Before we analyse mean system throughput achieved in the different configurations, let us consider the number of rejected task allocation queries for both. This is shown in Figure 8.4, where the number of rejected task allocation queries for the AF model is indicated by a solid line and for the NF model by a dotted line. Considering these results, we can observe that despite the lower amount of on-line offered resource capacity, the AF model configuration effectively minimises the amount of rejected queries in all demand configurations to its minimal achievable value, residing near 250. For the NF model, on the other hand, we can observe certain periods within which the increased difficulty of resource allocation conditions affects the efficiency of consumer agents in identifying resource providers capable of satisfying their requests.

These results suggest that the AF model not only exhibits much better power management features than the NF model but is capable to achieve high allocative throughput at the same time. Consider Figure 8.6, showing mean system throughput achievable by three model configurations (AF, NF and FF) in increasing consumer agent influx conditions. The corresponding analysis of rejected allocation queries for these configurations is illustrated in Figure 8.5.

The performance results depicted in Figure 8.6 suggest that for conditions when consumer turnover probability is equal to 0 and thus the system is closed, the NF model achieves slightly better performance throughput than the AF one. However, as the power management efficiency analysis in Figure 8.7 shows, this happens at the cost of less efficient energy saving as it involves more on-line provider agents. The percentage of saved energy is calculated by measuring the proportion of providers that moved into an off-line state in comparison to the model configuration in which all providers are configured to all remain in an on-line state.

Considering this power management efficiency, in conjunction with system's throughput performance, we can observe that the NF model achieved greater throughput performance at the expense of less efficient power usage. During these conditions a lower utilisation imposed on a greater number of provider agents allowed them to offer services in a quicker time and eventually contribute to a slightly better throughput than in the AF model case. Within the AF model, on the other hand, consumers tend to employ the least amount of required resources that can still satisfy their task allocation, which results in the slightly longer service provisioning (that still satisfies stringent task time limit properties) thus resulting in a slightly worse throughput than achieved by the NF model.

For conditions where the system is open (turnover probability is greater than 0) the best throughput as well as power management efficiency is achieved by the AF model, followed by the NF one. The worst allocative performance with almost no energy saved is experienced by the FF model. In this configuration the resource market was frustrated to the point where it become impossible to preserve high throughput and efficient power management.

8.3.2 Consumer Agent Communities

Efficient allocative performance of the AF model and the corresponding poor and unstable behaviour of the FF model are confirmed by the community analysis illustrated in Figure 8.8, showing the number of extracted communities, and Figure 8.9, showing the mean community homogeneity values achieved by both models as a function of consumer agent turnover. In the first figure, it is shown that only agents within AF model are able to reorganise adaptively into appropriely sizeed (as denoted by dashed line)

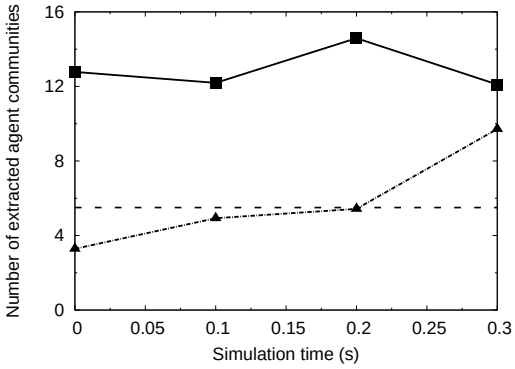

FIGURE 8.8: Mean number of extracted communities as a function of increasing consumer agent turnover probability for two system model configurations: AF model (line with rectangles) and FF model (line with triangles).

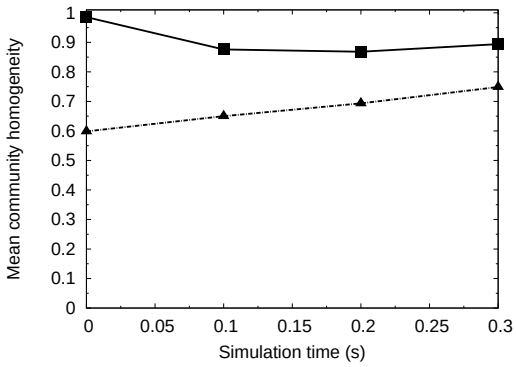

FIGURE 8.9: Mean community homogeneity as a function of increasing consumer agent turnover probability for two system model configurations: AF model (line with rectangles) and FF model (line with triangles).

communities that reflect the current service type demand situation. Also, as the second figure illustrates, only communities formed by the AF agents are characterised by high homogeneity, suggesting that most agents established beneficial information exchange interactions among peers of the same 'kind'. During this analysis, the counterintuitive and gradual increase in community homogeneity for the FF model results from the greater influx of consumer agents that were initially co-located with correct (with respect to offered service type) provider agents. However, this initial configuration was quickly lost due to the disorganised flow of information, giving rise to very poor throughput and power management performance reported in all FF configurations.

Exemple graphs showing communities for the AF and FF models captured at simulation time equal to 800 seconds are provided in Figure 8.10 and Figure 8.11.

8.4 Power Management with Continuous Environmental Change

The previous experiments assumed that the demand intensity was varied only at certain simulation periods between which it remained static and fixed. This allowed agents to converge into the organisation that was predefined by the step function and was perturbed only through new consumer agent before another demand change was introduced. In this section, we explore the efficiency of decentralised power management in conditions where the transition between different demand configurations occurs in a smoother and more continuous fashion and is based on a sinusoidal function that facilitates the gradual turnover of consumer agents and thus the dynamic demand intensity change.

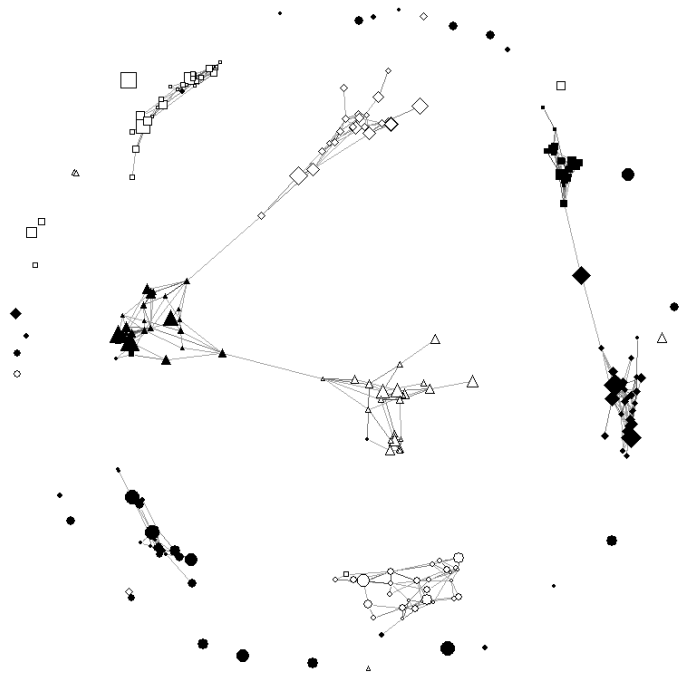

FIGURE 8.10: Correctly organised consumer agent communities extracted from AF model for conditions where consumer turnover probability is equal 0.1 ($\chi = 0.1$). Edges between nodes represent information exchanges between consumer agents where node shapes correspond to specific service type the consumer is required to allocate. The size of each node indicates the amount of resource capacity that is currently demanded by the task.

$\lvert Capability\alpha_p \rvert$	Set of service types ($Capability_{\alpha_p}$)	Demand-supply ratio	Number of Consumers	Number of Providers
8	$\{A, B, C, D, E, F, G, H\}$	varies	250	60

TABLE 8.2: The demand intensity configuration for the sinusoidal demand function. In here, depending on the sinusoidal demand function configuration, the demand intensity oscillates between $0 - 1$ demand-supply ratio range.

The detailed explanation of how turnover of consumer agents is regulated by the sinusoidal function is provided in Section 7.5.1, where the sinusoidal function was applied to define the dynamic service type environment. Since in this chapter we are investigating power management, the sinusoidal function has been adapted towards this purpose in the following manner.

Throughput the simulation time, the key role of the sinusoidal function is to probabilistically influence the number of consumer agents that actively allocate services within the system, such that the overall system demand varies in a continuous manner over the simulation time. During this gradual change in demand intensity, it is assumed that the set of unique service types ($\lvert Capability\alpha_p \rvert$) demanded by consumer agents remains unchanged over the simulation time and the system is perturbed through the consumer

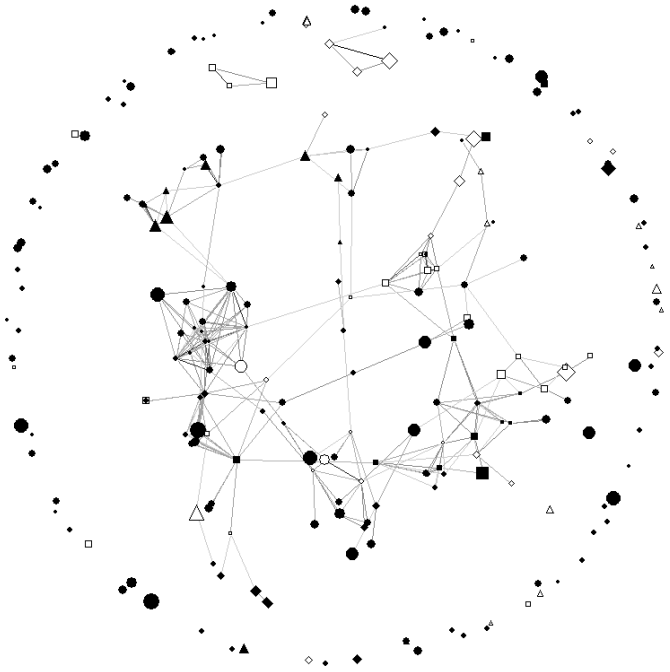

FIGURE 8.11: Disorganised consumer agent communities extracted from FF model for conditions where consumer turnover probability is equal 0.1 ($\chi = 0.1$). Edges between nodes represent information exchanges between consumer agents where node shapes correspond to specific (denoted by node label) service type the consumer is required to allocate. The size of each node indicates the amount of resource capacity that is currently demanded by the task.

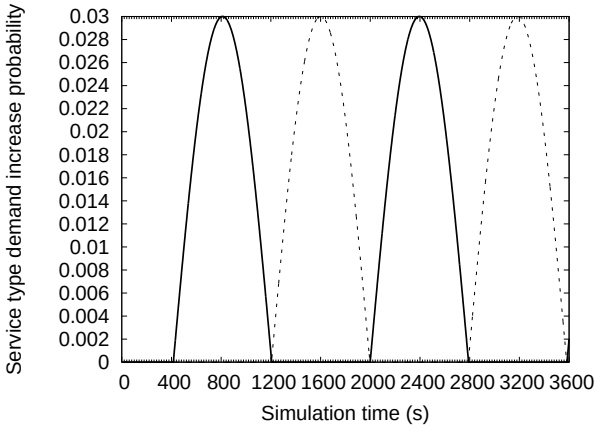

FIGURE 8.12: Figure illustrates sinusoidal function according to which demand intensity within the system varies proportionately to the probability defined on Y axis. Here, the function period is set to $\Xi = 2.2$ where the maximum probability value the function achieves is $\Theta = 0.03$.

turnover mechanism that introduces local demand perturbations.

As in the experiments applying the sinusoidal function to influence the change of service type demand (Section 7.5.1), here we also assume that in each experiment the system experiences a period of stability for the first 400 simulation seconds. In this period there is no demand influence from a sinusoidal function, demand-supply ratio remains unchanged at 0.5, and the system is prone only to perturbations introduced by consumer agent influx. However, after the initial $400s$, the sinusoidal function triggers a change in the number of resource-allocating consumer agents that lead to the demand intensity variation and thus require the population of providers to respond appropriately by adjusting the amount of on-line offered resources. An example sinusoidal function according to which such change is triggered is illustrated in Figure 8.12, whereas the general model configuration for this set of experiments is outlined in Table 8.2.

Given such a demand setup, the behaviour of the AF, NF and FF models was evaluated under conditions in which the sinusoidal function period (Ξ) is varied (thus defining the length in time of each phase duration) along with consumer turnover probability (χ), defining the degree of system openness to the introduction of new agents.

8.4.1 On-line Resource Market Adaptation in Continuously Changing Environment

The power management efficiency for single runs of the AF and NF models in different sinusoidal function period configurations are presented in Figure 8.13 (for AF model) and Figure 8.14 (for NF model). Each figure presents four sub-figures, each for a different sinusoidal function period configuration. The X axis on every sub-figure denotes the simulation time that lasts 3600 simulation seconds in total, whereas the Y axis defines the amount of resource capacity offered (and demanded) by the system. The power management efficiency in every sub-figure is represented by the analysis of three measures: sinusoidally varying consumer demand (dotted line), total on-line capacity offered by providers (bold dashed line) and the available on-line resource capacity (solid line). The periods according to which the demand changes in each sub-figure are incremented in a clock-wise manner, starting from the top-left sub-figure, and have the following values: $\Xi = 1.1, 2.2, 4.4, 8.8$.

The results show that the best resource market adaptation to the changing demand intensity is achieved by AF model configurations. However, as can be seen, for relatively small periods ($\Xi = 1.1$), the system has a tendency to equilibrate between small but frequent demand intensity changes. As the demand intensity varies to the greater extremes (for $\Xi > 2.2$) the resource market reconfigures its on-line resources in a more accurate and efficient manner that almost perfectly matches the changes within the demand intensity imposed by consumer agents.

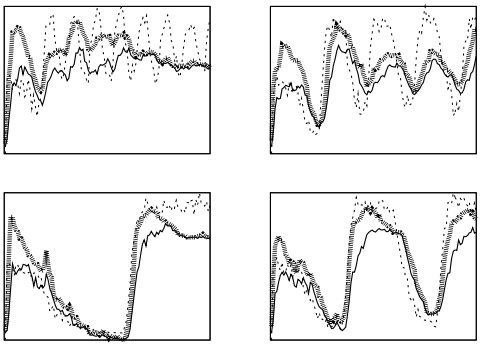

FIGURE 8.13: Figures (organised in a clockwise manner, starting from a top-left corner) illustrate power management efficiency for AF model for model configuration where sinusoidal function period (Ξ) has following values: $1.1, 2.2, 4.4, 8.8$. Each figure illustrates the total amount of capacity demanded by consumer agents (dotted line), available on-line capacity offered by provider agents (solid line) and the total on-line capacity (bold dashed line). All results are obtained from open system model where consumer turnover probability equals 0.1 ($\chi = 0.1$)

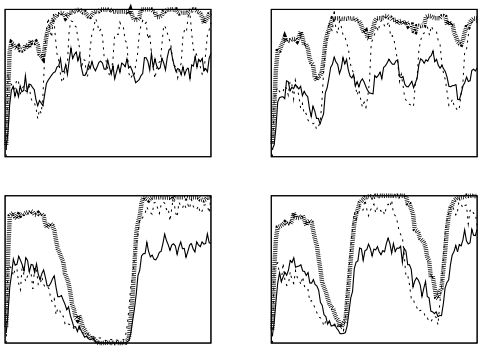

FIGURE 8.14: Figures (organised in a clockwise manner, starting from a top-left corner) illustrate power management efficiency for NF model for model configuration where sinusoidal function period (Ξ) has following values: $1.1, 2.2, 4.4, 8.8$. Each figure illustrates the total amount of capacity demanded by consumer agents (dotted line), available on-line capacity offered by provider agents (solid line) and the total on-line capacity (bold dashed line). All results are obtained from open system model where consumer turnover probability equals 0.1 ($\chi = 0.1$)

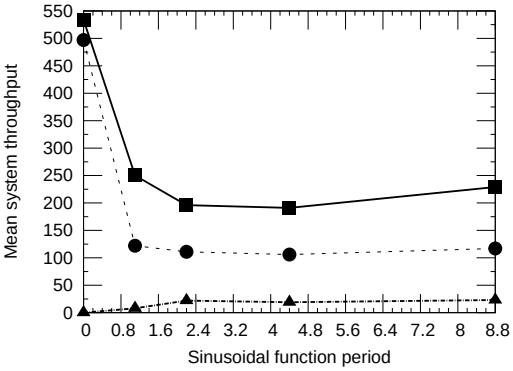

FIGURE 8.15: Mean system throughput as a function of changing sinusoidal demand function period (Ξ). Results summarise performance of three model configurations: AF (rectangles), NF (circles) and FF (triangles). In all experiments the maximum consumer turnover probability is set to 0.1 ($\chi = 0.1$).

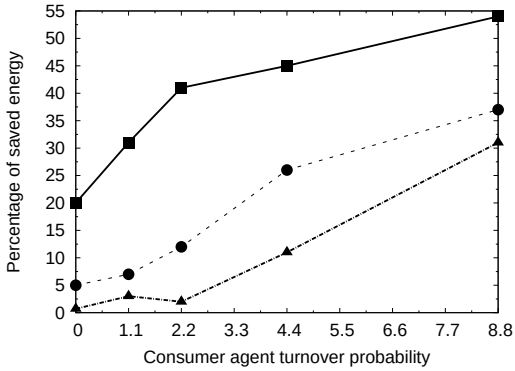

FIGURE 8.16: Percentage of saved energy by providers in off-line mode (as compared to the system configuration where all providers are online) as a function of increasing sinusoidal function period (Ξ). Results summarise performance of three model configurations: AF (rectangles), NF (circles) and FF (triangles). In all experiments the maximum consumer turnover probability is set to 0.1 ($\chi = 0.1$).

As Figure 8.14 shows, the inefficiency of the NF model in achieving proper power management stems from two reasons. Firstly, the resource market becomes frustrated (especially for conditions in which demand intensity varies to a greater extent ($\Xi > 2.2$)) and the amount of offered resources is insufficient to satisfy consumer demand. As a result of this, consumers that are unable to identify the available providers tend to encourage most of the providers to remain on-line and thus prevent efficient power management. Unfortunately, even this surplus of providers that are kept on-line is unable to offer demanded services as they become frustrated by consumers and continually reconfigure to offer different services types. These observations are confirmed by mean system throughput and power management efficiency analyses for the three models in conditions where the sinusoidal function period was varied and the system was open. Both results are presented in Figure 8.15 (showing mean system throughput) and Figure 8.16 (providing mean power management efficiency results).

The results show that the best performance is obtained by the AF model, followed by the NF model and finally the FF model, exhibiting very poor behaviour in every experiment. The highest allocative throughput for the AF model is obtained in situations where demand does not oscillate but is kept constant at the highest level (demand-supply equals 1) and is illustrated in Figure 8.15 for $\Xi = 0$. This results simply from the fact that throughout the whole experiment duration, demand intensity does not change and is kept at maximum, so the system operates at a full throttle, resulting in the highest allocative throughput. The performance of the NF model, for the same demand conditions ($\Xi = 0$) is degraded as a result of consumer agents being unable (relying only on their personal knowledge) to fully stabilise the resource market.

The analogous analysis of individual system runs for the AF and NF models to the one that has been presented at the beginning of this section, but now focusing on the power management efficiency in conditions where the system 'openness' is varied, is presented in Figure 8.17 (for AF model) and Figure 8.18 (for NF model). Here, the same superiority of the AF power management performance over the NF response is observed where the more accurate and responsive adaptation of on-line resources is observed for the former. These results are also confirmed by a more general analysis of mean system throughput (Figure 8.19) and power management efficiency (Figure 8.20) for increasing consumer turnover probabilities.

These results suggest the existence of a certain trade-off between power management efficiency and the overall system allocative throughput. A system that encourages more resources to be on-line is capable of providing resources in a quicker fashion. However, at the same time it introduces additional power management costs involving increased power consumption costs by maintaining a greater number of on-line provider agents than necessary.

Given this consideration, it is interesting to observe that only the AF model is capable to

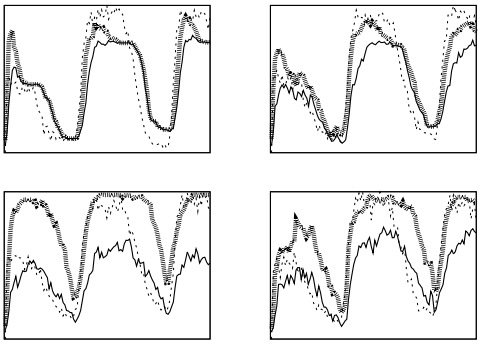

FIGURE 8.17: Figures (organised in a clockwise manner, starting from a top-left corner) illustrate power management efficiency for AF model for model configuration where consumer turnover probability (χ) has following values: $0.0, 0.1, 0.2, 0.3$. Each figure illustrates the total amount of capacity demanded by consumer agents (dotted line), available on-line capacity offered by provider agents (solid line) and the total on-line capacity (bold dashed line). All results are obtained from open system model where sinusoidal function period is set to 4.4 ($\Xi = 4.4$).

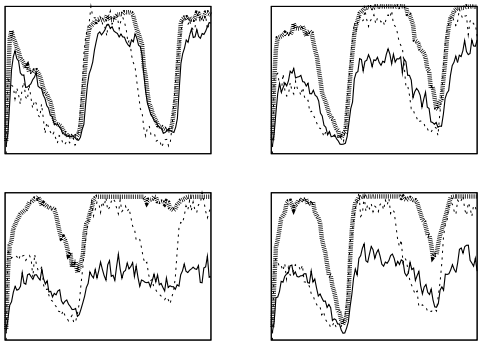

FIGURE 8.18: Figures (organised in a clockwise manner, starting from a top-left corner) illustrate power management efficiency for NF model for model configuration where consumer turnover probability (χ) has following values: $0.0, 0.1, 0.2, 0.3$. Each figure illustrates the total amount of capacity demanded by consumer agents (dotted line), available on-line capacity offered by provider agents (solid line) and the total on-line capacity (bold dashed line). All results are obtained from open system model where sinusoidal function period is set to 4.4 ($\Xi = 4.4$).

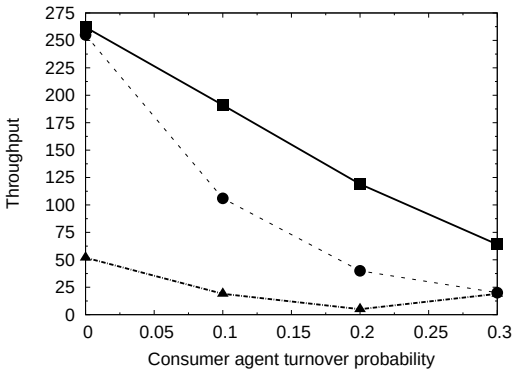

FIGURE 8.19: Mean system throughput as a function of increasing consumer turnover probability (χ). Results summarise performance of three model configurations: AF (rectangles), NF (circles) and FF (triangles). In all experiments sinusoidal function period is set to 4.4 ($\Xi = 4.4$).

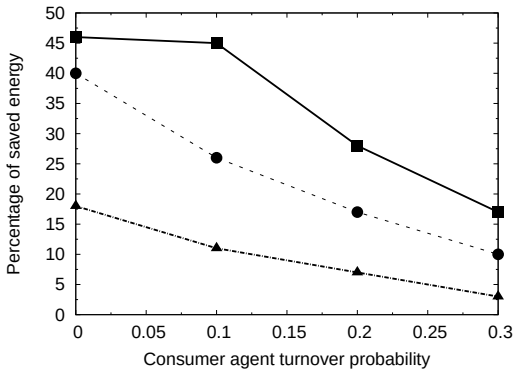

FIGURE 8.20: Percentage of saved energy by providers in off-line mode (as compared to the system configuration where all providers are online) as a function of increasing consumer turnover probability (χ). Results summarise performance of three model configurations: AF (rectangles), NF (circles) and FF (triangles). In all experiments sinusoidal function period is set to 4.4 ($\Xi = 4.4$).

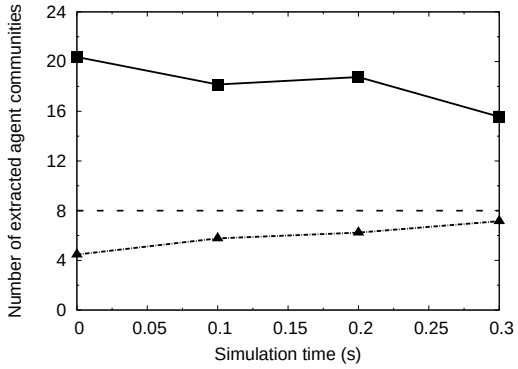

FIGURE 8.21: Mean community homogeneity for AF model configuration (solid line) and FF model configuration (dotted line) as a function of increasing consumer turnover (χ). In all experiments the value of sinusoidal function period is equal to 4.4 ($\Xi = 4.4$).

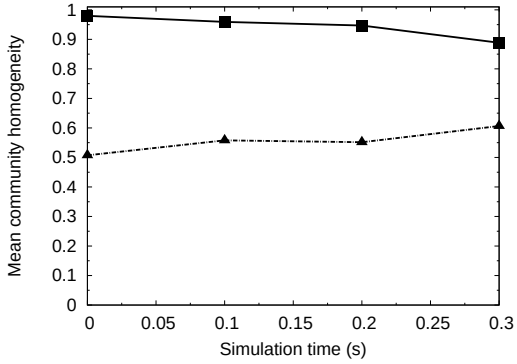

FIGURE 8.22: Mean community homogeneity for AF model configuration (solid line) and FF model configuration (dotted line) as a function of increasing consumer agent turnover probability (Θ). In all experiments the value of sinusoidal function period is equal to 4.4 ($\Xi = 4.4$).

achieve an efficient, equilibrating properties that tend to regulate the amount of on-line resources appropriately to the changing demand conditions and without the negative impact on the system performance.

8.4.2 Consumer Agent Communities

The efficiency of the AF model is also confirmed by the community analysis illustrated in Figure 8.21, showing the number of extracted communities as a function of consumer turnover probability, and Figure 8.22 showing the homogeneity value for these communities. Both figures show that only the AF model of these involving information exchange is capable of organising into distinct communities characterised by high homogeneity, and thus represent collectives of agents that reliably and cooperatively allocate and provide distinct types of resources.

Exemple graphs showing communities for the AF and FF models captured at simulation time equal to 2040 seconds, and where $\Xi = 4.4$, are provided in Figure 8.23 and Figure 8.24.

8.5 Conclusions

In this chapter we have proposed a set of local decision-making algorithms tasked to facilitate power management functionality within a system comprising a population of resource-consuming and resource-providing agents. The efficiency of these mechanisms

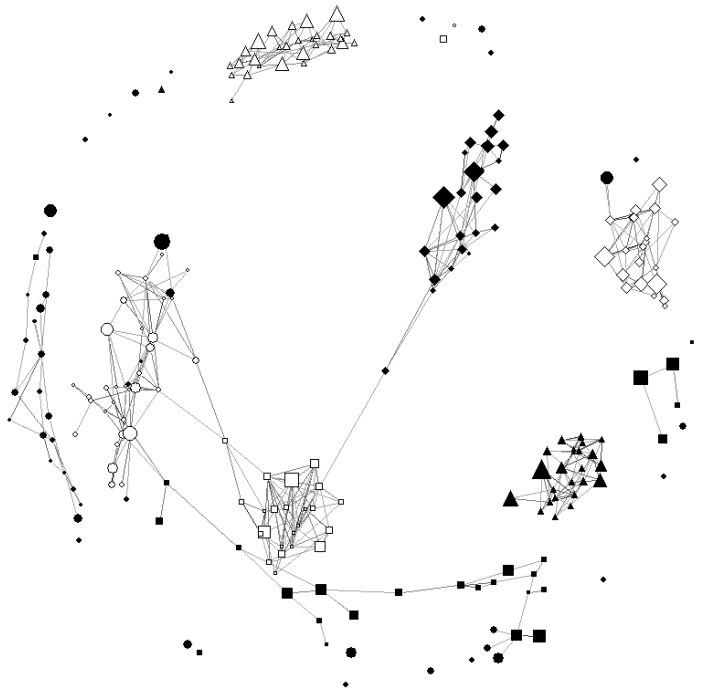

FIGURE 8.23: Correctly organised consumer agent communities extracted from AF model for conditions where consumer turnover probability is equal 0.1 ($\chi = 0.1$). Edges between nodes represent information exchanges between consumer agents where node shapes correspond to specific service type the consumer is required to allocate. The size of each node indicates the amount of resource capacity that is currently demanded by the task.

was then evaluated in a range of dynamic demand intensity conditions including step and sinusoidal demand functions. The results obtained show that the system is capable of achieving adaptive power management despite having no central or hierarchical control nor access to the full information about providers state and current system demand.

These results suggest also that load-balancing, adaptive service provisioning and power management are interrelated and may affect each other's performance in a complex and difficult to predict manner. For example, the FF model performance was poor as a result of the inability of agents to establish coherent and homogeneous communities due to a disorganised flow of information. As a result, not only is the model unable to perform adaptive service provision (as observed in a series of experiments in Chapter 7) but it also incurs an excessive (almost maximal in the case of experiments where the step function was applied) use of system resources when this could be avoided, thus incurring the additional power management cost. For this as well as the NF model in open system conditions, the inability to perform adaptive service provisioning functionality is a direct cause that prevents the system from achieving adaptive power management.

The interrelation between the efficiency of load-balancing, adaptive service provisioning and power management implies that their adaptive maintenance cannot be considered

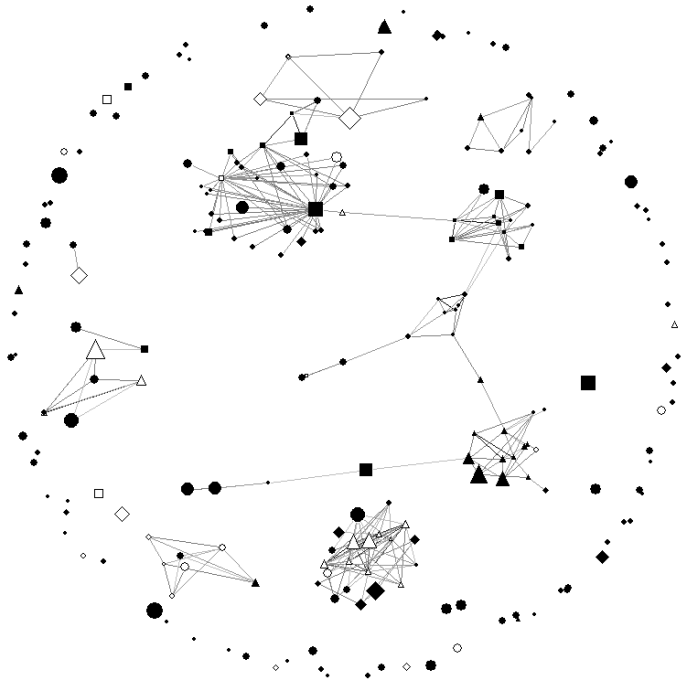

FIGURE 8.24: Disorganised consumer agent communities extracted from FF model for conditions where consumer turnover probability is equal 0.1 ($\chi = 0.1$). Edges between nodes represent information exchanges between consumer agents where node shapes correspond to specific (denoted by node label) service type the consumer is required to allocate. The size of each node indicates the amount of resource capacity that is currently demanded by the task.

as a set of distinct, unrelated, problems but requires a coherent and integral solution. Consequently, preserving these three functions within a large and dynamic system may become challenging and require to an understanding of the complexity of interdependencies between each such function and system dynamics. Although the experimental evaluation provided in this chapter does not explore this problem to a great extent, it shows that the proposed local decision-making mechanisms offer potential in facilitating an integral, yet scaleable and robust, solution that achieves both adaptive service provisioning (with load-balancing) and power management at the same time. However, more experimental evaluation is required to identify to what degree distinct system level functions affect each other and how varying demand conditions impact on such a relationship.

8.6 Summary

Throughout the last three chapters we have focused on the provision of three different functionalities that are expected from large scale computational systems: load-balancing, adaptive service provision and power management. With decentralised multi-agent system models and simple local decision-making algorithms, we have shown that, for each

such functionality, a stable and adaptive system response can be facilitated with no recourse to centralised control nor access to global information about the current system state.

The efficiency of such a decentralised approach is the result of a collective response of locally interacting agents that, under certain model parametrisations, exhibit self-organising properties. These model configurations enable agents to establish self-sustaining flows of information (captured as distinct communities) that influence their future interactions such that a resource market is able to adaptively adjust to the changing demand intensity level.

So far, in all evaluated model configurations we assumed that agents were implicitly cooperating by willingly revealing (if employing AF or FF strategy) information to other peers. Whilst in this work we assumed that the whole infrastructure is maintained by the single provider and thus agent cooperation is a rational strategy, there may exist open computational environments where agents are no longer under the control of a single entity. In these situations one of the possible extensions to the empirical experiments conducted in this work would be to have agents that do not cooperate eg., lie or provide incorrect information to others and to analyse impact of such defective behaviour on the stability of the system and performance of individual agents.

Whilst in the last three chapters we considered load-balancing, adaptive service provisioning and power management as three separate challenges, the obtained experimental results show clear relationship between these tasks. This suggests that decision-control mechanisms that we have applied in separate chapters can be combined in the form of a single utility function. Such an approach would allow us to constuct a single but multi-objective function within which different aspects of resource management could be integrated and considered as a single decision-making mechanism, thus simplifying the engineering process of the system. To give an example, in order to achieve adaptive service provisioning and power management we have provided two similar threshold based functions and considered them as separate solutions to a particular problem. However, in principle, these functions can be easily integrated such that 'off-line' movement of the node is yet another 'service type'. Although moving off-line is regulated through slightly different threshold adjusting function than switching to a different service type, both actions could be merged to form a multi-objective function.

As in the absence of any central 'intelligence' governing and regulating the behaviour of the system, it is mandatory to understand how *stability* and *adaptation* arises within a population of autonomously interacting agents. So far, the last three chapters identified which model parametrisations were efficient at achieving expected system functionalities.

However, as our main goal is to be able to engineer these kinds of systems in a principled manner, in the next chapter we focus explicitly on understanding conditions and properties of local decision-making mechanisms that achieve this. This analysis will be

provided within the thermodynamic framework, primarily used for understanding and exploiting self-organisation within natural systems that we will extend to comprehend socio-technological systems as well.

Chapter 9

Thermodynamic Interpretation

9.1 Introduction

Engineering of systems that exploit laws of physics and extract work based on thermodynamic principles that underly the phenomenon of self-organisation has been approached almost 200 hundred years ago when Nicolas Sadi Carnot (in 1824) developed the model of a heat engine [52]. Initiated by him understanding how energy flow through a physical system can be transformed into a useful work allowed for the construction of man-made systems such as thermodynamic engines that underpin the supply of most of the world's electric power and almost all motor vehicles nowadays.

Considering the impact of these findings on the development of human society, it is not surprising that the principles derived from thermodynamics of self-organisation have become increasingly important in studying and understanding complex self-organising phenomena observed both in physical as well as biological world.

Whilst much is still to be understood in relation to the role the thermodynamics plays at producing order and life in particular [54], initial models how it may be applied to study and understand the self-organising properties of biological, decentralised and information-driven systems have already been provided [24, 29]. However, as these studies show, the problem of engineering computational systems that exploit laws of physics is much harder than the one addressed by Nicolas Sadi Carnot and his followers. In particular, much remains to be understood as to what exact conditions are needed for self-organisation to arise and what decentralised mechanisms are required for its effective stabilisation in a bottom-up manner.

In this chapter we will focus our attention on these problems and suggest means how they may be approached whilst engineering self-organising computational systems. To this end, in Section 9.2 we provide a set of design principles that we consider relevant whilst addressing the autonomic system control that is inspired by the phenomenon of

natural self-organisation, whereas Section 9.3 offers a thermodynamic interpretation of such a model functioning.

9.2 Design Principles

As stated earlier in this chapter, systems that exploit thermodynamics to extract work are not novelty and have been engineered since Sadi Carnot provided the first heat engine model in 1824. However, there exist certain differences between these artifacts and modern computational systems that need to be addressed in the first place, before we start applying self-organisation for the purpose of controlling modern IT systems. To this end, we identified two general problems that underly the design of self-organising software systems.

Firstly, Sadi and his followers had a much simpler task at engineering their thermodynamic systems because these artifacts operated in a real world and for this reason conformed to certain physical laws specifying the behaviour of the system (and its elements). These laws and, in particular, the second law of thermodynamics, as we have discussed in Chapter 2.5, play the key role in achieving self-organisation and work extraction from energy driven system.

Unfortunately, the software engineers have no such privilege of relying on the already existing laws, as it is up to them to define individual system behaviours and interactions through algorithm-based rules or policies. This leaves software engineers with a much greater freedom of creating their own *rules of the game* that are tailored for their own design specifics. We find that care must be taken on this level of development as, whilst some of these ad-hoc rules may be consistent with thermodynamics and facilitate self-organisation at the global system level, the others may not.

The second issue is that Sadi and his followers did not have to bother with constructing systems that comprise vast number of elements which need to, somehow, self-organise in order to work efficiently. Rather, he built rigid and mechanistic systems the operation of which was deterministic and had only one purpose: to regulate the energy flow through the system such that a useful work can be extracted from it. During this process, the system structure and possible configurations were pre-imposed at a design time and never required reconfiguration during its life-time. For example, there are no such engines that modify their internal structure such as displacement or piston size as the engine operates. With respect to this, such thermodynamic engines are crude and simplistic artifacts when compared to the highly dynamic and sophisticated software systems.

This makes life harder for software engineers aiming to build their systems relying on thermodynamic principles as here the thermodynamics becomes intertwined with the extensive understanding and application of self-organising processes. As a consequence,

the engineers of technological systems are faced with the problem of designing models that do not rely on static and non-changing configuration of their elements but, in contrast, are required to reconfigure their interactions in order to autonomously discover the ones that maximise the system efficiency given the current system functioning conditions. For this purpose, more in depth analysis of natural systems and processes they employ for self-organisation is needed.

In what follows, based on the study of our self-organising system model, we present one of the possible ways one can undertake in order to engineer a self-organising computational system that is based on thermodynamic principles of self-organisation. In doing so, we will organise the remaining content into two separate parts, each addressing one of the problems that we have breifly outlined above. In the first part, we will start by sharing our experiences of what features of the physical world need to be represented within a computational system, thus forming universal 'rules of the game' that all system elements need to comply with. Here we will present these features and explain how they were incorporated into our model with the help of specific algorithms and decision-making mechanisms.

Once these rules are laid out, we will then move on to the second issue that is how can we engineer system elements that, by following these rules, self-organise their interactions such that a useful work, in accordance to the system objectives, is extracted from the model. Here we will explain the role of self-organising processes observed within natural systems and our local decision-making mechanisms that we applied to incorporate them within our computational system.

9.2.1 Conditions for Self-organisation

Recall the classical self-organisation experiment conducted by Benard over 100 years ago that we briefly described in Chapter 2.5. Here, Benard observed that heating up a viscose fluid contained in a reservoir gave rise to spontaneous organisation of initially disorganised system elements into coherent and self-sustaining columns, referred from then as Benard Cells.

As thermodynamics shows there is nothing 'magical' about such spontaneous organisation and, whilst arising in a complex and emergent manner, it is a consequence of certain conditions that influence the way in which the system responds to the energy or information flows across its constituents [116, 56].

To explain the role of these conditions, consider a general model of a self-organising system illustrated in Figure 9.1. Here, three different levels at which the system can be analysed are provided: micro, meso and macro. Typically, on the micro-level we distinguish individual system elements and their interactions, whereas meso-level illustrates organisation of these elements into certain structures, be they hexagonal cells within

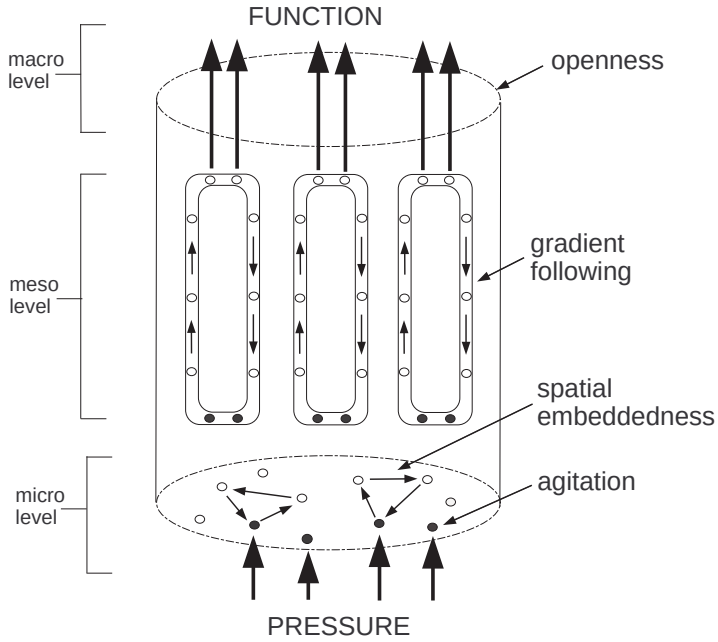

FIGURE 9.1: Self-organisation in natural open systems arises as a result of following conditions: *openness*, *agitation*, *spatial embeddedness* and *gradient following*. Bottom-up organisation in decentralised software systems is dependent on re-interpretation and engineering of these features within a computational environment.

Benard Cells example or foraging trails formed by an ant colony. Finally, at the macro-level we consider the global outcome (or function) of system self-organisation. Within insect society this corresponds to the adaptive division of labour during which different collectives of ants (identified as organised structures at the meso-level) carry out various system level functions (eg., food foraging, brood feeding, nest construction). In the Benard Cells example, on the other hand, it is the most efficient heat transportation to the system surroundings that is achieved through convection.

Given this model, consider its features illustrated in the figure as: *openness*, *agitation*, *spatial embeddedness* and *gradient following*. In what follows we explain their role in achieving natural self-organisation and then provide means how they can be re-interpreted and realised within a software system.

9.2.1.1 Openness and energy flow

According to thermodynamics, self-organisation is directly linked with energy and its flow across an open system. As observed by the Nobel prize winner Ilya Prigogine [56], one of the preconditions for achieving self-organisation within a physical system is *openness* manifested by the ability of the system to accept energy from its surroundings as well as dissipate it outside its boundaries.

Consider the effects of energy inflow observed in the physical system explored by Be-

nard. Here, its supply in the form of a heat agitates system elements such that the overall system becomes more dynamic. Once the inflow of energy stops, its remaining surplus becomes dissipated from the system that eventually moves back to its initial, low-energy state. Similar pattern of behaviour is observed within a more sophisticated living self-organising system such as ant society. Here, rather than directly sensing physical 'energy', ants react to chemical information (pheromones) deposited by their peers within the environment. Once the concentration of such information becomes sufficiently high, ants become agitated and stimulated to perform certain activities, eg., food foraging or nest defense. Accordingly, once such chemical information disappears, the ants become less agitated and either rest or pursue other activity.

Unsurprisingly, openness and information flow have analogous consequences in a computational system and thus do not require additional mechanisms or thermodynamic re-interpretation. Here, we assume that the system is open and fed by allocation requests that originate from users. These requests have the same effect as a supply of heat in a physical system or the perception of a pheromone substance in an ant society, that is, they agitate agents and initialise their resource allocation process where the overall system dynamics increases proportionally to the amount of task requests fed into it. Similar to physical systems, the return of the system back into the less dynamic state takes place when there is no inflow of new tasks and the successful or failed ones become dissipated from the system. At this state, individual agents, alike elements of a physical system, move back to a settled state.

9.2.1.2 Agitation

Self-organisation within Benard Cells example arises as a result of local interactions between system elements. The frequency and extent to which these elements interact is proportional to the agitation level they perceive as a result of heat supply to the system. The more heat is pumped into it, the more frequent and dynamic such interactions become and the more stressed the system becomes as a result of surplus of energy that is injected into it.

Consequently, this suggests that the system elements are provided with mechanisms that allow them to perceive and respond to the stress manifested by the inflow of energy. This, as discussed in Chapter 2.4, has an important role in achieving system organisation since the increasing level of stress facilitates faster exploration of possible system element configurations, once the most efficient ones (given system objective) are discovered and sustained.

To reflect this within our computational model, we provided each software agent with a local measure of stress that, for consumer agents defined how quickly they could allocate requested tasks and for provider agents how reliably they were offering demanded services

types. In both cases, the increase in stress level indicated that the system element become more agitated (eg., consumers experienced more unsuccessful interactions with providers whereas providers reconfigured their service provision frequently). Defined in this manner level of stress was then shared among peers and was used by consumers to prevent interaction and information flow between highly stressed providers. As a result of this, the agents were able to detect arising system instability in the form of highly stressed agents and to reorganise their interactions. During this process, new and more efficient system elements configurations were explored until the most efficient one was found.

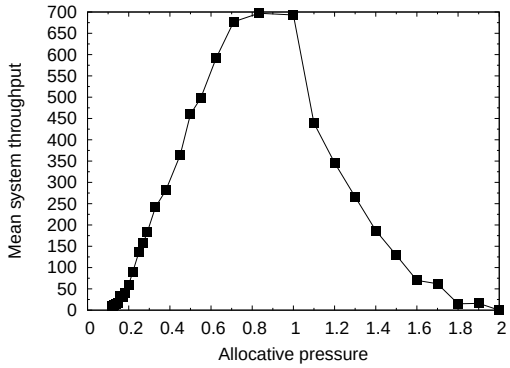

FIGURE 9.2: Mean system throughput as a function of increasing allocative pressure (ν) applied to the model. The pressure is reflected by the shorter ω time intervals that define probabilistic (Poisson based) task time arrival to the system. For presented results $\omega \in \langle 40s, .., 5s \rangle$.

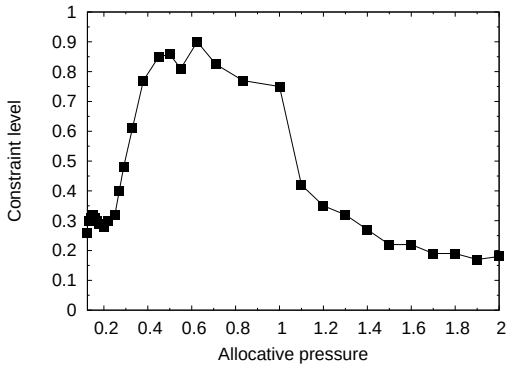

FIGURE 9.3: Level of mean constraint measured for consumer population (empty rectangles) and provider population (solid rectangles) as a function of increasing allocative pressure. The pressure is reflected by the shorter ω time intervals that define probabilistic (Poisson based) task time arrival to the system. For presented results $\omega \in \langle 40s, .., 5s \rangle$.

As suggested earlier, the level of stress that the individual system elements experience is dependent on both its ability to discover efficient organisation that allows them to 'dissipate' successful tasks from the system at a faster rate, and the pressure the system has to cope against, here represented by the amount of arriving to the system tasks in a given time interval. Results showing the self-organising response to such systemic pressure is illustrated in Figure 9.2. Here, the frequency of tasks arrival (depicted on X axis) to the system is gradually increased. In situations when it is equal 1, the state where demand-supply ration is equal $1 : 1$, whereas further increase generates conditions in which demand outstrips supply, eventually reaching demand-supply ratio equal $2 : 1$. Interestingly, as the level of agent constraint illustrated in Figure 9.3 shows, the system self-organises only once the systemic pressure is high enough (greater than 0.3) [1]. This can be understood by the fact that for low demand intensity there is no

[1] Provider constraint level defines how reliably provider agents offer the same service type over a period of time. Where constraint equal 1 suggests that providers will never deviate from offering the

need for organisation of agents into communities, as there are enough resources to satisfy the small quantity of demanded tasks in a timely manner. For conditions when demand outstrips supply, on the other hand, the system is unable to maintain self-organised state as the level of stress experienced by agents is too high and there exists no configuration able to suppress it.

9.2.1.3 Spatial Embeddedness

The interactions between system constituents in natural systems are defined by spatial proximity that exists between them. As a result, elements that are closer to each other are more likely to interact and affect each other behaviour than the elements that are located at a greater distance. Consequently, any local differences in the energy or information that arise within a certan system region become gradually diffused to the neighbouring system elements before they have a chance to impact on the further regions of the system.

Existence of such spatial embeddedness is considered as another property of natural systems that plays important role during their self-organisation. In particular, it has been observed that the ability to affect the state of only neighbouring system elements facilitates the necessary conditions for the emergence of element organisations that are resistant to perturbations and instabilities occurring within other system regions. For example, the studies focusing on the division of labour within insect societies [29] suggest that deposition of pheromones within a certain locations of the system (eg., foraging trail) and their gradual decay in other system regions allow for sufficient amount of ants to be attracted to carry out food transportation, but prevent destabilisation of tasks performed by other ants that are at distant locations from pheromones that attract for food foraging task. As these models show, the removal of such spatial situatedness, allowing thus all system elements to perceive pheromones that are propagated to the environment, prevents the system organisation to take place as the system elements become confused and distracted which tasks to carry on.

Whilst natural systems have spatial embeddedness incorporated 'by default', no universal law or principle exists within software systems that defines spatial proximity between individual agents. As a consequence, there are no restrictions imposed on what system elements are allowed to interact or how the outcome of such interactions propagates through the system. This, as suggested by Guerin in [29], may prevent any system organisation to take place. Consequently, we consider decision-making mechanisms that facilitate spatial embeddedness as an important part of self-organising software systems design.

---

same service, 0 means that no constraint exists and the provider will choose to offer any service type with equal probability. The mechanism for calculating constraint level is discussed in Section 5.5.2.1.

For this purpose we introduced *affinity* algorithm the role of which was to facilitate such spatial embeddedness in the form of interaction topologies allowing agents to distinguish their local neighbours from further located peers. Formed in this manner topologies provided a 'virtual' proximity between other peers and incentivised the agents to interact with the ones that are within their close vicinity. As a result of this, within a system of interacting agents we were able to observe emergence of distinct agent communities that were sustained through local interactions and information exchanges.

As the experimental model evaluation provided in Chapters 6, 7 and 8 showed both interaction topologies as well as formed on top of them agent communities were critical for achieving organised and stable system response.

9.2.1.4 Gradient Following

As provided in Chapter 2.5 discussing the thermodynamics of self-organisation, all physical systems obey the second law of thermodynamics. This means that all differences and disparities (eg., temperature, pressure, chemical or electrical potential) that arise between the system and its environment or among the system elements become extinguished over time, thus bringing the system back to the state of equilibrium.

In natural systems such equilibrating force is 'given for free'. However, within the software system environment there is no such property and it is up to the system engineer to incorporate it. How is this realised in our computational, information driven system?

Recall that all consumer agents are designed to perceive, process and communicate information based on which they conduct their local actions such as provider selection (consumer agents) or service type configuration (provider agents). In the case of consumer agents, the information that is communicated between them is represented within each agent in the form of a local registry maintaining the list of known providers and evaluation scores reflecting their efficiency at providing services. Such locally maintained knowledge is employed by each consumer during provider selection, where a roulette wheel mechanism is used in order to select the best (according to the evaluation score) provider. We can consider such registry as an *informational gradient*, where providers with higher evaluations have a greater chance of being selected than the ones with lower scores.

The exemplary representation of such a gradient obtained from three different model configurations (AF, NF and FF) are illustrated in Figure 9.4. Here, the figure illustrates values of preferences associated with selection of 20 providers (sorted in a descending evaluation score order) for three different model configurations: AF model (solid line), NF model (dotted line) and FF model (dashed line). The best gradient presented in this figure is achieved by agents exchanging information amongst their local neighbours and thus utilising for this purpose interaction network topology. In this case,

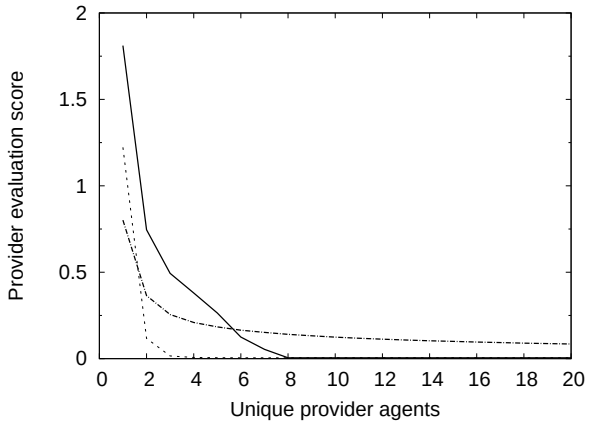

FIGURE 9.4: Informational gradient formed by the evaluation scores associated with selection of particular provider agents. In here **20** agents are illustrated in a descending evaluation scores order.

the gradient correctly identifies six provider agents that are available and configured appropriately for the service type these agents require. The most inefficient informational gradient configuration is obtained by FF model where lack of underlying interaction topology and thus the free flow of information across the system confuses provider agent population and thus consumers are unable to sustain and concentrate their selection towards a particular subset of resources. Consequently, the arising gradient is flat and does not guide consumer agent selection process in any efficient way.

Given such a gradient, we can interpret it as a local indicator that reflects how far the agent is displaced from equilibrium and, depending on this, how its behaviour becomes affected by such displacement. In this context, the disorganised and equilibrated state is reflected when the informational gradient is flat and thus agent has no constraint imposed on the selection of any provider. Under these conditions the selection of any provider is equally probable and we assume that the agent behaves analogously to the agitated particle within Benard Cells example that tends to get rid of the excessive heat (allocative task in our case) through inefficient and chaotic collisions with other particles (provider agents in our model). However, if the shape of such informational gradient is non-uniform, meaning that some providers have a greater chance of being selected, we consider the agent to be displaced from equilibrium as the constraint on the selection of particular providers has been imposed. In this situation the agent aims to 'dissipate' such gradient by probabilistically selecting the best provider for its task allocation. During this process the agent successfully employs the provider and, by doing so, consumes its resources for a limited period of time. At this state the level of agent stress (or otherwise agitation) decreases as a result of successful allocation during which useful work was extracted from the system.

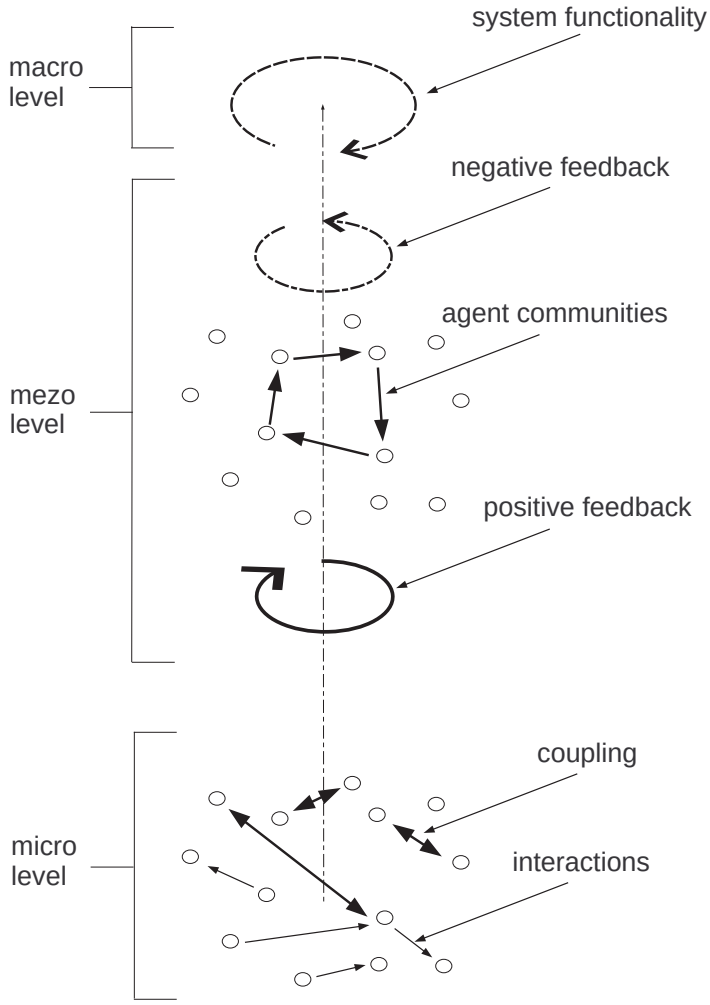

FIGURE 9.5: Facilitation of global system functionality such as load-balancing, adaptive service provisioning or power management is achieved through self-organising agent communities. Formation and stabilisation of these communities is achieved through local decision-making mechanisms that give rise to *coupling*, *positive feedback* and *negative feedback*.

9.2.2 Mechanisms for Self-organisation

Bottom-up self-organisation arises and is sustained as a result of local interactions. Using network analysis tools, we captured these interactions as distinct agent communities that represent organised global system structures.

As throughout experimental evaluation we have already observed that there exists only a subset of interactions (and decision-control mechansims) contributing towards the system organisation into the agent communities, in what follows we will outline key processes responsible for the *formation*, *maintenance* and *reconfiguration* of such dynamic structures. Whilst keeping this description on an abstract level, we will exemplify the role and realisation of these processes within a computational system by referring to our multi-agent system model design.

9.2.2.1 Interactions

The first precondition to achieve self-organisation is the existence of interacting and autonomous processes. Autonomy in this context allows individual elements to diversify and adjust their possible set of actions in response to changing system conditions that would not be possible if simple software objects were considered. Interactions, on the other hand, allow single processes to influence the behaviour of other elements. Given these requirements, decentralised multi-agent systems comprising a population of autonomous and interacting agents are suitable models for this kind of computation.

In Figure 9.5 we illustrate such autonomous processes at micro-level of the system as empty circles and interactions between them as one-directed arrows drawn between the circles.

9.2.2.2 Agent Co-adaptation Through Coupling

Achieving organised and coherent state within a population of interacting and autonomous elements requires their individual behaviour adjustment, often redescribed in terms of a co-adaptation process. For this to take place, individual elements need to be able to sense and respond to changes occurring within their local environment that are inflicted by other elements. Ability to do so requires system individuals to exhibit *coupling*.

Coupling here is an emergent property and arises when the behaviour of a subset of unrelated system elements becomes interdependent to each other, such that a change in one elements' behaviour inflicts the change in the other one (or others). For example, in our model the coupling can be captured by the co-adaptation between consumer and provider agents arising as a result of persistent interactions between each other. The provider, as a result of consumer interactions reconfigures to offer the service that is demanded by the consumer. The consumer, on the other hand, establishes a preference towards that provider and thus continues to exploit its resources. As a consequence, the configuration of both agents becomes interdependent: provider keeps its current configuration and consumer sustains the current attraction towards that provider. However, as soon as this coupling is broken (eg. provider has not enough resources) the consumer will seek for another provider, whereas the current provider (no longer stimulated by the consumer) will either move into a sleep state or offer different service type. At this state both agents are no longer coupled as they do not influence each other behaviour any more.

The coupling within a multi-agent system can be realised through local decision-making mechanisms that allow agents to detect the interactions that are mutually beneficial and persist at such configurations as long as they contribute towards the welfare of

the involved agents. In our model this local co-adaptation was facilitated through the application of *stimulus-response* mechanisms (described in Chapter 5) that are employed by natural self-organising systems such as insect societies. In Figure 9.5 the coupling between pairs of system elements is illustrated on the micro-level in the form of two-directional arrows drawn between pairs of agents.

9.2.2.3 Community Formation Through Positive Feedback

As discussed in Chapter 2.4, the dynamics of a natural self-organising system is typically non-linear because of circular or feedback relations between the components. Such non-linearity can be understood from the relation of feedback that holds between the system's components, where each component affects other components, but these components in turn affect the first component. Thus the cause-and-effect relation is circular and any change in the first component is fed back via its effects on the other components to the first component itself. Such behaviour is often re-described in terms of autocatalytic potential suggesting the ability of the system to facilitate explosive growth of particular system configurations [11, 91].

This progressive aspect of self-organising systems has been identified by cybernetics in one of their fundamental principles [36] referred to as 'The Principle of Autocatalytic Growth' which states that 'stable configurations that facilitate the appearance of configurations similar to themselves will become more numerous'.

In our model the positive feedback is employed during agent community formation process that arises not as a result of sophisticated AI planning but non-linear interactions between system peers. To realise this, a combination of the following mechanisms is employed: *affinity algorithm* (described in Section 5.3.3.6), information exchange (described in Section 5.3.3) and *stimulus-response* learning (described in Section 5.3.1) influencing the emergence of coupling.

To understand how agent community arises out of responses of individual agents, consider example community formation scenario. Here, emergence of the community and the relevance of the above defined mechanisms is illustrated based on the four following figures, Figure 9.6, Figure 9.7, Figure 9.8 and Figure 9.9.

The first figure (Figure 9.6) illustrates system configuration in which three consumer agents ($C1$,$C2$ and $C3$), all interested in the allocation in the same service type, consume resources offered by three provider agents ($P1$,$P2$ and $P3$). Initially neither of consumers belongs to a community and each allocates resources independently. At this state we assume that consumer-provider pairs are coupled, meaning that each consumer established attraction towards a single provider agent (reflected in the provider evaluation score maintained by consumer within its local registry) and each provider learned to offer the currently demanded service type.

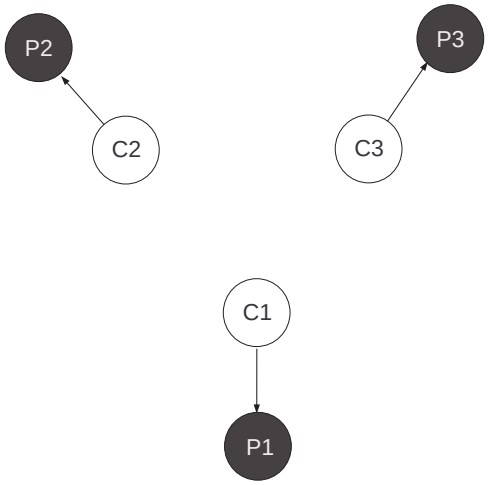

FIGURE 9.6: Community formation through positive feedback. Here, three distinct pairs of consumer-provider agents are identified. Each pair is represented as a coupled set of consumer and provider agents that reliably offer and consume (as denoted by solid arrow) resources.

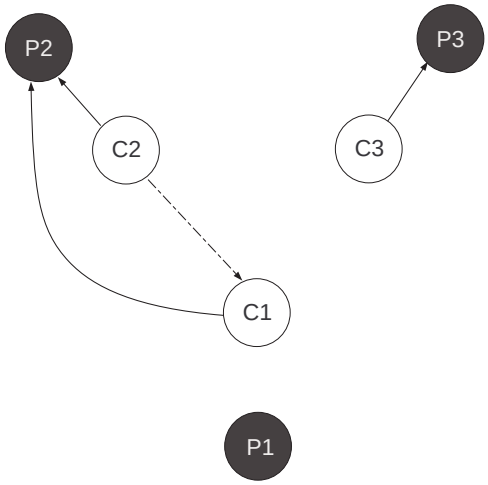

FIGURE 9.7: Community formation through positive feedback. Here, $C1$ consumer allocation request is rejected by $P1$ provider. In response, the consumer identifies and employs $P2$ provider agent. The dashed arrow between $C2$ and $C1$ consumers illustrates information sharing that is mediated between both agents through $P2$ provider agent (for simplicity not shown).

To understand how feedback relation arises between independently acting consumer agents, recall that the life-cycle of every consumer (described in Chapter 5) is followed by two main activities: information exchange (information pull and information push) and resource allocation. During the information pull (preceding resource allocation) the consumer interacts with the provider it is currently co-located with and queries personal registry information from co-located with this agent consumer agents. This action, in the example presented in Figure 9.6, has no result as there exist no other consumers utilising the same provider. During information push, rather than obtaining information from the provider, the consumer reveals its local knowledge to a subset of provider agents that, in turn, disseminate this information to co-located with them consumers. The subset of provider agents to which this information is sent is determined by the affinity scores contained within consumer registry. In the example considered in Figure 9.6 we assume that consumer agents have positive affinity scores only for provider agents they are currently employing and thus, at this stage, the information is communicated only to these agents and reaches no consumer recipients, as there are none co-located with the same provider.

If the system was functioning in deterministic conditions in which consumers would never need to reconfigure their provider selection, described above organisation would

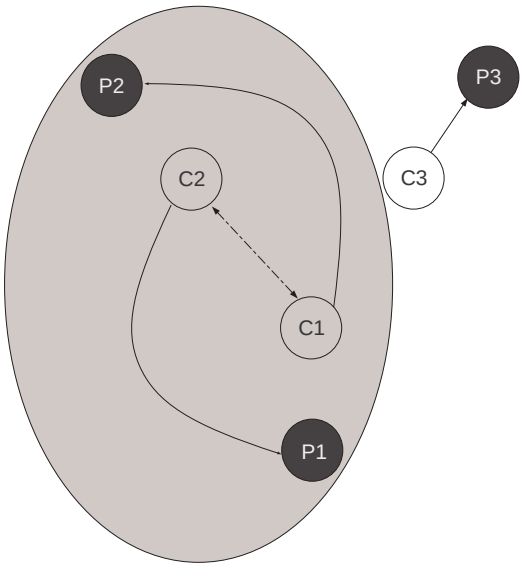

FIGURE 9.8: Community formation through positive feedback. Here, a causal and circular relationship is established between $C1$ and $C2$ consumer agents that both start to share their local provider evaluations between each other (as illustrated by the two-directed dashed arrow). The shaded area represents a subset of consumer agents that form a community as well as resources that are shared and employed by the community members.

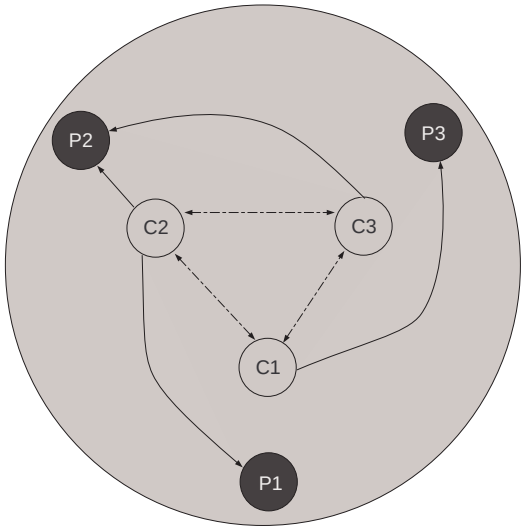

FIGURE 9.9: Community formation through positive feedback. Here, the two-agent community incorporates another consumer agent ($C3$) as well as another resource ($P3$). The dashed arrows illustrate the information flow that is collectively sustained by the community members, whereas the shaded area represents consumer agents that form the community as well as resources that are shared and employed by the community members.

reflect the optimal system configuration. However, the system is dynamic and there are situations where either provider agents may occasionally become unavailable (eg., for short, maintenance period of time) or consumers may be requested to allocate resource capacity that is greater than the previously employed provider could offer.

A situation reflecting this short term instability is illustrated in the next figure (Figure 9.7), where $P1$ provider denies service provision to $C1$ consumer agent. In response to rejected allocation request $C1$ consumer performs random search for other provider agent available to satisfy its request [2] and eventually obtains a positive response from $P2$ provider. Consequently, $C1$ consumer employs $P2$ provider with whom it becomes co-located with as a result of the successful allocation. At this moment $C2$ agent, which is also co-located with $P2$ provider, gossips its local knowledge to $P2$ provider that passes this information to $C1$ agent. In response, $C1$ agent decides whether to accept foreign information (using stress measure defined in Chapter 5.3, Section 5.3.3) and, assuming

[2] Recall that up to this moment $C1$ consumer agent was relying only on $P1$ provider thus all remaining provider agents kept in its local registry had evaluation scores equal 0. As a consequence, after $P1$'s unavailability, the probability of selecting any other provider using roulette selection mechanism was equal.

that provider exhibits low stress, merges the foreign information with its local registry. As an outcome of the information merge process, $C1$ agent updates its evaluation score about $P2$ provider as well as increases affinity score for $P2$ provider in order to reflect inflow of external information from a reliable (non-stressed) source of information.

As a result of $P2$ provider allocation and the information exchange that followed this, $C1$ consumer extended its local knowledge about existing providers (now it has evaluation scores for $P1$ and $P2$) as well as established affinity score to $P2$ provider, meaning that in the next information sharing procedure it will reveal its local registry to both $P1$ and $P2$ agents.

This next step in community formation through positive feedback is illustrated in Figure 9.8. Here, $C1$ consumer after successful resource allocation shares its local provider estimates with providers that have high affinity scores within its local registry ($P1$ and $P2$). Since this information is communicated to $P2$ provider, $C2$ consumer that is co-located with this agent becomes a recipient of this information and follows the same procedure of new information merging as $C1$ agent did in the previous allocation round when obtaining knowledge from $C2$ agent. Assuming that $P1$ agent has low stress estimate and thus $C2$ consumer decides to accept information originating from this agent, the outcome of information exchange and its incorporation in $C2$'s registry is reflected by the appearance of evaluation score for $P1$ provider that is higher than 0, as well as the positive affinity score for $P1$ provider from which this valuable information originated.

After such information exchange an important change takes place that affects the future causal relation between $C1$ and $C2$ agents. Recall that now both consumers have affinity scores for the pair of $P1$ and $P2$ providers and, motivated by these scores, continue to share their local knowledge to these agents and thus indirectly to each other. Here, each information sharing activity performed by one of the consumer catalyses the same reciprocative action by the other consumer. As a result of this self-reinforcing feedback, both agents establish a collective knowledge about provider agents they know about ($P1$ and $P2$) that improves their future service allocation since now, even the occasional failure of one provider would attract them to the second, available and properly configured provider.

From now on, we consider that actions of both $C1$ and $C2$ consumer agents become interdependent as they establish a feedback relation between each other through the indirectly communicated information about resources they know about. Such self-reinforcing flow of information establishes the existence of a community comprising $C1$ and $C2$ consumer agents together with employed by them resources ($P1$ and $P2$) the availability of which is now shared across the community members.

The established flow of information among community members not only sustains the achieved organisation but also has a positive effect on its growth as other consumer

agents that are alike (interested in the allocation of the same service type), and which occasionally employ providers that belong to the established community, become attracted towards this community in the same manner as $C2$ consumer agent was in the previous step. This 'autocatalytic growth' of structure is illustrated in the last figure (Figure 9.9), where $C3$ agent employs $P2$ provider and, as a result, obtains local registry information shared by $C1$ and $C2$ agents. As a consequence, its local registry knowledge becomes extended by additional evaluation scores of $P1$ and $P2$ providers, followed by positive affinity scores update for these agents. Eventually, another agent becomes added to the community together with additional resource ($P3$) that is introduced to the pool of currently shared ones ($P1$ and $P2$).

The above presented community formation process achieved through the positive feedback underlies the formation of all agent communities that we identified during our autonomic system model evaluation in the previous three chapters focusing on the provision of load-balancing (Chapter 6), adaptive service provisioning (Chapter 7) and power management (Chapter 8).

Recall that whilst *stimulus-response* learning techniques allow consumer and provider agents to establish coupling that improves their informed decision-making, it is the information exchange and *affinity* algorithm that enforces stability of such coupled agents and thus eventual community formation observed within our model. Here, *affinity* algorithm facilitates growth of communities and their maintenance by 'consuming' and 'drawing' additional system resources and incorporating them into the pool that is shared among community members only. The demand pressure imposed by such community members minimises the risk of provider agents to deviate from offering the service type that is different from the one demanded by the community. On the other hand, the attraction towards the pool of such providers that is continually shared among community members prevents them from attempting to interact with providers that do not belong to this community and thus destablilise their provisioning.

9.2.2.4 Community Stabilisaton Through Negative Feedback

Recall self-organisation process overview described in Chapter 2.4 where we suggested that the overall system organisation is stabilised by positive and negative feedback loops. Whereas positive feedback, that we discussed in the previous section, is used to enforce structure formation through non-linear causal relations that arise between system elements, the role of the negative feedback is to suppress such dynamic growth before it consumes all system resources and destabilises system functioning.

Such negative effects of applying only positive feedback are exemplified in Figure 9.10 where a system comprising two distinct agent communities (A and B) is shown. Here, each community demands different service type but requires the amount of system re-

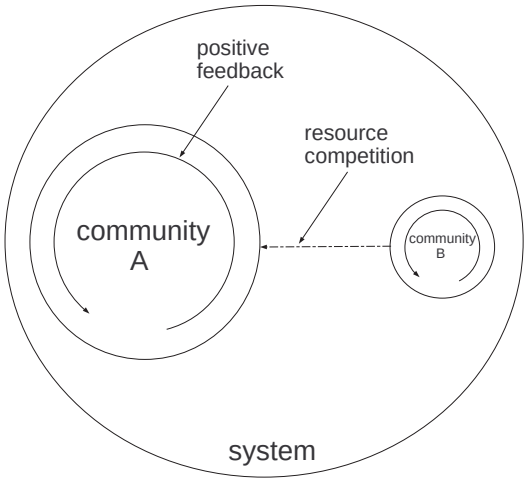

FIGURE 9.10: System configuration where only positive feedback exists. Here, two consumer sub-populations that are equal in demanded resource capacity exist, each requiring different service type for allocation. The unbalanced size of both communities (reflecting the amount of resources they consume) shows negative effect of positive feedback (denoted for both communities as an arc) that causes one community to grow at the expense of resources shortage for the latter community. This leads to the4 unstable system functioning due to resource competition reflected by the dotted arrow.

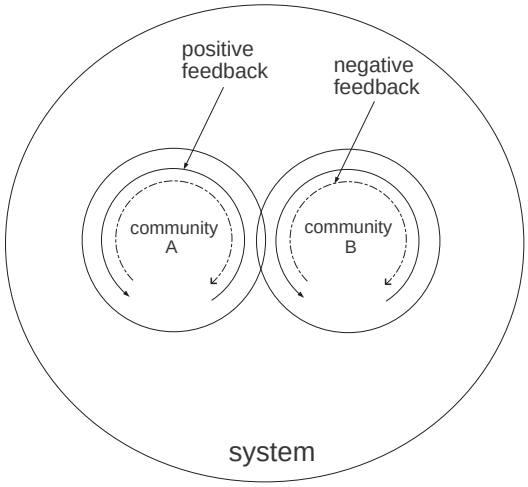

FIGURE 9.11: System configuration where positive and negative feedback exist. Here, two consumer sub-populations that are equal in demanded resource capacity exist, each requiring different service type for allocation. As a result of negative feedback (dotted arc), the system is capable to regulate the growth of both communities such that they consume equal resource capacity and the effects of resource competition are minimal.

sources that is equal in size. Given this configuration, for an optimal system configuration we should observe the emergence of two communities that are equal in size, meaning that the system resources were equally distributed among both communities.

However, as Figure 9.10 shows, the reliance of consumer agents only on mechanisms reinforcing positive feedback causes the disproportional growth of the stronger and larger A community at the cost of limited amount of resources left for the smaller B community. Such uneven distribution of system resources across the two different communities brings about negative effect in the form of resource competition, during which consumer agents from the B community encourage provider agents from the A community to reconfigure and offer demanded by them service. Such reconfiguration generates opposite conditions to the ones presented in Figure 9.10, in which most of the resources are consumed by the B community at the cost of resource shortage for the A community. This uncontrolled community growth leads to a pathological and continuous competition that destabilises resource market and degenerates system performance.

A stabilising effect of negative feedback for the identical system configuration but now employing mechanisms generating both positive and negative feedback is illustrated in Figure 9.11. Here, the positive feedback allows communities to be formed, however, their growth is no longer unlimited but becomes inhibited as soon as enough resources are drawn by the community to preserve its efficient functioning. As a result of this, both communities are of the equal size meaning that the system resources were fairly divided among the agents requiring two different service types. As a result of this, the negative effect of resource competition is avoided and the minimal competition (showed by the slightly overlapping communities) allows for run-time reconfiguration of a small quantity of resources in situations when the demand for one community is slightly greater than for the other.

Given the exemplary description of the stabilising effects of the negative feedback, let us now present how such feature was provided within our multi-agent system model. In the previous section we outlined that the emergence of the positive feedback that arises among a group of system agents is facilitated through the continuous flow of information across consumer agents that, by sustaining positive affinity scores for provider agents which act as intermediaries during such information exchange, influence reciprocal knowledge sharing among a subset of community members.

As stated, such cooperative information sharing, whilst supporting agent communities, may also destabilise resource market once too much resources become 'consumed' by such a community. Therefore, one possible way of inhibiting negative effects of the positive feedback, which we apply in our model, is to limit the information flow across community members once the destabilising effects of such community growth are perceived. Such information flow regulatory process is conducted on a local basis and aims at excluding from the information flow provider agents that become unstable at offering resource type that is required by the community members. Once information flow from such agents becomes blocked, the affinity scores kept by consumer agents for these providers decay (as discussed in Section 5.3.3.6) and thus consumer agents are no longer incentivised to share their personal evaluation scores with these agents.

In our model the above described information flow regulation stems from two important mechanisms. First, each consumer and provider agent maintains its personal measure of performance (described in Section 5.3.1 for consumers and in Section 5.3.2 for providers) that indicates how stressed (and thus inefficient) it is. Whereas for consumers such measure indicates difficulty in allocating requested tasks, the provider level of stress increases as a result of more frequent or probable service type reconfiguration. Provided these two simple measures, consumers rely on simple function (discussed in Section 5.3.3.7) based on which they decide to accept information from providers that exhibit low stress and under circumstances when their personal stress level is sufficiently low.

The application of such mechanism not only prevents negative effects of community

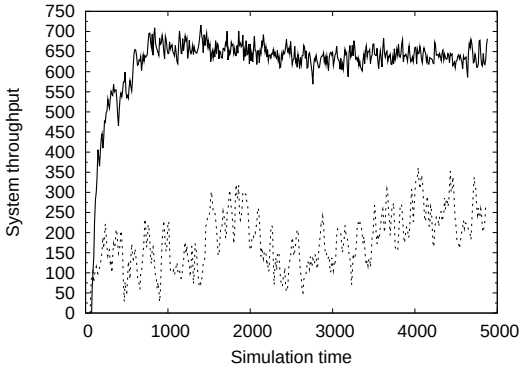

FIGURE 9.12: System throughput achieved by two model configurations. The configuration in which agents are provided with mechanisms that facilitated both positive and negative feedback is illustrated by the solid line. Poor performance shown by dotted line corresponds to model configuration in which agents are equipped with positive feedback mechanisms only.

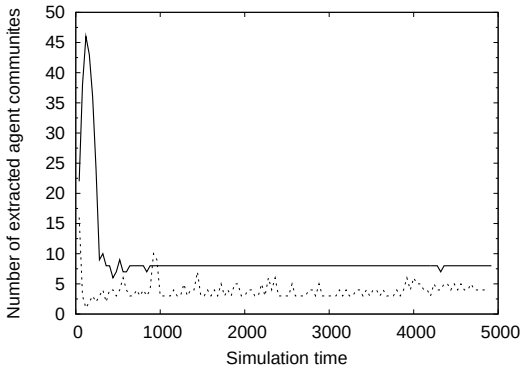

FIGURE 9.13: Number of extracted agent communities for two model configurations. The configuration in which agents are provided with mechanisms that facilitated both positive and negative feedback is illustrated by the solid line. Poor performance shown by dotted line corresponds to model configuration in which agents are equipped with positive feedback mechanisms only.

growth but also avoids inflow of information from provider agents that exist at the outset of the two communities and thus may lead to propagation of information that is circulated within other community and thus could frustrate resource market stability.

The stabilising role of this information flow regulatory technique is presented based on the experimental results provided in Figure 9.12. Here, the performance of the two identical system models is illustrated that both employ positive feedback mechanisms enforcing community formation, but only one (illustrated in the figure by solid line) relies on the information flow regulatory mechanism that acts as a negative, stabilising, feedback. As illustrated in Figure 9.13, only the latter model configuration is capable to organise into the proper (for this particular model configuration equal to eight) number of highly homogeneous consumer agent communities.

9.3 Thermodynamic Interpretation

9.3.1 Autonomic system self-organisation process overview

Given conditions and mechanisms for self-organisation that we outlined in the two previous sections, let us explain in detail how both combine in order to facilitate self-organisation in our decentralised computational system model.

For this purpose consider Figure 9.14 that illustrates autonomic system model viewed

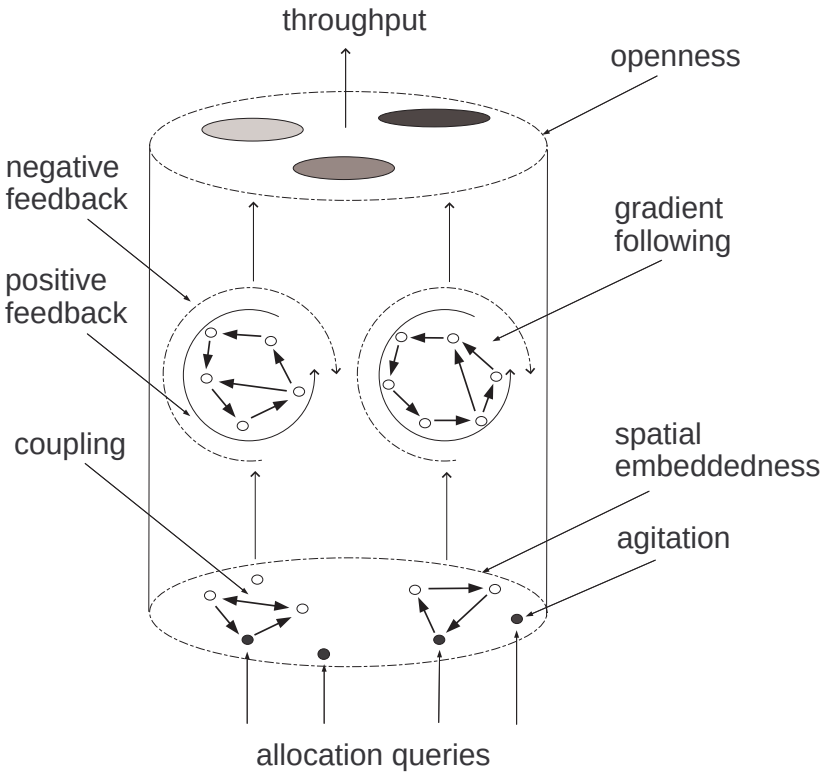

FIGURE 9.14: Delivery of global system functionality such as load-balancing, adaptive service provisioning or power management is facilitated through self-organising agent communities. Formation and stabilisation of these communities is achieved through local decision-making mechanisms that give rise to *coupling*, *positive feedback* and *negative feedback*.

from three distinct levels: micro, meso and macro. As in the general self-organising system model illustrated in section discussing conditions for self-organisation (Section 9.2.1), also in here we assume micro-level to be the level at which individual agents and their interactions are considered. The outcome of these interactions, in the form of agent communities, is presented on the meso-level. Finally, the global system functionality (load-balancing, adaptive service provisioning and power management) that arises from the organisation of system into the agent communities is illustrated on the macro-level.

Given this description, the conditions that are necessary for system self-organisation are indicated on the right-hand side of Figure 9.14, whereas the mechanisms that influence system organisation are depicted on the left-hand side of the figure.

The triggering event that initiates self-organisation is the inflow of resource allocation queries that are denoted as arrows at the bottom of the figure. These queries become intercepted by consumer agents that start interacting with provider agents. As individual agents become agitated, the system is put under allocative pressure that is proportional to the amount of inflowing queries.

During these conditions the system is in flux, where both consumers and provider agents are not organised for efficient delivery of requested resources and thus experience high

degree of pressure. Such pressure is perceived by individual agents in the form of stress the estimate of which is calculated by each agent based on its personal efficiency and then communicated to other agents. As described in Section 9.2.2.4 discussing negative feedback mechanisms, consumer agents react to high stress by cutting the information flow from highly stressed peers and thus preventing growth of the community structures that are inefficient at suppressing their locally sensed stress.

Facilitated in this manner reorganisation of agent interactions explores the ones that allow individual agents to minimise their stress. This takes place once efficient coupling between consumer-provider pairs is established as a result of which providers learn to reliably offer required service types and consumers become attracted to such providers. Once such locally stable configurations emerge, the consumer agents become encouraged by the low stress to disseminate their local provider evaluation scores to other agents that are co-located on the provider agents to which this information is sent. Such information flow catalyses formation of positive feedback (discussed in Section 9.2.2.3), where agents receiving such information become incentivised to reveal their own personal provider evaluations to the information sender. As a result of such reciprocal acts, a network topology is build within agent's local memories (based on affinity scores) that defines which peers are neighbouring and thus should be considered as a members of community to which information should be disseminated.

As a result of these local adjustments, the underlying topology of preferred interactions starts to emerge between interacting agents and is manifested by the organisation of system agents into communities (illustrated in Figure 9.14 at meso-level). Guided by the underlying interaction topology, the community members consider interactions between other members of the same community more frequently than the other, external to the community agents. Consequently, they enforce reciprocity and up-to-date information flow about the state of the system resources that are relevant for consumer agents forming such a community.

Such collectively maintained knowledge that is circulated within the boundaries of the community acts on community members as a 'force' that, in thermodynamic terms, displaces them from equilibrium through the formation of non-uniform informational gradient, example of which we illustrated in Section 9.2.1.4. Given such a gradient, agents act in a manner that tends to dissipate it and thus continue selection of resource providers with the highest evaluation scores. During this process useful work is extracted from the system as agents, guided by the collectively established knowledge, contribute towards the system throughput.

The more detailed explanation of the role that agent communities play in achieving high system throughput is presented in the following section.

9.3.2 Agent Communities as Computational Thermodynamic Engines

In this section we propose one of the possible interpretations of the role that agent communities play at achieving high system efficiency. In doing so, we suggest that these dynamic structures can be viewed as computational 'thermodynamic engines' that, when supplied with information, transform it into a usable work. More importantly, such work extraction is conducted through constraint formation and its gradual release that underly the operation of any thermodynamic engine.

To draw on these assumptions, we first provide two models of such engines: mechanical heat engine and a living organism. Relying on these two examples we then provide one of the possible interpretations of agent communities and their role in extracting work that is analogous to thermodynamic engine operation.

According to thermodynamics, work can be extracted from the system only when supplied to it energy becomes constrained, thus preventing its free flow within the system. At these conditions, it is said that the system is pushed from equilibrium state, manifested by the formation of the strong enough gradient that is visible on the level of individual system elements as a constraint on their possible actions. The gradual and controlled removal of such constraints constitutes then the period at which work can be extracted from the system. This universal feature is exemplified in Figure 9.15 illustrating the diagram of a heat engine. Here, the system is located in between two reservoirs (hot and cold). The hot reservoir is the heat source that provides energy (in the form of a heat denoted as Q_H) to the engine, whereas the cold reservoir is the 'heat sink' to which the waste energy [3] (Q_C) is dissipated from the system. Given this diagram, once the flow of energy between both reservoirs is allowed, the second law of thermodynamics acts so as to equilibrate the temperature of both, with the strength proportional to the the difference between T_H and T_C that defines the temperature gradient. Work (illustrated in the figure by W) is thus achieved by placing an artifact capable to exploit such directed energy flow between the two reservoirs. Below we illustrate two examples of such artifacts, a mechanical one and organic.

9.3.2.1 Mechanical thermodynamic engine

A human realisation of a thermodynamic engine is provided based on the steam engine the main architectural components are illustrated in Figure 9.16 where, similar to the heat engine model described above, we distinguish heat source (T_H) on the left of the engine and the cooling source (T_C) on its right. Since the purpose of any heat engine is to convert heat into mechanical work, in between heat and cool sources we have an

[3] Waste energy refers to the energy that dissipates from the system and cannot be used for work extraction by the same system.

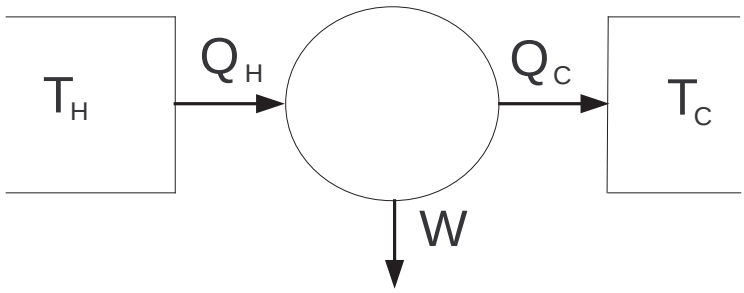

FIGURE 9.15: Heat engine diagram. Here, the engine (illustrated by the circle) is situated between the heat source (T_H) and the cold sink (T_C). Q_H is the heat flowing into the engine whereas Q_C is waste heat going into the cold sink. W is the useful work coming out of the engine.

engine consisting of cylinder head, piston and crankshaft. How work is extracted from such a mechanical device?

The engine operation begins with the piston at the top of its stroke and located near the cylinder head. The working gas is all contained in the cylinder space between the piston face and the cylinder head. The heat source is now applied to the outside of the engine cylinder, heat transfers to the cylinder and into the gas, and the gas temperature and pressure begin to rise. After some period of time, the temperature and pressure in the working gas reach a maximum. The piston is now allowed to travel downward until it reaches it most downward position. As the piston moves downward the gas begins to expand. Normally, an expanding gas would cool but heat is continually transferred into the working gas from the heat source and the hot cylinder wall keeping the gas at its maximum temperature. Since the pressure force on the piston is acting downward and the piston is traveling downward, work is being done by the gas. In simple terms, the gas is forcing the piston downward, is twisting the crankshaft, and is doing work in the process.

When the piston reaches the most downward position, the next step in the cycle begins. The heating source is removed from the cylinder and the cooling source is applied to the cylinder while the piston remains at this position. As heat is transferred from the gas to the cooling source the gas temperature and the gas pressure both fall. After some amount of time the lowest gas temperature is attained. The next step in this engine thermodynamic cycle is to push the piston back in to the starting position. As the piston is pushed in the gas is being compressed and the gas temperature would normally begin to rise. However, the cooling source is still applied to the cylinder and prevents this temperature rise. The next and final step is to remove the cooling source, apply the heating source, and heat the gas back up to the starting high temperature.

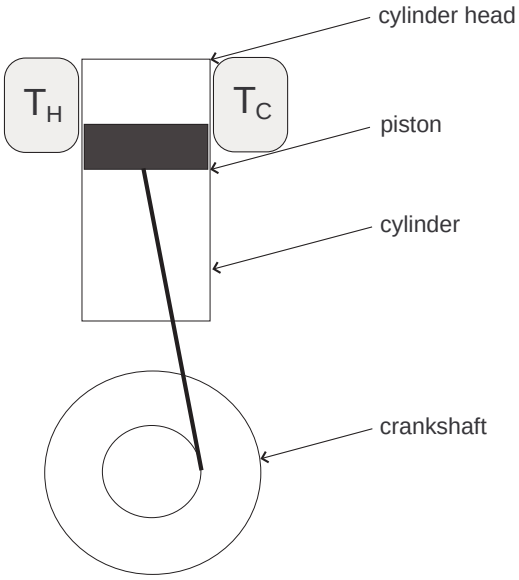

FIGURE 9.16: Mechanical thermodynamic engine. Here, the system comprises following mechanical elements: cylinder, piston and crankshaft. The mechanical work is extracted by heating up the cylinder (using provided T_H) that causes the gas, which is located between the cylinder head and the piston, to expand and thus to move piston downards. This propels the crankshaft as a result of which mechanical work is extracted.

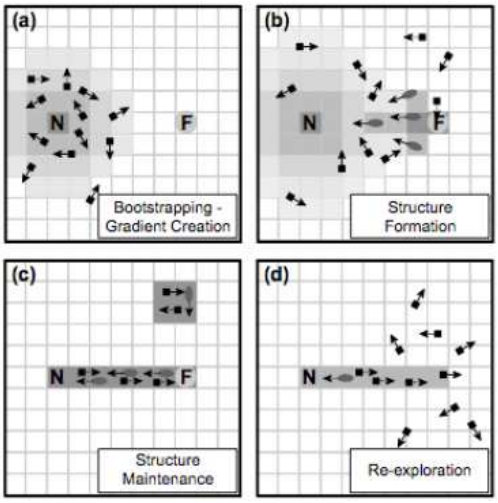

FIGURE 9.17: Organic thermodynamic engine. Here, a model of an ant colony is presented and the process of ants self-organisation into a foraging trail re-described in terms of thermodynamic work extraction involving four key steps: (a) Gradient creation; (b) Structure formation; (c) Structure maintenance and; (d) Re-exploration.

Above described process constitutes a single thermodynamic work cycle that conducted repetitively underpins the supply of most of the world's electric power and almost all motor vehicles.

9.3.2.2 Organic thermodynamic engine

It is suggested that the thermodynamic principles that mankind has harnessed for mechanical work extraction underly the existence of all living structures that, according to Kauffman [55], employ thermodynamics for work extraction that is then used for maintaining their internal metabolism and thus existence. However, when compared with mechanistic engines, there are several important for us differences that distinguish such natural thermodynamic systems from their man-made artifacts. Firstly, living systems do not comprise of a static and pre-imposed configuration in the form of a piston, cylinder and crankshaft but consist of a number of autonomous and interacting elements. Secondly, whereas the existence of hot and cold reservoirs that define gradient from which work can be extracted in man-made thermodynamic engine is provided as a part

of the engine design, natural systems need to discover such gradients autonomously. As suggested by Kauffman, a thermodynamic work-cycle of natural systems requires them to: (1) measure useful displacements from equilibrium from which work can be extracted; (2) discover the devices to couple to those energy sources such that work can be extracted; and (3) apply work to develop constraints to extract further work.

An example of a thermodynamic work extraction conducted by an organic system that comprises a population of autonomous system elements is provided by Gambhir *et al.* in [24]. Here, the authors suggest that the typical evolution of the ant system that self-organises to efficiently transport food to the nest can be re-described in terms of a thermodynamic work-cycle. As illustrated in Figure 9.17, four steps can be distinguished during this process.

1. Gradient Creation — Ants move randomly out from the nest, creating a gradient of nest pheromones.
2. Structure Formation — Some ants find the food and begin following the nest pheromones while dropping food pheromones that food-seeking ants begin to follow.
3. Structure Maintenance — A stable path of both food and nest pheromones is established. As shown in the upper-right corner, cycles that do not transport food can also form.
4. Re-exploration — Once all of the food has been transported to the nest the pheromones begin to evaporate and the ants disperse.

As illustrated by this example, the realisation of a thermodynamic work-cycle within such systems is inherently dependent on their organisational adaptation that facilitates the most efficient transfer of energy across the system such that a usable work is produced. To achieve this, living systems often employ decision-making involving local information exchange and co-adaptation of individual system elements to each other. In such a setting, an 'organic' artifact that extracts work in an analogous manner to the mechanical steam engine described above, is the organisation of ants into a collective that forms a foraging trail. This dynamic structure and the information propagated through its constituents imposes a constraint on their *behavioural repertoire* which, in turn, achieves organised and efficient food transportation.

9.3.2.3 Computational thermodynamic engine

Given the fact that we designed our model based on thermodynamic principles of self-organisation, it is instructive to attempt to interpret its functioning analogously to the operation of thermodynamic engines. For this reason, in what follows we re-describe

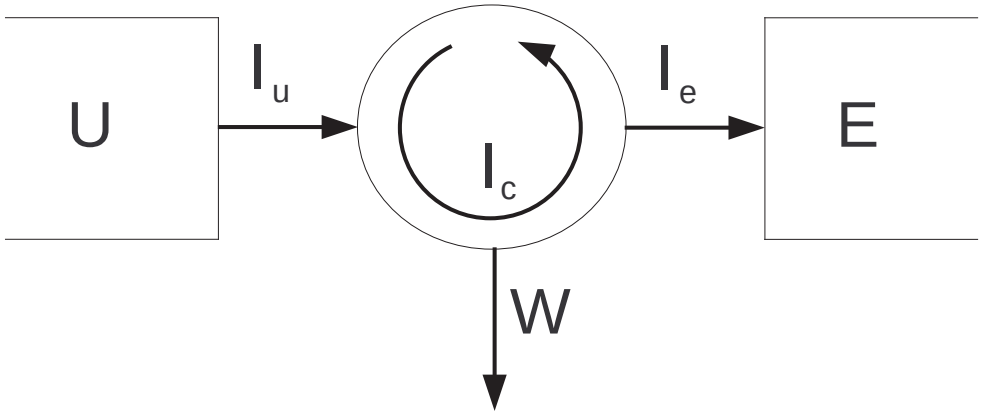

FIGURE 9.18: Computational thermodynamic engine diagram. Here, the 'heat' source ($\boldsymbol{U}$) represents a group of infrastructure users from which a stream of information ($\boldsymbol{I_u}$) (representing service allocation requests) is fed to the agent community (denoted by the circle). The information that is considered by the community as a 'waste' ($\boldsymbol{I_e}$) is dissipated outside its boundaries to the environment ($\boldsymbol{E}$). Task allocation and thus work extraction ($\boldsymbol{W}$) by the community is achieved through organised transfer of information $\boldsymbol{I_c}$ across community members

agent communities as artifacts which act analogously to thermodynamic engines but in stead of energy are fed with information that they transform into informational gradients from which useful work can be extracted.

Figure 9.18 illustrates the diagram of a computational thermodynamic engine. Analogously to the heat engine diagram illustrated in Figure 9.15, we assume that agent community (depicted as the circle) is situated in between 'energy' source (U) and the 'energy' sink (E). However, whilst in the steam engine example 'energy' defines a heat flowing through the system, we assume that our computational model operates based on the organised flow of information and thus both the source and the sink distinguish endpoints from and to which information flows. More specifically, the information source is a group of infrastructure users (U) that send out a stream of information (I_u) in the form of allocative requests. These requests, once received by the agent community, agitate its members and catalyse their interactions aimed at delivering the requested tasks. As an outcome of this, useful work is performed by the community (W), and the waste product during this process (I_e) is disseminated to the information sink (E) represented by the environment [4]. The new feature that the traditional heat engine lacks and that underlies the function of agent communities is the existence of information I_c that is propagated within the community boundaries in a circular fashion. As this information is the motive force for achieving work by the community, we discuss it in a greater detail below.

[4]The waste product is defined as a set of evaluation scores identifying provider agents that no longer are employed by the community and thus their evaluation scores gradually dissipate from the memories of individual community members.

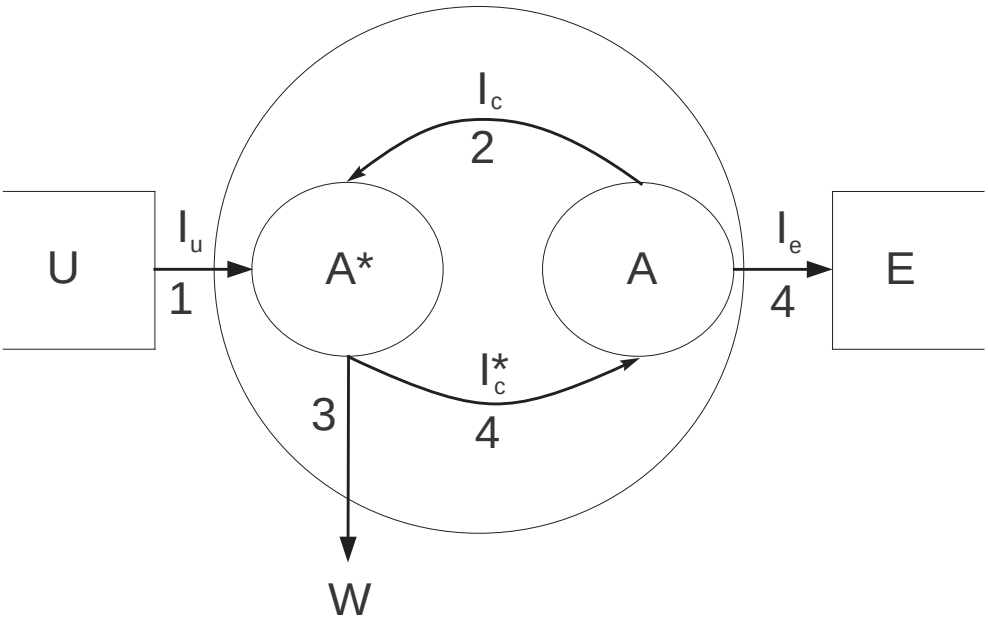

FIGURE 9.19: Single agent thermodynamic work-cycle. Step 1: Inflowing allocation request (I_u) agitates consumer agent (denoted by A^*). Step 2: Constraint required for a useful work extraction is established as a result of I_c information inflow from community members (A). Step 3: Work is extracted by following the informational gradient. Step 4: Constraints are released as a result of local information dissemination to other agents as well as its dissipation (I_e) to the environment (E).

Given this general overview, let us focus on the re-interpretation of a thermodynamic work-cycle within an *engine* represented by an agent community. For this purpose consider Figure 9.19 that details the inner workings of such an engine. Here, we take a perspective where a consumer agent (depicted in the figure as A^*) is required to allocate a single task. In what follows we re-describe such a task allocation as a thermodynamic work-cycle consisting of four general steps that are illustrated in Figure 9.19 and explained below.

Step 1: Agent agitation

To operate, thermodynamic engine requires supply of energy. In the case of a steam engine such energy inflow takes place as a result of heat transfer from the heat source. The analogous triggering event in a computational model illustrated in Figure 9.19 is represented by the inflow of information (I_u) that represents allocation requests originating from the information source (U).

Step 2: Gradient formation

Recall that work extraction from a thermodynamic engine is possible only when the system is able to establish a thermodynamic gradient. This, in the case of the steam

engine, requires engine pre-set, during which piston is set to reside at the upward position and the heat source is set into the contact with the cylinder. At this stage, the heat is transferred inside the cylinder and by increasing gas temperature causes the gas to expand, generating the gradient reflected by the increasing gas pressure.

The emergence of an analogous informational gradient within a computational model is represented by the inflow of I_c information to A^* agent before it starts its allocation (depicted in Figure 9.19 as step 2). Since the communicated information originates from other consumer agents and represents their provider evaluation scores, such information, once accepted by the consumer, imposes a constraint on its *behavioural repertoire* as it attracts the agent towards a subset of available provider agents [5].

Step 3: Work extraction

Work extraction in the heat engine takes place once the gradient reflected by the gas pressure is strong enough to push the piston downwards such that it perpetuates the crankshaft and thus generates useful work. In a computational system work extraction (depicted in the figure as W) follows from the selection of provider agent capable to satisfy the demanded allocation task. Because the A^* consumer agent employs for this purpose informational gradient, the selection is based on recent knowledge about the resources state and thus increases the chance of task allocation success in the smallest amount of requests. Analogously to the steam engine, where some quantity of the energy is being used to push the piston, here the provider allocation consumes its computational resources and thus renders the information about its availability stale and outdated.

Step 4: Gradient dissipation

To finalise the thermodynamic work-cycle, the piston within the steam engine needs to be shifted to its upward state. To realise this, the waste heat inside the cylinder is removed to the heat sink thus allowing the piston to move upwards by compressing the cold air. Finally, the engine is ready for another work cycle to be performed.

How can we interpret this step within a computational system? Recall that once A^* agent successfully allocates the task, it updates the evaluation score of the employed provider and disseminates its personal provider evaluation scores in the form of I_c information to A agents that represent the information sink. The exposure of personal evaluations to other agents has two consequences. Firstly, the information sender propagates the constraint to other agents, as such information will impose (or sustain) gradient within the memories of these agents in the same manner as it did for this agent at the beginning of its allocation cycle. Secondly, as other consumer agents become informed

[5] In this example we are assuming that community is stable and organised to effectively allocate demanded tasks.

about the up-to-date availability of a particular provider, the consumer that revealed this information lowers its chance of employing the same provider in the next round. Although such information reflects the equilibrating tendency, during which informational gradient becomes less efficient, recall that it is precisely such flow of information that displaced the consumer in the first place and allowed it to perform useful work. It is important to note that whilst in the steam engine example the heat that has remained in the cylinder after work extraction is considered as a waste heat that is not re-applied for work extraction, the 'waste' product of resource allocation in the form of useful information about providers state is not dissipated from the system but circulated among community members [6]. Such continually updated and re-cycled information constitutes the main driving force for generating and maintaining up-to-date informational gradient.

The provided above interpretation suggests that each individual agent can be considered as computational engine that performs work on behalf of the infrastructure user. However, as we also suggested, work extraction within such a system is possible only when the constraint is imposed on the consumer agent at the beginning of its allocation process. Whilst single agent is unable to achieve such constraint alone, as we explained above, it is the collective information sharing performed by the group of agents that not only establishes informational gradient but also, by disseminating it, propagates the constraint to the system.

As such development of constraints is critical for further work extraction, the role of agent community viewed from this perspective is to establish information flow topology that maximises the propagation of constraints between its members. The community analysis results suggested that the precondition for such collective gradient emergence is the formation of highly homogeneous community in which the locally communicated information is continually recycled and flows only within the community boundaries. Only within such a subset of interdependent agents the positive feedback can arise and agents start propagating constraints that are, as a result of circular relations, fed back to them thus allowing useful work extraction in subsequent allocations.

9.4 Conclusions

Engineering of systems that exploit thermodynamics of self-organisation opens an exciting area of research in which novel and nature inspired control mechanisms can be provided. These do not rely on pre-programmed plans or solutions that are prone to failure in dynamic and harsh conditions but, in stead, exploit run-time reconfiguration and adaptation capabilities that stem from their decentralised and autonomous nature.

[6] The only information that is dissipated as a waste (depicted in Figure 9.19 as I_e) are the evaluation scores of provider agents that have been identified by individual consumer agents as configured to offer different service types than the ones required by the community and thus are no longer of the interest for the community

Whilst such systems are assumed to continually seek to improve their organisation, and do so in a wide range of unexpected conditions, it is assumed that one of their main strengths lies in the ability to minimise operation costs involved with the regulation of vital system functions such as the ones discussed in this thesis.

As the ability to deal with such complex settings is far beyond human administration and the highest profit can be achieved from large scale deployments, control mechanisms that are scaleable, dependable and cheap are of critical importance.

Considering the still increasing ubiquity of complex software systems and their growing management costs, the development of autonomic control mechanisms that preserve low operation and maintenance costs of such infrastructures may turn out as important as the development of the first thermodynamic engine almost 200 years ago.

To address this autonomic computing challenge, in this chapter we have provided a thermodynamic interpretation of our computational model functioning. In doing so, we suggested that the design of self-organising computational systems should take into account both, conditions that allow self-organisation to take place and mechanisms that facilitate it. Following this, we presented a set of design principles that aid in construction of computational systems that exploit thermodynamics of self-organisation and proposed an interpretation of agent communities as dynamic structures which maximise system throughput in a manner that is analogous to work extraction performed by thermodynamic engines.

As a result of this work, a number of practical pointers can be provided to software engineers aiming to deliver control mechanisms that are scaleable and robust to dynamic system functioning conditions. First of all, engineers should focus their attention on the design of systems that act in accordance to thermodynamic laws. As we have showed in this chapter, there are several rules that need to be incorporated within a computational system in order to facilitate this. In particular, attention to the role of local information flow and its effects on the efficiency of the system should be paid. From the thermodynamic viewpoint, it is assumed that the information flow should give rise to informational gradients that constrain the behavioural degree of freedom of individual system elements and thus give rise to self-sustainable system organisation, reflected in our model by agent communities. Such organisation can be then employed for efficient work extraction from the system. For example, agent communities that were identified in our model as structural organisaitons were exploited for load-balancing, adaptive service provisoining and power management as they allowed individual agents belonging to a community to effectively regulate the access to required system resources through the collective pressure exerted by the community.

Whilst thermodynamics provides a basis for understanding how such organisation is achieved through self-sustaining flows of information, the theory of self-organisation provides a practical pointers to processes that need to be incorporated within the system

in order to achieve the self-perpetuating information flow in the first place. Relevant in achieving this processes such as *coupling*, *positive* and *negative feedback loops* are described in Section 9.2.2.

Whilst this provides a promising potential for engineering computational self-organising systems, the thermodynamic account of self-organisation provided in this chapter should be considered as a preliminary work. With respect to this, identified principles should be considered as rules of the thumb, where much work is still required at understanding how bottom-up system organisation can be effectively regulated within this kind of systems. The limitations of provided interpretation as well as future work directions aiming to extend the work discussed in this thesis are provided in the following final chapter.

Chapter 10

Conclusions and Future Work

Having described a bottom-up approach to control resource provision within autonomic systems, the aim of this chapter is to summarise our work by outlining both what we have achieved and what questions remain unanswered. For this purpose, in Section 10.1, we first outline what we have achieved so far by providing a brief summary of each of the chapters presented in this thesis. In Section 10.2 we provide a more detailed overview of contributions we have made in this work, whereas Section 10.3 discusses the main limitations of our work. The main direction in which our work could be extended in order to address existing limitations as well as make further advances is provided in Section 10.4. Finally, Section 10.5 draws the main conclusions from the thesis.

10.1 Thesis Summary

Efficient resource management is one of key problems associated with large-scale distributed computational systems. Taking into account their increasing complexity, inherent distribution and dynamism, such systems are required to adjust and adapt their resource markets at run-time and with minimal cost. However, as observed by major IT vendors such as IBM, SUN or HP, the very nature of such systems prevents any reliable and efficient control over their functioning through human administration.

For this reason, autonomic system architectures capable of regulating their own functioning are suggested as an alternative solution to the looming software complexity crisis. Here, large-scale infrastructures are assumed to comprise myriads of autonomic elements, each acting, learning or evolving separately in response to interactions in its local environment. The self-regulation of the whole system, in turn, becomes a product of local adaptations and interactions between system elements.

Although many researchers suggest the application of multi-agent systems that are suitable for realising this vision, not much is known about regulatory mechanisms that are

capable of influencing efficient organisation within a system comprising a population of locally and autonomously interacting agents.

To address this problem, our aim was to develop local decision control mechanisms that are capable of preserving efficient resource management in dynamic and unpredictable systems and where the control over this process is fully decentralised and arises in a bottom-up manner. To do so, we have identified complex natural systems and their self-organising properties as an area that may deliver novel control solutions within the context of autonomic computing.

In such a setting, a central challenge for the construction of distributed computational systems was to develop an engineering methodology that can exploit self-organising principles observed in natural systems. This, in particular, required the identification of conditions and local mechanisms that give rise to useful self-organisation of interacting elements into structures that support the required system functionality. To achieve this, we proposed an autonomic system model exploiting self-organising algorithms and its thermodynamic interpretation, providing a general understanding of self-organising processes that need to be taken into account within artificial systems exploiting self-organisation.

Before addressing these aims directly, in Chapter 2 we reviewed important characteristics of natural complex systems and presented existing approaches in applying self-organisation to control artificial systems. Here, various techniques and local decision-making mechanisms have been applied such as evolutionary algorithms or *stimuli-response* mechanisms inspired by self-organisation phenomenon observed in insect societies. Despite these advancements, the main difficulty in achieving reliable and decentralised control in most reviewed approaches was associated with the unpredictability resulting from emergent system properties that are difficult to analyse and interpret analytically. As a consequence, although the reviewed complex system models showed interesting self-organising capabilities, they are achieved at the cost of large amount of experimentation and model tinkering.

Whilst the attempt to engineer reliable complex systems was not the main issue within discussed works, the application of such complex systems to preserve control over autonomic computing infrastructures requires more principled and methodological approach. In particular, given the fact that complex systems exhibit emergent behaviour that, as we have seen, is the motive force for achieving organised (or chaotic) system response, a careful understanding of such phenomenon is required if we are to utilise it for system's control. For this purpose, our aim was not only to provide adaptive decision-making mechanisms but, more importantly, to understand why and under what conditions such mechanisms lead to increased system efficiency.

To advance the current state of research in this direction, we considered a study of thermodynamics and self-organisation as key investigation areas. According to the thermo-

dynamics of self-organisation, the increase in system organisation could be understood as the emergence of constraint imposed on individual system elements that arises as a result of energy (or information) flow across the system's boundary. If properly regulated, such flow displaces the system from equilibrium and allows useful work to be extracted from the system.

We began to explore this property in Chapter 4, where a simple autonomic model was proposed and its self-organising characteristics analysed based on thermodynamic concepts of constraint, work and equilibrium. Building on this, Chapter 5 introduced a more realistic and fully decentralised autonomic system model. Here, local decision making mechanisms inspired by the behaviour of insect societies were employed, and two novel regulatory mechanisms introduced: *affinity algorithm* and *information flow regulatory mechanism*. Whereas the first mechanism influences and catalyses the formation of agent communities, the second is responsible for the regulation of how much information is communicated between interacting agents such that the system stability is preserved. Finally, after evaluating the performance of the model in the three following chapters (Chapter 6, Chapter 7 and Chapter 8), in Chapter 9 we offered a thermodynamic interpretation of our model functioning and provided a set of design principles helpful in engineering artificial self-organising systems employing thermodynamics of self-organisation.

10.2 Research Contributions

The main research contributions arise from the specification of the general framework for engineering self-organising computational systems based on thermodynamic principles of self-organisation and the development of autonomic system model employing these principles for adaptive resource management. Both of these show how open, decentralised autonomic systems may be constructed and applied in a bottom-up manner to control the dynamic and indeterministic process of service provisioning.

In addition, the computational models provide a basis of understanding and analysing the impact of three features that are often neglected and avoided in exemplary models of autonomic systems. First, we considered open and dynamic system environments where agents may enter or leave the system and where the demand for particular resource types may undergo change at run-time. In particular, we applied and combined demand functions that simulate influx of new agents and varying demand conditions. This allowed us to analyse both the efficiency of self-organisation in conditions where the system is required to adapt and adjust to dynamic conditions together with a testbed for investigating system stability and resilience to external perturbations.

Second, we did not study each system-level function (load-balancing, adaptive service provisioning and power management) in isolation but incrementally introduced one after

another to the model. This gradually increased the difficulty and realism of resource management, where the system was required to preserve efficient distribution of requests, resource market adaptation and power management efficiency at the same time.

Third, we provided a fully decentralised model where no central or distributed information repository existed within the system that agents could employ in order to increase their awareness about the current resources or demand state. Rather, agents employed local information communication and their awareness was the direct result of emergent organisation into communities preserving up-to-date information flow about the system state.

More significantly, we contribute to the state-of-the art in following areas: understanding the means of achieving bottom-up control in dynamic and open autonomic systems; provision of decentralised techniques that achieve this relying on self-organisation process; and interpretation of this process as arising from certain laws and conditions existent in natural self-organising systems. We discuss each of these in the subsections that follow.

10.2.1 Decentralised Autonomic System Model

We offer a decentralised autonomic system model that is tasked to control resource management in dynamic and open environments. Using this model, we study how adaptive system-level response arises out of local interactions of individual system elements that employ for this purpose only simple stimuli-response mechanisms and local information exchange.

10.2.2 Self-organising Agent Communities

Understanding how local interactions between system elements give rise to certain global system dynamics becomes one of the major difficulties whilst engineering decentralised systems. In this thesis we address this problem by extending the system analysis to include an intermediary level (*meso-level*), residing between the level at which the behaviour of individual agents is analysed (micro-level) and the level at which the collective response of the whole system is considered (macro-level).

At this intermediary level we identify system organisation into agent communities that are the key structures that support adaptive and efficient maintenance of the three system functions: load-balancing, adaptive service provisioning and power management.

10.2.3 Importance of Spatial Embeddedness for Self-organisation

We show the relevance of spatial embeddedness for achieving system self-organisation. To achieve such spatial property, the system agents are instructed to establish interaction

topology according to which the peers that are closer to each other are more likely to interact and affect each other behaviour than the elements that are located at a greater distance. Only when such underlying interaction topology arises, can system components organise into globally efficient collective structures referred to as communities.

To realise such topology in our model we propose an *affinity* algorithm (described in detail in Section 5.3.3.6) the role of which during community formation is discussed in Section 9.2.2.3.

10.2.4 Thermodynamics in Computational System

Whilst thermodynamics is mostly related to the study of heat engines and provides basis for understanding how mechanical work can be extracted from the supplied energy, we stress the relevance of this discipline in achieving computational system self-organisation.

More specifically, we show that natural self-organising systems employ the same work extraction principles for achieving adaptive respnose and that understanding of conditions and mechanisms influencing such process will contribute towards principled engineering of adaptive and decentralised autonomic systems.

For this purpose we propose a set of design principles which, when incorporated into our model, based on exemplary mechanisms, achieve system organisation that is analogous to the one existent in the natural systems. As such bottom-up organisation is driven by thermodynamic principles of self-organisation, we provide a thermodynamic interpretation of agent communities as artifacts which act analogously to thermodynamic engines but in stead of energy are fed with information that they transform into informational gradients from which useful work can be extracted.

10.2.5 System stabilisation through positive and negative feedback

A system in which its constituents employ only locally available information whilst conducting resource allocation decisions is prone to instablilities or even chaotic response resulting from resource competition. In this study not only we show that organised system structures in the form of agent communities are critical in suppressing this pathological behaviour within a model where no central or hierarchical control is imposed, but also suggest how reliable control over the system can be provided based on positive and negative feedback loops that stabilise the formation and operation of such communities.

In particular, *affinity* algorithm (discussed in Section 5.3.3.6) is proposed that gives rise to a positive feedback responsible for triggering the agent community formation, whereas agent stress-based information inflow regulatory mechanism (discussed in Section 5.3.3.7) is introduced to facilitate negative feedback that preserves stable operation of such communities.

10.3 Limitations

Although we outlined principles of engineering decentralised systems that possess self-regulatory properties allowing them to achieve efficient resource management in a bottom-up manner, there are several open issues that were not considered within this work. In particular, we have identified following limitations.

10.3.1 Model realism

Although we paid careful attention to simulate realistic and dynamic resource allocation conditions within which self-organising system properties could be evaluated, we made an assumption that all allocation tasks involve only a single service provision.

This is not realistic as in real autonomic deployments a single allocation task may involve a number of sub-tasks that may be satisfied by different service providers. As a consequence, the overall task accomplishment is dependent on successful completion of an ordered allocation of individual services that conform to the workflow specification.

In our work only the first and simplest autonomic model (introduced in Chapter 4) was designed to take into account service workflows. As introducing workflows to a more advanced model would introduce additional complexity and require additional mechanisms responsible for achieving reliable workflow completion, we decided not to introduce this feature. By doing this, we simplified the analysis of arising agent communities and concentrated on our main objective of preserving adaptive system response in a bottom-up manner.

In practice, introducing workflows to the autonomic system proposed in this thesis would introduce more complicated system organisation, where distinct agent communities (currently homogeneous with service type demand) would overlap and form complex interrelations through which agents could affect each other's behaviour and thus influence the system dynamics.

10.3.2 Thermodynamic Work-cycle

The thermodynamic interpretation of self-organisation within computational systems provided in this thesis lays the groundwork for further investigation. In particular, an interpretation of the thermodynamic work-cycle within an information driven system would provide us with more accurate system organisation measures and local decision-making mechanisms that could directly contribute to the engineering of this kind of system.

In this context, the concept of a thermodynamic work-cycle can be understood as a repetitive and self-sustaining set of coherent actions performed by a collective of agents. Whereas in physical systems such a work-cycle is constituted through energy flow across the system that, if properly steered, establishes constraints that are further released during work extraction, we have observed that the same pattern of behaviour arises in our autonomic system model as a result of organised information flow.

However, the current analysis focused only on the global system properties and did not provide detailed analysis of individual agent communities. As these communities constitute the primary collectives of agents that we assume to be the motive force for performing thermodynamic work-cycles, a closer investigation into their inner workings would facilitate a better understanding of this phenomenon, followed by an improved efficiency of the system that employs thermodynamic engineering principles.

10.3.3 Formal Models for Maximal System Efficiency

Drawing on the thermodynamic analysis, the current model lacks a formal analysis of how beneficial the application of proposed thermodynamics-inspired design may become in comparison to the existing centralised and distributed approaches. In particular, the current model lacks any formal analysis of the maximal performance it could achieve, considering its functioning conditions and the allocation process difficulty.

Although providing this in a general manner may become difficult, we consider addressing this issue as an important step towards application of self-organisation inspired mechanisms into the design of artificial systems.

10.3.4 Model Complexity

Addressing load-balancing, adaptive service provisioning and power management through local decision-making mechanisms was not a trivial task and resulted in additional model complexity. Such a complexity was manifested by a number of parameters and constants that were involved during operation of local decision-making mechanisms and that required correct setup in order to facilitate global system self-organisation.

10.4 Future Work

To address the limitations introduced above, we identify the following two areas in which further research could be continued.

10.4.1 Towards Complex Computational Ecologies

Despite the careful consideration of the environmental dynamics and system openness, the advanced model proposed in this thesis did not consider cases where individual tasks require allocation of more than a single service. Addressing this problem requires little change within the existing architecture and may provide an interesting interplay between agents that would give rise to multi-level organisations not observed within the existing simulations. For example, in the existing model we experienced resource market segregation that gave rise to a proliferation of distinct agent communities. From a consumer perspective, each such community specialised at allocating one distinct type of a service and did not interact with communities allocating other kinds of resources.

Introducing workflows and thus requiring consumers to employ different providers during single task allocation can be facilitated by enabling consumers to hold more than a single service type registry (for each unique service type the agent is required to allocate) and thus become responsive to information flows related to these service types. Under these assumptions, each consumer would maintain a separate informational gradient for a distinct group of known service providers that specialise to offer a unique service type.

Consequently, consumer agent communities that allocate different tasks, but whose workflows are overlap, would no longer be isolated but establish interesting interdependencies and relations through which information would flow between both organisations. The stability of each such community would, in turn, depend on the performance of other interlinked communities, giving rise to even more complex computational ecologies.

10.4.2 Identifying Thermodynamic Work-cycles

Further advancements in applying a thermodynamic interpretation to the phenomenon of self-organisation within artificial systems would concentrate on the identification and interpretation of thermodynamic work-cycles. These repetitive cycles of periodic system behaviour underly the design of any thermodynamic system from which work is extracted and should gain consideration within autonomic systems relying on bottom-up organisation mechanisms.

Addressing this area of research would first require the development of system analysis tools based on which thermodynamic work-cycles could be identified within a population of interacting agents, and then, experimentation with local decision-making mechanisms responsible for gain in work extracted during such cycle. Addressing the first step can be done through a community and information flow analysis that would reveal groups of agents that collectively communicate information within the scope of such community. Given the fact that each thermodynamic work-cycle consists of two stages, that is constraint formation (during which gradient arises and the system is shifted from

equilibrium) and constraint release (during which work is extracted and the gradient dissipated), both stages could be identified within each captured community through measures of order from statistical mechanics that we have already applied to identify the level of system constraint.

Both, information which agents comprise local community and how far each community is being displaced from equilibrium in between both thermodynamic work-cycle stages could offer a means of identifying how efficient such community is. This information could then be passed as a local performance indicator to individual system agents comprising the community and thus offer a more efficient information flow regulatory mechanism aimed at maximising the displacement from equilibrium.

During this regulatory process, agents that are aware of being a part of the collective performing a thermodynamic work-cycle could direct the flow of information across other community members such that the maximal information gradient is formed and thus the system is shifted from equilibrium. Then, once the system is shifted sufficiently far from equilibrium, these agents could release constraints that were imposed by the arising information gradient by performing resource consumption. During this process, currently available resource providers would become utilised and thus information contained within the formed gradient would become stale and dissipated. This step of thermodynamic work-cycle, during which the information gradient becomes dissipated and and work is performed by the collective, would finalise single thermodynamic work-cycle.

10.5 Conclusions

Efficient resource management is becoming increasingly important in computer science as a result of emergence of open and large-scale computational infrastructures. In our work we have considered autonomic system models in which a large quantity of system resources is offered to users requesting them. Achieving efficient resource provision within such environments becomes a challenging problem due to the inherent dynamism associated with the availability of offered resources and the resource requirements imposed by the infrastructure users. As a result, many approaches aiming to deliver efficient resource management suggest the application of techniques from the field of multi-agent systems in order to introduce a certain amount of flexibility and adaptation into such systems.

For such an approach to be effective, control mechanisms must be developed that would preserve a stable and efficient global system response out of the local interactions between autonomous system elements. One way to achieve this is to rely on existing centralised or distributed control approaches that either impose central or hierarchical control over the actions of individual agents or assume that each agent operates on full information

about the system state and coordinates its decisions with others such that a globally efficient system response is attained.

However, most proposed mechanisms have been developed for small scale systems that are closed and operate in conditions where system dynamics is sufficiently low to preserve a timely response and adaptation to changing conditions. As a consequence, their application to modern and large scale open environments is less straightforward and may introduce scalability bottlenecks, brittleness and high maintenance cost that may impede further progress in the engineering and deployment of autonomic systems.

To address this, many approaches point to the decentralised control mechanisms that exist within natural distributed systems and suggest them as a source of inspiration for the design of self-organising computational systems. Before this can be realised, a sufficient understanding of natural self-organising processes needs to be obtained and computational frameworks employing them studied.

To this end, we have approached this problem by offering a thermodynamic account of self-organisation within artificial computational systems and introducing novel bottom-up control mechanisms preserving adaptive resource management within a dynamic and open autonomic system. The efficiency of this approach was empirically evaluated against three system level functions: load-balancing, adaptive service provisioning and power management with the application of decentralised multi-agent system models.

Both the models and the thermodynamic account of self-organisation within autonomic systems contribute to the state-of-the-art in the autonomic system design, and constitute a significant step toward practical autonomic management of large and open resource provisioning systems.

Appendix 1 Simulator design

Empirical results that were presented in this thesis were provided on the basis of an autonomic system model architecture described in Chapter 5.

Since the aim of the model was to simulate decentralised software system, the underlying simulation model reflects this through the lack of any central components, asynchronous interactions between agents and autonomous thread of control provided to each such element.

To realise this, each agent (consumer and provider) were created as separate software threads written in Java language, each having its own thread of control over its actions. Furthermore, decisions performed by individual agents are independent of other agents, meaning that agents interact asynchronously as they are driven by their own decision-making mechanisms.

During service provisioning process the provider spawns a (limited by total service capacity) number of service instances, where each such instance can be considered as an individual service instance that is offered to requesting it consumer agent. Within the model each such service instance is provided as a separate thread which is created by the service provider agent and maintained only for the service allocation period.

As a result of object and multi-threaded oriented design, the software platform offers a number of reusable components in the form of consumer and provider agents. Whilst these componenets are the main building blocks of the model, their internal behaviour can be easily extended through the addition of new decision-making strategies.

Furthermore, the model offers a three dimensional visualisation allowing the viewer to observe allocation and information exchange interaction among agents as the simulation runs. This offers an interesting insight into both the pattern of local interactions among agents as well as the global system organisation into communities that can be easily observed and tracked through visual output. As the visualisation is interactive and allows the viewer to modify the system state (add/remove agents, change resource demand) the viewer is provided with the visual tool allowing it to influence and observe system self-organisation process that would be difficult to comprehend through analytically analysis only. Apart from such a pedagogical contribution, the visualisation offers a novel way of supporting human administration tasks as it allows the potential administrator to observe effects of his system adjustments on the global system level.

Bibliography

[1] K.R. Abbott and D.R. McCarthy. Administration and autonomy in a replication-transparent distributed dbms. In François Bancilhon and David J. DeWitt, editors, *Fourteenth International Conference on Very Large Data Bases, August 29 - September 1, 1988, Los Angeles, California, USA, Proceedings*, pages 195–205. Morgan Kaufmann, 1988.

[2] W. B. Arthur. Inductive reasoning and bounded rationality. *American Economic Review*, 84:406–411, 1994.

[3] Sujata Banerjee, Sujoy Basu, Shishir Garg, Sukesh Garg, Sung-Ju Lee, Pramila Mullan, and Puneet Sharma. Scalable grid service discovery based on uddi. In *MGC '05: Proceedings of the 3rd international workshop on Middleware for grid computing*, pages 1–6, New York, NY, USA, 2005. ACM Press.

[4] N. Bertschinger and T. Natschlager. Real-time computation at the edge of chaos in recurrent neural networks. In *Neural Computation*, volume 16, pages 1413 – 1436, 2004.

[5] U. Bhatti, S. Youcef, L. Mokdad, and V. Monfort. Simulation-based response-time analysis of composite web services. In *Proceedings 10th IEEE international Multitopic conference, INMIC'06*, pages 1–6, 2006.

[6] E. Bonabeau. Predicting the unpredictable. *Harvard Business Review*, 80:1–9, March 2002.

[7] E. Bonabeau, A. Sobkowski, G. Theraulaz, and J. Deneubourg. Adaptive task allocation inspired by a model of division of labor in social insects. In Dan Lundh, Bjorn Olsson, and Ajit Narayanan, editors, *Biocomputing and Emergent Computation*, pages 36–45. World Scientific, 1997.

[8] F. Boschetti, J. Finnigan, P. Valencia, I. Enting, G German, and D Newth. Does anything emerge? 2005.

[9] Craig Boutilier, Rajarshi Das, Jeffrey Kephart, Gerald Tesauro, and William Walsh. Cooperative negotiation in autonomic systems using incremental utility

elicitation. In *Proceedings of the 19th Annual Conference on Uncertainty in Artificial Intelligence (UAI-03)*, pages 89–97, San Francisco, CA, 2003. Morgan Kaufmann.

[10] F. M. T. Brazier, M. van Steen, and N. J. E. Wijngaards. On MAS scalability. In T. Wagner and O. Rana, editors, *Proceedings of Second International Workshop on Infrastructure for Agents, MAS, and Scalable MAS*, pages 121–126, 2001.

[11] S. Brueckner and H. V. D. Parunak. Self-organizing MANET management. In G. Di Marzo, A. Karageorgos, O. F. Rana, and F. Zambonelli, editors, *Engineering Self-Organising Systems*, pages 1–16. Springer, 2003.

[12] S. A. Brueckner and H. V. D. Parunak. Information-driven phase changes in multiagent coordination. In *Proceedings of the second international joint conference on Autonomous Agents and Multiagent Systems*, pages 950–951, NY, USA, 2003. ACM Press.

[13] M. Jacyno S. Bullock. Energy, entropy and work in computational ecosystems: A thermodynamic account. In *In Artificial Life XI: Proceedings of the Eleventh International Conference on the Simulation and Synthesis of Living Systems*, pages 274–281. MIT Press, Cambridge, MA., 2008.

[14] S. Bullock and D. Cliff. Complexity and emergent behaviour in ICT systems. Technical Report HP-2004-187, Hewlett-Packard, 2004.

[15] C. Castelfranchi. Engineering social order. In *Lecture notess In Computer Science*, volume 1972 of *Proceedings of the First International Workshop on Engineering Societies in the Agent World: Revised Papers*, pages 1–18, 2000.

[16] V. Cicirello and S. Smith. Ant colony control for autonomous decentralized shop floor routing. In *ISADS-2001: Fifth International Symposium on Autonomous Decentralized Systems*, pages 383 – 390. IEEE Computer Society, March 2001.

[17] T. Czerwinski. *Coping with the bounds. Speculations on nonlinearity in military affairs.* Institute for National Strategic Studies at the National Defense University (NDU), 1998.

[18] J. Dobrzanski and J. Dobrzanska. About the joint action by ants: collaboration or participation? *Panstwowe Wydawnictwo Naukowe*, 36:61–75, 1987.

[19] E. H. Durfee. Scaling up agent coordination strategies. *IEEE Computer*, 34(7):1–8, 2001.

[20] D. Estrin, D. Culler, K. Pister, and G. Sukhatme. Connecting the physical world with pervasive networks. *IEEE Pervasive Computing*, 1(1):59–69, 2002.

[21] S. Forrest, J. Balthrop, and M. Glickman. *Internet as a Large - Scale Complex System*, chapter Computation in the Wild, pages 203–227. Oxford University Press, 2003.

[22] S. Forrest, A. Somayaji, and D. H. Ackley. Building diverse computer systems. *Sixth Workshop on Hot Topics in Operating Systems*, 1997.

[23] I. Foster, N. R. Jennings, and C. Kesselman. Brain meets brawn: Why grid and agents need each other. In *Proceeding of the 3rd International Conference on Autonomous Agents and Multi-Agent Systems*, pages 8–15, New York, USA, 2004.

[24] M. Gambhir, S. Guerin, S. Kauffman, and D. Kunkle. Steps toward a possible theory of organization. In Y. Bar-Yam A. Minai, editor, *Proceedings of International Conference on Complex Systems*, page 9. W. H. Freeman, 2004.

[25] C. Gershenson and F. Heylighen. *How can we think the complex?*, volume 1 of *Managing Organizational Complexity: Philosophy, Theory and Application*, chapter 3, pages 1–14. Information Age Publishing, 2005.

[26] N. Glance, T. Hogg, and B.A. Huberman. Computational ecosystems in a changing environment. *International Journal of Modern Physics*, 2:735–753, 1991.

[27] D. Goldin and D. Keil. Toward domain-independent formalization of indirect interaction interaction. In *Proceedings of the 13th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WET-ICE'04)*, volume 0, page 1, Washington, DC, USA, 2004. IEEE Computer Society.

[28] D.M. Gordon. The organization of work in social insect colonies. *Complex.*, 8(1):43–46, 2002.

[29] S. Guerin and D. Kunkle. Emergence of constraint in self-organizing systems. *Journal of Nonlinear Dynamics, Psychology, and Life Sciences*, 8:2:131–146, 2004.

[30] D. Hales. Cooperation without space or memory:tags, groups and the prisoner's dilemma. In I.M. Davidsson, editor, *Multi-Agent-Based Simulation*, page 9, Berlin, 2000. LNAI.

[31] I. Harvey and T. Bossomaier. Time out of joint: Attractors in asynchronous random Boolean networks. In P. Husbands and I. Harvey, editors, *Proceedings of the Fourth European Conference on Articial Life*, pages 67–75. MIT Press, Camrbidge, MA., 1997.

[32] C. Hewitt. Offices are open systems. In *ACM Transactions on Information Systems (TOIS)*, volume 4, pages 271–287, 1986.

[33] F. Heylighen. Self-organization, emergence and the architecture of complexity. In *Proceedings of the 1st European Conference on System Science (AFCET)*, pages 23–32, Paris, 1989.

[34] F. Heylighen. Cognitive levels of evolution: from pre-rational to meta-rational. In F. Geyer, editor, *The Cybernetics of Complex Systems - Self-organization, Evolution and Social Change*, pages 75–91, Intersystems, California, 1991.

[35] F. Heylighen. Modelling emergence. *World Futures: the Journal of General Evolution, special issue on creative evolution*, pages 1–10, 1991.

[36] F. Heylighen. Principles of systems and cybernetics: an evolutionary perspective. *Cybernetics and Systems*, 1992.

[37] F. Heylighen. *The Science Of Self-Organization And Adaptivity*, volume 5, chapter The Encyclopedia of Life Support Systems Heylighen F. (1999): Collective Intelligence and its Implementation on the Web: algorithms to develop a collective mental map, Computational and Mathematical Theory of Organizations 5(3), 253-280.",, pages 253–280. 1999.

[38] F. Heylighen and C. Gershenson. The meaning of self-organization in computing. *IEEE Intelligent Systems*, pages 1–6, 2003.

[39] F. Heylighen and C. Joslyn. Cybernetics and second-order cybernetics. In R. Meyers, editor, *Encyclopedia of Physical Science and Technology*, volume 4, pages 155–170. Academic Press, New York, 2001.

[40] T. Hogg and B. A. Huberman. Controlling chaos in distributed systems. *IEEE Transactions on Systems, Man and Cybernetics*, 21:1325–1332, 1991.

[41] T. Hogg and B. A. Huberman. Dynamics of large computational ecosystems. Technical Report HPL-2002-77, Information Dynamics Laboratory, Hewlett-Packard, Palo Alto, 2002.

[42] P. Horn. Autonomic computing manifesto. Technical report, IBM, http://www.research.ibm.com/autonomic/manifesto/, 2001.

[43] B. A. Huberman and N. S. Glance. Evolutionary games and computer simulations. *Proc. Natl. Acad. Sci, USA*, 90:7716–7718, 1993.

[44] B. A. Huberman and T. Hogg. In Lynn Nadel and Daniel Stein, editors, *Lectures in Complex Systems*, chapter The Emergence of Computational Ecologies, pages 185–205. Addison-Wesley, 1993.

[45] B. A. Huberman and J. O. Ledyard. Information dynamics in the networked world. volume 5, pages 7–8, Hingham, MA, USA, 2003. Kluwer Academic Publishers.

[46] Gartner Inc. Gartner says 50 percent of data centers will have insufficient power and cooling capacity by 2008. *Press Release*, November 29,2006.

[47] M. Jacyno, S. Bullock, M. Luck, and T. Payne. Understanding decentralized control of resource allocation in a minimal multi-agent system. In *In Proceedings of the International Conference on Autonomous Agents and Multi-Agent Systems.*, 2007.

[48] S. Jain and S. Krishna. Emergence and Growth of Complex Networks in Adaptive Systems. In *eprint arXiv:adap-org/9810005*, pages 10005–+, October 1998.

[49] S. Jaina and S. Krishna. Graph theory and the evolution of autocatalytic networks. 2002.

[50] E. T. Jaynes. Gibbs vs Boltzmann entropies. *American Journal of Physics*, 33(5):620–630, 1965.

[51] E. T. Jaynes. Where do we stand on maximum entropy? In R. D. Levine and M. Tribus, editors, *The Maximum Entropy Formalism*, pages 15–127. MIT Press, 1979.

[52] E. T. Jaynes. The evolution of Carnot's principle. In G. J. Erickson and C. R. Smith, editors, *Maximum-Entropy and Bayesian Methods in Science and Engineering*, pages 267–284. Kluwer, 1988.

[53] N. R. Jennings. An agent-based approach for building complex software systems. *Communications of the ACM*, 44(4):35–41, 2001.

[54] S. Kauffman. *At Home in the Universe: The Search for the Laws of Self-Organization and Complexity*. Oxford University Press, 1995.

[55] S. Kauffman. *Investigations*. Oxford University Press, 2000.

[56] J. J. Kay. *Self-Organization In Living Systems*. PhD thesis, Deparment of Systems Design Engineering, University of Waterloo, 1984.

[57] D. Keil and D. Goldin. Modelling indirect interaction in open computational systems. In *Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises*, pages 1–6, 2004.

[58] J. O. Kephart and D. M. Chess. The vision of autonomic computing. *IEE Computer*, 36(1):41–50, 2003.

[59] J. O. Kephart and R. Das. Achieving self-management via utility functions. *IEEE Internet Computing*, 11(1):40–48, 2007.

[60] J. O. Kephart, T. Hogg, and B. A. Huberman. Dynamics of computational ecosystems. *Physical Review*, 40(1):1–18, 1989.

[61] J.O. Kephart, H. Chan, R. Das, D.W. Levine, G.Tesauro, F. Rawson, and C. Lefurgy. Coordinating multiple autonomic managers to achieve specified power-performance tradeoffs. In *ICAC '07: Proceedings of the Fourth International Conference on Autonomic Computing*, page 24, Washington, DC, USA, 2007. IEEE Computer Society.

[62] J. G. Koomey. Estimating total power consumption by servers in the u.s. and the world. *http://enterprise.amd.com/Downloads/svrpwrusecompletefinal.pdf*, 2006.

[63] A. J. Lotka. Contribution to the energetics of evolution. *PNAS USA*, 8:145–151, 1922.

[64] A. J. Lotka. Natural selection as a physical principle. *PNAS USA*, 8:151–154, 1922.

[65] M. Luck M. Jacyno, S. Bullock and T.R. Payne. Emergent service provisioning and demand estimation through self-organizing agent communities. In *In The Eighth International Conference on Autonomous Agents and Multiagent Systems*, 2009.

[66] T.R. Payne N. Geard M. Jacyno, S. Bullock and M. Luck. Autonomic resource management through self-organising agent communities. In *In Second IEEE International Conference on Self-Adaptive and Self-Organizing Systems*, 2008.

[67] C. M. MacKenzie, K. Laskey, F. McCabe, P.F. Brown, and R. Metz. Reference model for service oriented architecture 1.0. Technical report, OASIS, http://docs.oasis-open.org/soa-rm/v1.0/soa-rm.html, 2006.

[68] T. W. Malone and K. Crowston. The interdisciplinary study of coordination. *ACM Computing Surveys*, 26(1):87–119, 1994.

[69] M. Mamei, A. Roli, and F. Zambonelli. Emergence and control of macro spatial structures in perturbed cellular automata, and implications for pervasive computing systems. In *IEEE Transactions on Systems, Man, and Cybernetics - Part A*, volume 35, pages 337–348. 2005.

[70] M. Mamei, M. Vasirani, and F. Zambonelli. Experiments of morphogenesis in swarms of simple mobile robots. *Journal of Applied Artificial Intelligence*, 18(9-10):21, October 2004.

[71] M. Mamei and F. Zambonelli. *Self-* Approaches to Distributed Computing*, chapter Spatial Computing: the TOTA Approach, page 17. LNCS Hot Topics Series. Springer Verlag, 2005.

[72] F. Marinescu. *Ejb Design Patterns: Advanced Patterns, Processes, and Idioms with Poster*. John Wiley & Sons, Inc., New York, NY, USA, 2002.

[73] D. Merkle and M. Middendorf. Dynamic polyethism and competition for tasks in threshold reinforcement models of social insects. *Adaptive Behavior - Animals, Animats, Software Agents, Robots, Adaptive Systems*, 12(3-4):251–262, 2004.

[74] M.D. Mesarovic, S.N. Sreenath, and J.D. Keene. Search for organizing principles: understanding in systems biology. *The IIE Systems Biology*, 1(1):19–27, June 2004.

[75] C. Mohan. Dynamic e-business: Trends in web services. In *TES '02: Proceedings of the Third International Workshop on Technologies for E-Services*, pages 1–5, London, UK, 2002. Springer-Verlag.

[76] S. Abdelwahed N. Kandasamy and J. Hayes. Self-optimization in computer systems via on-line control: Application to power management. In *ICAC '04: Proceedings of the First International Conference on Autonomic Computing*, pages 54–61, Washington, DC, USA, 2004. IEEE Computer Society.

[77] R. Nagpal. Programmable self-assembly using biologically-inspired multiagent control. In *Proceedings of the first international joint conference on Autonomous agents and multiagent systems (AAMAS)*, pages 418–425, New York, NY, USA, 2002. ACM Press.

[78] P. Nelson. *Biological Physics: Energy, Information, Life*. W. H. Freeman, 2004.

[79] M.E.J. Newman. Modularity and community structure in networks. In *Proceedings of the National Academy of Science*, page 23, 2006.

[80] G. Nicolis and I. Prigogine. *Self-organization in Non-equilibrium Systems: From Dissipative Structures to Order Through Fluctuations*. J. Wiley & Sons, 1977.

[81] A. Omicini. Soda: Societies and infrastructures in the analysis and design of agent-based systems. In *SODA: Societies and Infrastructures in the Analysis and Design of Agent-Based Systems*, pages 185–193, New York, 2000. SODA: Societies and Infrastructures in the Analysis and Design of Agent-Based Systems.

[82] A. Omicini, A. Ricci, M. Viroli, C. Castelfranchi, and L. Tummolini. Coordination artifacts: Environment-based coordination for intelligent agents. In *3rd International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 286–293, New York, 2004. IEEE Computer Society.

[83] A. Omicini, A. Ricci, M. Viroli, and G. Rimassa. Integrating objective and subjective coordination in mult-agent systems. In *Proceedings of the 2004 ACM symposium on Applied computing*, pages 449–455. ACM Press, 2004.

[84] A. Omicini and F. Zambonelli. MAS as complex systems: A view on the role of declarative approaches. In L. Sterling J. Leite, A. Omicini, editor, *Declarative Agent Languages and Technologies: First International Workshop, DALT*, volume

2990 of *Lecture notess in Computer Science*, pages 1–16. Springer-Verlag GmbH, July 2003.

[85] N. H. Packard. Adaptation toward the edge of chaos. In A.J. Mandell J.A. Kelso and M.F. Shlesinger, editors, *Dynamic patterns in complex systems*, pages 293–301. World Scientific, 1988.

[86] P. Panzarasa and N. R. Jennings. The organisation of sociality: a manifesto for a new science of multi-agent systems. In *Proceedings of the 10th European Workshop on Multi-Agent Systems*, 2001.

[87] M. P. Papazoglou and Willem-Jan Heuvel. Service oriented architectures: approaches, technologies and research issues. *The VLDB Journal*, 16(3):389–415, 2007.

[88] H. Van Dyke Parunak. Go to the ant: Engineering principles from natural multi-agent systems. *Annals of Operations Research*, page 27, 1997.

[89] H. Van Dyke Parunak. The process-interface-topology mode overlooked issues in modeling social systems. In *MASHO Workshop at ECAI*, pages 1–7, 2000.

[90] H. Van Dyke Parunak and S. Brueckner. Entropy and self-organization in multi-agent systems. In S. Sen C. Frasson J. Müller, E. Andre, editor, *Proceedings of the fifth international conference on Autonomous agents*, pages 124–130. ACM Press, 2001.

[91] H. Van Dyke Parunak and S. A. Brueckner. Engineering swarming systems. In F. Bergenti, M.-P. Gleizes, and F. Zambonelli, editors, *Methodologies and Software Engineering for Agent Systems*, pages 341–376. Kluwer, 2004.

[92] H. Van Dyke Parunak, S. A. Brueckner, and J. Sauter. ERIMs approach to fine-grained agents. In *NASA Workshop on Radical Agent Concepts*, pages 1–10, Greenbelt, MD, USA, September19-21 2002.

[93] H. Van Dyke Parunak, S.A. Brueckner, J.A. Sauter, and M. Robert. Global convergence of local agent behaviours. In *Submitted to the Fourth International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS05)*, Utrecht, Netherlands, 2005.

[94] H. Van Dyke Parunak and Sven A. Brueckner. Analyzing stigmergic learning for self-organizing mobile ad-hoc networks (manets). In *Engineering Self-Organising Systems*, 2004.

[95] H. Van Dyke Parunak, Robert Savit, Sven A. Brueckner, and John Sauter. Experiments in indirect negotiation. In *AAAI Fall Symposium on Negotiation Methods for Autonomous Cooperative Systems*, page 9, 2001.

[96] F. Polack and S. Stepney. Emergent properties do not refine. In *REFINE workshop*, Electronic notes in Theoretical Computer Science, pages 1–17, Guildford, UK, April 2005. Elsevier.

[97] D. Pynadath and M. Tambe. The communicative multiagent team decision problem: analyzing teamwork theories and models. *Journal of Artificial Intelligence Research*, 16(1):389–423, 2002.

[98] C. Lefurgy G. Tesauro D. Levine R. Das, J. Kephart and C. Hoi. Autonomic multi-agent management of power and performance in data centers. In *The Seventh International Conference of Autonomic Agents and Multiagent Systems*, May 2008.

[99] G E Robinson. Regulation of division of labor in insect societies. In *Annual Review of Entomology*, volume 37, pages 637–665, 1992.

[100] E. Roman, S. W. Ambler, and F. Marinescu. *Mastering Enterprise Javabeans*. John Wiley & Sons, Inc., New York, NY, USA, 2001.

[101] D. De Roure. On self-organization and the semantic grid. *IEEE Intelligent Systems*, 18:77–79, 2003.

[102] E. Sanchis. Systemions: a model for open agents. In *13th IEEE International Workshops on Enabling Technologies (WETICE 2004)*, Infrastructure for Collaborative Enterprises, Modena, Italy, 14-16 June 2004. IEEE Computer Society.

[103] E. D. Schneider and J. J. Kay. Order from disorder: The thermodynamics of complexity in biology. In M. P. Murphy and L. O'Neill, editors, *What Is Life: The Next Fifty Years. Reflections on the Future of Biology*, pages 161–172. Cambridge University Press, 1995.

[104] F. Schurmann, K. Meier, and J. Schemmel. Edge of chaos computation in mixed-mode vlsi - a hard liquid. In *Neural Information Processing Systems Conference*, 2004.

[105] T. D. Seeley. When is self-organization used in biological systems? *Biological Bulletin*, 202:314–318, 2002.

[106] S. Sen, S. Roychowdhury, and N. Arora. Effects of local information on group behavior. In *Proceedings of the Second International Conference on Multi-Agent Systems*, pages 315–321. AAAI Press, Menlo Park, CA, 1996.

[107] C. R. Shalizi. *Causal Architecture, Complexity and Self-organization in Time Series and Cellular Automata*. PhD thesis, University of Wisconsin, Physics Deptartment, 2001.

[108] C. E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27:379–423 and 623–656, 1948.

[109] O. Shehory and S. Kraus. Methods for task allocation via agent coalition formation. *Artificial Intelligence*, 101:165–200, 1998.

[110] Y. Oono S.N. Beshers, Z.Y. Huang and G.E. Robinson. Social inhibition and the regulation of temporal polyethism in honey bees. In *Journal of Theoretical Biology*, volume 213, pages 461–479(19). Academic Press, 2001.

[111] K. Sreekala and N.J. Vriend. Is the study of complex adaptive systems going to solve the mystery of adam smith's "invisible hand"? *The Independent Review*, 3(1):53–66, 1998.

[112] S. Staab, F. Heylighen, C. Gershenson, G. W.Flake, D. M. Pennock, D. C. Fain, D. De Roure, K. Aberer, W. Shen, O. Dousse, and P. Thiran. Neurons, viscose fluids, freshwater polyp hydra-and self-organizing information systems. *IEEE Intelligent Systems*, 18(04):72–86, 2003.

[113] S. Stepney. Critical critical systems. In Steve Schenider Ali E. Abdallah, Peter Ryan, editor, *Formal Aspects of Security: FASec*, volume 2629 of *Lecture notess in Computer Science*. Springer, 2003.

[114] P. Stone and M. Veloso. Layered learning and flexible teamwork in robocup simulation agents. *Lecture Notes In Computer Science*, 1856:495 – 508, 2000.

[115] G.J. Sussman. Robust design through diversity. In *Workshop on Amorphous Computing*, page 3, Cambridge, MA, September 13-14 1999.

[116] R. Swenson. Autocatakinetics, evolution, and the law of maximum entropy production: A principled foundation towards the study of human ecology. *Advances in Human Ecology*, 6:1–47, 1997.

[117] R. Swenson and M.T Turvey. Thermodynamic reasons for perception-action cycles. *Ecological Psychology*, 3(4):317–348, 1991.

[118] A.S. Tanenbaum and R. Van Renesse. Distributed operating systems. *ACM Comput. Surv.*, 17(4):419–470, 1985.

[119] G. Tesauro. Reinforcement learning in autonomic computing: A manifesto and case studies. *IEEE Internet Computing*, 11(1):22–30, 2007.

[120] G. Theraulaz, E. Bonabeau, and J. Deneubourg. Response threshold reinforcement and division of labour in insect societies. In *Proceeding of the Royal Society of London*, pages 327–332, 1998.

[121] G. Theraulaz and E. Bonbeau. A brief history of stigmergy. *Artificial Life*, 5(2):97–116, 1999.

[122] M. Tian, T. Voigt, T. Naumowicz, H. Ritter, and J. Schiller. Performance impact of web services on internet servers. 2003.

[123] P. J. Turner and N. R. Jennings. Improving the scalability of multi-agent systems. In *1st International Workshop on Infrastructure for Scalable Multi-Agent Systems*, Barcelona, Spain, 2000.

[124] C. Vawter and E. Roman. J2ee vs. microsoft.net: A comparison of building xml-based web services. Technical report, Sun Microsystems, Inc., 2001.

[125] M. Wang, N. Kandasamy, A. Guez, and M. Kam. Distributed cooperative control for adaptive performance management. *IEEE Internet Computing*, 11(1):31–39, 2007.

[126] H.F. Wedde and M. Lischka. Cooperative role-based administration. In *SACMAT '03: Proceedings of the eighth ACM symposium on Access control models and technologies*, pages 21–32, New York, NY, USA, 2003. ACM.

[127] J. S. Wicken. Evolution, thermodynamics, and information: Extending the darwinian program. *The Quarterly Review of Biology*, 63(1):84–85, 1988.

[128] J. S. Wicken. Evolution and thermodynamics: The new paradigm. *Systems Research and Behavioral Science*, 6(3):181–186, 1989.

[129] T. De Wolf and T. Holvoet. Emergence and self-organisation: a statement of similarities and differences. In S. Brueckner, G. Di Marzo Serugendo, A. Karageorgos, and R. Nagpal, editors, *Proceedings of the International Workshop on Engineering Self-Organising Applications*, pages 96–110, 2004.

[130] M. Wooldridge, N.R. Jennings, and D. Kinny. The Gaia methodology for agent-oriented analysis and design. *Autonomous Agents and Multi-Agent Systems*, 3(3):285–312, 2000.

[131] F. Zambonelli, M. Mamei, and A. Roli. What can cellular automata tell us about the behavior of large multi-agent systems? In A. F. Garcia, C. J. P. de Lucena, F. Zambonelli, A. Omicini, and J. Castro, editors, *Software Engineering for Large-Scale Multi-Agent Systems, Research Issues and Practical Applications*, volume 2603 of *Lecture notess in Computer Science*, pages 216–231. Springer, 2002.

[132] F. Zambonelli and A. Omicini. Challenges and research directions in agent-oriented software engineering. *Autonomous Agents and Multi-Agent Systems*, 9(3):253–283, 2004.

[133] F. Zambonelli and H. Van Dyke Parunak. Signs of a revolution in computer science and software engineering. In *2 nd Italian Workshop on Objects and Agents*, pages 1–12, 2001.

[134] F. Zambonelli and H. Van Dyke Parunak. Towards a paradigm change in computer science and software engineering: A synthesis. *The Knowledge Engineering Review*, 18(4):329–342, 2004.