Modelling Interactive Real-time Applications on Service Oriented Infrastructures

Matthew Addis, Zlatko Zlatev, Bill Mitchell, Mike Boniface

University of Southampton IT Innovation Centre, Southampton, UK

E-mail: {mja, zdz, mjb, wm}@it-innovation.soton.ac.uk

Abstract: **The European Commission supported IRMOS project is developing tools and techniques that allow real-time applications to be planned and executed on distributed Service Oriented Infrastructures (SOI) operated by third-party service providers. The exemplar applications within the project are all multimedia based and include support for interactive and collaborative film post-production, the use of virtual and augmented reality within the engineering design process, and the use of 3D virtual worlds as interactive online eLearning environments. In each case, there is a need for well defined and managed Service Level Agreements that have stringent Quality of Service (QoS) terms referring to applications hosted on third-party virtualised resources (storage, processing, networking). This paper presents techniques developed within IRMOS for modelling and predicting the resource and QoS requirements of interactive media applications on SOIs. These models have value in many stages of the application lifecycle, for example when estimating resource needs in advance of execution, when negotiating QoS with service providers, when assessing the probable technical and economic outcomes of provisioning policies and management actions if either the application or resources do not perform as expected or need to be adjusted.**

Keywords: real time, modelling, service oriented, interactive, multimedia, quality of service, performance, prediction

1 OVERVIEW

The IRMOS project [1] is developing tools and techniques for modelling, simulating, analysing, and planning interactive real-time applications on service oriented infrastructures. These tools and techniques support the processes involved in designing, developing, deploying and executing applications where guaranteed QoS is needed.

This paper reviews the value-chain for real-time applications hosted by third-party service providers. In the context of this value chain, we then analyse who might benefit from the use of models, how and why these models might be used, and when during the application lifecycle modelling is most useful.

Techniques are then presented for building models of real-time applications including the use of stochastic process algebras and finite state automata. We then focus on the use stochastic process algebras and finite state automata and show their practical application using an example real-time application scenario based on collaborative film post production.

2 BACKGROUND

New value chains are emerging for outsourced hosting and execution of interactive media applications that have strict requirements on quality of service in order to operate effectively (e.g. latency, bandwidth and jitter for video streaming, or processing power for interactive special effects rendering). Actors in the value chain emerge where value can be added, e.g. at the infrastructure level this might be providing virtualised storage, networking and compute resources using a Infrastructure as a Service (IaaS) model, or at the application level it might be offering a suite of post-production tools on a pay-per-use basis using a Software as a Service (SaaS) model. In-between these two levels, comes the possibility of Platform as a Service (PaaS), for example where a provider makes it easy for a developer to build and deploy new applications on top of service oriented infrastructures (Google Apps [3] and Microsoft Azure [2] being examples).

The value chain is shown in Figure 1 highlighting where Service Level Agreements feature at the infrastructure and application levels.

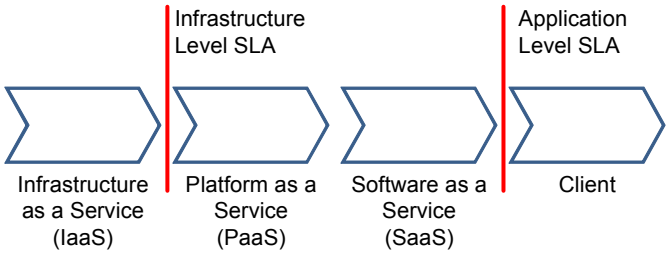

Figure 1 Value chain for applications on service oriented infrastructure.

Applications with real-time attributes (e.g. the need for guaranteed completion by a particular deadline) require careful planning when selecting service providers so that neither under-provisioning (likely failure of the application to execute) nor massive over-provisioning (unnecessarily high costs) occur. This planning, especially when there is uncertainty involved, is not currently well supported in conventional tools. From a real-time application modelling perspective, the important aspects of planning in these new value chains, and hence the need for modelling, is the

Corresponding author: Matthew Addis, IT Innovation Centre, 2 Venture Road, Chilworth, Southampton, UK, + 44 (0) 23 80 760 834

separation of the resource-level (storage, processing, networking) parameters from the application-level parameters (e.g. video effects processing).

At the application level, the parameter space can be large and complex, even for simple applications. For example, as shown in Figure2, colour correction as a component application in film post production is characterised by a large number of parameters, including the characteristics of the film to be corrected, the corrections to be applied, and the use of colour correction within a workflow. Likewise at the hardware resource level the space is equally large and complex due to the details of architectures and configurations, e.g. buses and caching, memory, networking, processors, storage.

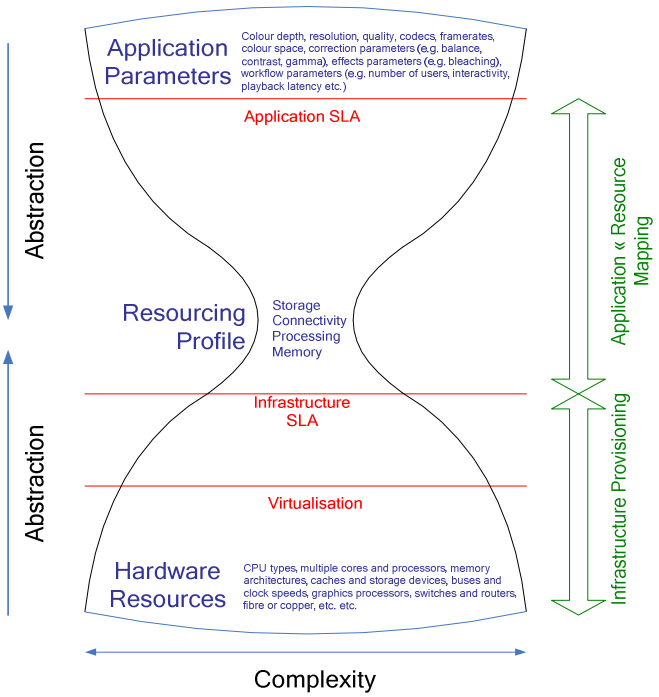

Figure 2 Mapping of applications to resources and the role of virtualisation in service provision.

The role of the SaaS provider is particularly interesting. The SaaS provider uses domain knowledge to abstract application-level complexity and characterise the application execution requirements in terms of its need for storage, processing, memory and networking. It delivers value to the client in terms of abstraction from how the application is resourced and does so through a service that is in terms that the Client understands (e.g. video rendering at a particular frame rate). However, with this comes the risk from a SaaS provider perspective of not provisioning the application cost effectively. The behaviour and execution time of applications and workflows is often uncertain, which ia a significant risk for providers who on the one hand need to meet QoS commitments to their Clients but on the other hand need to minimise their outlay on resources procured from IaaS providers. The challenge is how much abstraction can be sustained by the SaaS provider? The larger the abstraction then the more uncertainty and hence the more risk the provider takes when delivering guaranteed application QoS. To consider and manage this risk the SaaS provider needs to use methods that provide accurate mapping between the Client and the IaaS provider worlds and models that assess the performance of the services offered.

The approach to performance modelling in IRMOS is based on discrete time stochastic finite state automata. The business workflows within which one or more applications are used (e.g. in film production a workflow might be dailies review, colour correction, special effects rendering and audio correction) are modelled as a number synchronised timed stochastic finite state automata, one or more per application. These models are to be used for estimating the completion time of a workflow and of the individual activities within the workflows, i.e. to experiment with questions of the form: 'if I am constrained to use only these resources for this activity when what is the probability of completing the activity within a given time span?' This formalism is one that can be analytically investigated using the PRISM model checking tool [5], [6].

The rest of this paper focuses on the proposed approach to performance modelling, including the worked example in the results, namely how can we build models that predict completion time of a workflow given a particular level of QoS for the resources used to execute the application and given particular users' behaviour.

3 APPROACH

Process algebras, such as CCS and CSP, have been used for some time as an algebraic means of describing interleaving behaviour of communicating distributed systems at an abstract level that provided a mechanism for formal verification. For SOI however there are some shortcomings of these original algebras. For example, traditional process algebras are not well suited to describing compensation or long running transactions and it is difficult to adequately handle composition of services in an elegant way. Some calculi have been developed in an attempt to address these issues [7],[8] Other calculi have been developed in order to address some of the wider issues of service behaviour [9],[10]. Verifying properties of choreography of web services is also an area where bespoke process algebras have been developed, [11],[12] Another current effort along these lines is Stochastic COWS[13],[14] process algebra. This is one of several process algebras that adopt the approach of generating a continuous time Markov chain (CTMC) from a process algebra term. In our case, we choose to use discrete time probabilistic finite state automata with time constraints defined with respect to discrete clocks. This allows parallel composition of automata, e.g. so complex processes can be described, and also model checking that can be automated when the model is guaranteed finite branching. The later is important as it is possible to automate model checking for temporal properties, e.g. to determine whether the system has a finite probability of completing in all circumstances. This has been studied in [15], [16].

In IRMOS we assume that the SaaS are not applications written specifically for operation as IRMOS services, but

rather, software applications already in general use wrapped up as SaaS applications. As a consequence the actual internal operation of the application will be very difficult to be ascertained and used for the purposes of performance modelling, i.e. it is not feasible to create a performance model accurately representing the internal application behaviour. What we can do is to use background knowledge, possibly acquired experimentally, of the externally observed application behaviour, the behaviour of the execution environment of the application and the pattern of the user interactions with the application, and to create a statistical high level performance model incorporating the different behavioural aspects. We use the discrete time stochastic finite state machines (FSM) modelling technique to create the statistical high level performance model.

For IRMOS we derived a customisable generic FSM model, depicted in Figure 3, to guide the service analyst in the process of identifying the model states and transitions. The states of this FSM are high-level externally observed states and the transitions are stochastic, i.e. are triggered according to probability distributions. These distributions can be derived by observation or experimentation and statistical analysis of the transition frequencies from one observable state to another.

The model has three macro-states corresponding to three main application behavioural aspects:

- Uninterrupted-fault-free application operation
- User and application interrupts
- Infrastructure faults (including critical and due to QoS degradation faults)

The macro-states are common for all modelled applications. Each macro-state includes sub-states (referred to just as 'states' from here onwards) which in general can be specific for each modelled service application.

To keep the model with as less states as possible (the model execution time increases exponentially with respect to the number of states in the FSM) we suggest maximum of three states when modelling the normal (*uninterrupted-fault-free*) application behaviour, namely:

- Processing initiation
- Processing
- Processing wrap-up

Processing initiation could be e.g. setting up some footage dirt-removal application parameters after previewing a set of frames. *Processing wrap-up* could be final reviewing of the whole footage after dirt-removal. *Processing* is the dirt-removal of the whole sequence of frames assuming no fault or interruption occurrence. Probability distributions of transitions between states included in the normal application behaviour macro-state are obtained by general experimentation and benchmarking activities. Application runs can be performed for a series of workloads, with different application setup, on a variety of platforms in order to obtain an application uninterrupted-fault-free completion time estimator.

For creating the states in the other two FSM macro-states an analyst will need to use any user, system, application and infrastructure knowledge available. In IRMOS we intend to compile and provide lists of optional states to be included in these macro-states. For example, for the *infrastructure faults* macro-state we might suggest the states of:

- QoS degradation fault: bandwidth below required minimum value
- Critical fault: virtual machine (VM) crash

And for the *user/application interrupts* macro-state we might suggest the states of:

- User interrupt: quality assurance (QA) interrupt
- Application interrupt: additional I/O for process flow control

It is assumed that after QoS degradation fault only a temporal delay is encountered, but after a critical fault a new process initiation is needed. The user and application interrupts only introduce additional delays. From the suggested list the analyst shall choose only the states that are relevant to the modelled application.

The probability distributions of transitions between states included in the faults or interrupts macro-states can be obtained from background knowledge or by statistical modelling of the user, the application and the infrastructure behaviour (e.g. parametric statistical modelling and/or Bayesian Believe Networks).

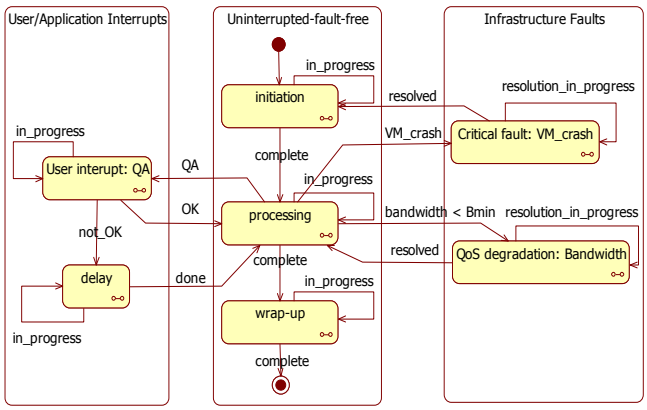

Figure 3 Generic service application FSM. All the events are probabilistic and some are synchronising with a local clock FSM, which is not depicted to avid clutter.

The timed behaviour of the FSM is facilitated by an additional state machine that models local time. The FSM depicted in Figure 3 will in practice have additional states and transitions for synchronising with this local time FSM, which are not depicted in order to avoid clutter. The synchronising states and transitions are shown in the worked example described in the following section.

4 RESULTS

In this section we use a simplified and hypothetical scenario of colour correction as part of film post-production workflow. Imagine that a post-production house is contracted to perform

colour correction to some film shots that will be selected by a film director during his preview of the digital dailies of a film currently under production. The footage length needing colour correction cannot be determined in advance as this depends on the output from the shooting session. It is estimated that colour correction will be applied to approximately 40 +/- 10 minutes of footage. The director needs to preview the footage 1, 2 or 3 times in order to make a decision on the manner of colour correction, with number of previewing iterations assumed to be equally probable. An IaaS provider provides storage, processing and networking resources for the post-production workflow. The footage is stored in IaaS storage and is streamed to the director for a preview. If the link bandwidth drops below a critical value, streaming is not possible and previewing stops. After a streaming failure the director will need to go back and preview on average about 5 minutes of footage and it takes another about 5 minutes to recommence previewing, i.e. there is on average 10 minutes delay. From experience we know that the streaming failures occur with rate of 2 per hour. The colour correction is performed on an application installed on an IaaS computing resource. The colourist has a client side application component which enables real-time interaction with the colour correction application running on the IaaS resource. The colour correction is done in real-time and the corrected footage is streamed to the director for in-time feedback. It is estimated that on average the director will ask for colour correction interrupt every 30 minutes and colour regarding of the last 5 minutes will need to be done, where it takes the colourist another 5 minutes to reset the application.

The above scenario is modelled with three state machines which are grouped in two workflow macro states that indicate the activities of *Director Preview* and *Colourist Colour Correction* that are to be performed. Figure 4 depicts the model in the form of a UML state machine diagram.

Note that the *Preview* and *Colour Correction* finite state machines are customisations of the generic finite state machine as depicted in Figure 3. In addition, there is an implicit time clock state machine to model local model time, which executes in parallel with the activities state machines. The time clock state machine is not included in the diagram as the only event we are interested in is the *clockTick* event which signals the end of each elapsed time unit. Each activity state machine is synchronised with the *clockTick* event of the time state machine in order to count the activity elapsed time. The synchronisation is facilitated by the transitions denoted by the *<<synchronising>>* stereotype. The stochastic behaviour of the state machines is facilitated by the transition denoted by the *<<stochastic>>* stereotype, i.e. these transitions are triggered according to some probability distribution as a function of time. For example, when previewing footage the probability to have more footage to preview at time t is $P\{T_{pr} > t | T_{pr} > t\text{-}1\}$, where variable T_{pr} denotes the preview time (similarly, T_{bf} is the time to come out of a bandwidth fault, T_{cg} is the time to change colour grading after a in-time director feedback, T_{cc} is the colour correction time). Transitions depicted by the *<<probabilistic choice>>* facilitate a probabilistic choice behaviour by probabilistic assignment occurring during the transition, e.g. assign the preview iterations number to be 1, 2 or 3 with probability of 1/3 for each assignment.

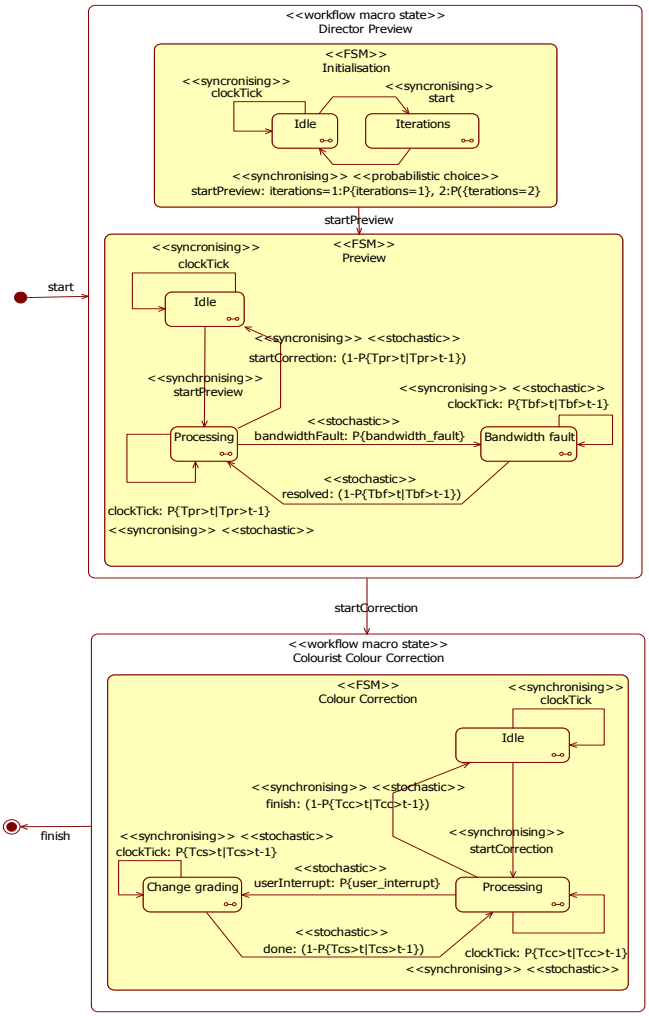

Figure 4 State chart of the FSMs for the preview and colour correction post-production workflow activities.

To perform the model analysis we used the PRISM model checking tool [5], [6]. The UML diagrams are directly mapped to the PRISM modelling language. The PRISM modelling language is a state based language based on the Reactive Models formalism [17]. A model described in PRISM is compiled into a Markov chain (one of the following: CTMC, DTMC or MDP) and analysed using PRISM temporal logic algebra based on PCTL [18], CSL [19] and LTL [20]. We analyse the completion time probabilities of the different workflow activities and the workflow as a whole. We performed a number of experiments to demonstrate the use of the proposed modelling technique. See Table 1 for the experiments setup. The experiments are performed in time steps of 10 minutes in order to constrain the model state space and the overall execution time. This coarse quantisation is sufficient for understanding the main features of the processes

and the experiments can be repeated with finer granularity if more accuracy is required. For the experiments described here it take about 5 minutes to complete on a Dell Latitude with Intel Core2Duo 2.4GHz T8300 4GB DDR2-667 SDRAM.

Experiment	Footage Length [min.]	Preview Iterations Probability	Streaming Fault Rate [per hour]	Delay after Streaming Fault [min.]	Correction Interrupt Rate [per	Delay after Correction Interrupt [min.]
1	30 to 50	P(1)=1/3, P(2)=1/3, P(3)=1/3	2	10	2	10
2	30 to 50	P(1)=0.6, P(2)=0.3, P(3)=0.1	2	10	3	10
3	30 to 50	P(1)=0.6, P(2)=0.3, P(3)=0.1	1	10	3	10

Table 1 Experiments setup.

For the first experiment we executed the model set as per the scenario described at the beginning of this section (see experiment number 1 in Table 1). We obtained the completion time probability density functions (PDFs) of the preview activity, colour correction activity, and the workflow (the sequence of the two activities), depicted in Figure 1, and the completion time cumulative distribution function (CDF) of the workflow, depicted in Figure 6 (the graph marked with circles).

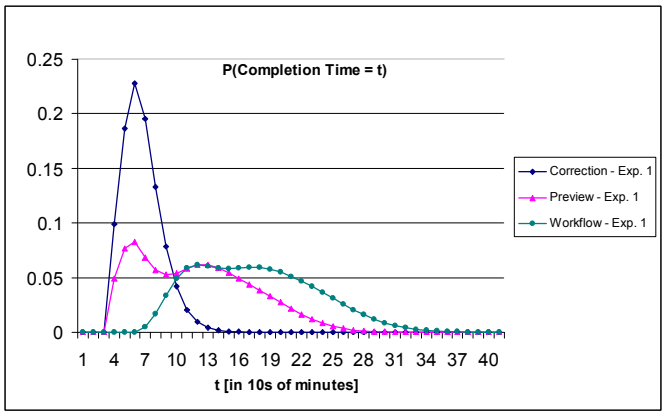

Figure 5 Workflow and individual activities completion times PDFs for experiment 1.

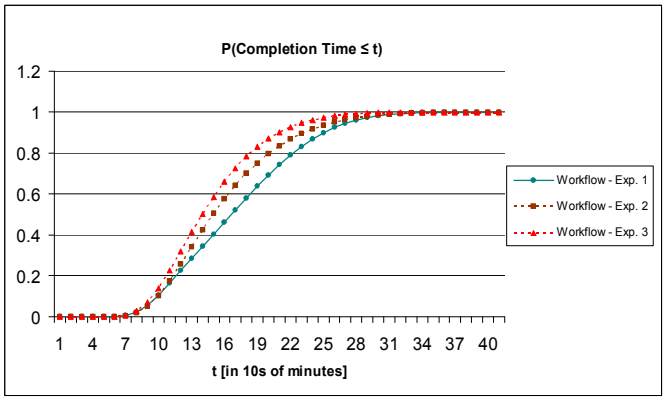

Figure 6 Workflow completion times CDF for the different experiments.

From the workflow CDF, for example, it can be seen that the probability of completing the workflow within 3 hours is 60%, i.e. the probability of the film director being involved with the colour correction activity for too long (arbitrary accepted to be 3 hours) is too high, i.e. 40%. To moderate this issue we could attempt to change the overall workflow strategy. Looking at the PDFs of the individual activities we note that the longest and most uncertain activity is the footage preview. We can propose that the film director should spend less time during this activity and proceed to the colour correction activity. This would mean that the probabilities of the footage preview iterations will change, and let assume that the new estimated iteration probabilities are 0.6, 0.3 and 0.1 respectively for 1, 2 and 3 iterations. This will most likely have an effect on the colour correction activity in a way that more interrupts for colour re-grading will be needed. Let assume that now there will be 3 interrupts per hour instead of 2. We performed a second experiment with the so amended scenario (Table 1, experiment 2). The workflow completion time CDF is depicted in Figure 6 (the graph marked with squares). Now the probability of completing the workflow within 3 hours is about 70%. Further, we know that we can improve the workflow completion time if we opt for paying extra for a better IaaS network resource for the footage preview activity. Let assume we can hire a link with more stable bandwidth so that the streaming failures are down to 1 per hour instead of 2. The workflow completion time CDF of the so amended scenario (Table 1, experiment 3) is depicted in Figure 6 (the graph marked with triangles). Now the probability of completing the workflow within 3 hours is about 80%.

The above 'what-if' experiments demonstrate how one can use the proposed modelling techniques to optimise practices employed in business workflows. Additionally, if the costs of the IaaS resources and the cost of the human resources are known, one can perform workflow const optimisation experiments to derive an optimal workflow cost. In the IRMOS project we intend to use this modelling technique to derive the optimal IaaS resource allocation for given workflow completion time constrains and workflow cost constraints, i.e. the technique will be facilitating optimal infrastructure QoS derivation when negotiating a SLA between a SaaS provider and an IaaS provider.

The results presented so far can be put to use in yet another way. Suppose a SaaS provider agrees a hard deadline for the completion of an activity, then the probability of meeting this deadline can be calculated. If there is scope to negotiate with the IaaS provider, then the trade-off between resources cost, completion time and possible penalties can be analysed. For example, Figure 7 shows how a SaaS provider might use a cost model when determining the optimum provisioning strategy for clients. Given the various levels of uncertainty that exist as discussed in previous sections, the most common way to be sure of meeting the obligations is through simple over-provisioning of resources, e.g. booking resources that it will have to pay for but might never get used. This can be expensive. On the other hand, if the SaaS Provider reduces its cost by reducing the resources reserved/used from IaaS providers then it increases the risk of not meeting obligations to its clients and hence incurring penalties. Somewhere in between is an optimum solution, which will depend on many

factors, some of which may not technical e.g. customer relationship management.

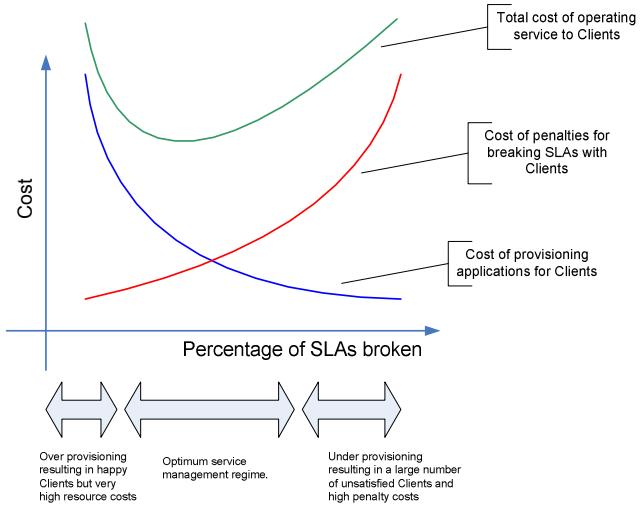

Figure 7 Trade-off between cost and risk used to identify optimum strategies.

Predictive models as presented earlier in this section clearly have an important role to play when the SaaS provider assesses the probability that a commitment to a client will be met against the cost of provisioning the application for that client and for any penalties if the resource is under provisioned.

5 CONCLUSIONS

This paper has discusses how models can be constructed to analyse interactive real-time applications on service oriented infrastructures. The techniques applicable are varied and include the use of stochastic process algebras and finite state automata.

We have emphasised the need to give particular attention to modelling uncertainty surrounding real-time applications. This includes modelling variation in the inputs to an application (e.g. data) and in performance of resources (e.g. bandwidth), and also modelling the user behaviour, all of which affect the probability of the application executing successfully, i.e. according to given constraints. The techniques used have been demonstrated using a specific application scenario, which provides valuable insight into the level of detail needed when developing meaningful models.

The key features of the modelling approach we have taken are:

1) Separate the application level parameters from the infrastructure level parameters and use mapping functions or statistical estimators to quantify the application performance (which in itself can be very challenging!).
2) Explicitly model uncertainty/variability at all levels using probability distributions.
3) Use stochastic modelling techniques as the basis of experiments to explore the range of possible outcomes.
4) Use the results of these experiments to quantify the resources that will be required to execute the application, including the performance needed (QoS).
5) Use cost models to make quantitative evaluation and comparison of options in order to make decisions on where the trade-offs can be made most effectively.

Acknowledgements

IRMOS is a collaborative research and development project supported by the European Commission under the Seventh Framework Programme FP7/2007-2011, ICT-2007.1.2. More information can be found on the IRMOS website www.irmosproject.eu

References

[1] www.irmosproject.eu
[2] http://www.microsoft.com/azure/default.mspx
[3] http://code.google.com/appengine/
[4] Web Services Business Process Execution Language Version 2.0, http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html
[5] PRISM web site http://www.prismmodelchecker.org
[6] Kwiatkowska, M., Norman, G., Parker, D., and Sproston, J. Performance analysis of probabilistic timed automata using digital clocks. Form. Methods Syst. Des. vol 29, no 1, pp 33-78, 2006, DOI= http://dx.doi.org/10.1007/s10703-006-0005-2
[7] R. Bruni, H.C. Melgratti, and U. Montanari. Theoretical foundations for compensations in flow composition languages. In POPL, pp. 209–220. ACM, 2005.
[8] L. Bocchi, C. Laneve, and G. Zavattaro. A calculus for long-running transactions. In FMOODS, LNCS 2884, pp. 124–138, 2003.
[9] C. Guidi, R. Lucchi, R. Gorrieri, N. Busi, and G. Zavattaro. SOCK: a calculus for service oriented computing. In ICSOC, LNCS 4294, pp. 327–338, 2006.
[10] Formalizing languages for Service Oriented Computing, Claudio Guidi, Technical Report, Department of Computer Science, University of Bologna, UBLCS-2007-07, March 2007.
[11] A Formal Model of Services, M. Broy, I. Krugger, M. Meisinger, ACMTransactions Software Engineering and Methodology,Vol. 16,No. 1, Article 5, Publication date:February 2007
[12] Li, J., He, J., Zhu, H., and Pu, G. 2007. Modeling and Verifying Web Services Choreography Using Process Algebra. In Proceedings of the 31st IEEE Software Engineering Workshop (March 06 - 08, 2007). SEW. IEEE Computer Society, Washington, DC, 256-268. DOI= http://dx.doi.org/10.1109/SEW.2007.105
[13] Lapadula, A., Pugliese, R., Tiezzi, F.: Calculus for Orchestration of Web Services. In: Proc. ESOP 2007. LNCS, vol. 4421, pp. 33–47. Springer, Heidelberg (2007) (full version available at), http://rap.dsi.unifi.it/cows
[14] Stochastic COWS, Davide Prandi, Paola Quaglia, Proc. 5th International Conference on Service Oriented Computing, ICSOC~'07. LNCS vol. 4749. 2007.
[15] A. Bianco and L. de Alfaro, Model checking of probabilistic and nondeterministic systems, in P. Thiagarajan, editor, Proc. Foundations of Software Technology and Theoretical Computer Science, LNCS 1026, pp 499–513, Springer Berlin/Heidelberg, 1995.
[16] C. Baier and M. Kwiatkowska, Model checking for a probabilistic branching time logic with fairness, Distributed Computing, 11:125–155, 1998
[17] R. Alur and T. Henzinger. Reactive modules. *Formal Methods in System Design*, 15(1):7-48, 1999.
[18] H. Hansson and B. Jonsson. A logic for reasoning about time and reliability. *Formal Aspects of Computing*, 6(5):512-535, 1994.
[19] C. Baier, J.-P. Katoen, and H. Hermanns. Approximate symbolic model checking of continuous-time Markov chains. In J. Baeten and S. Mauw, editors, *Proc. 10th International Conference on Concurrency Theory (CONCUR'99)*, volume 1664 of *LNCS*, pages 146-161. Springer, 1999.
[20] C. Baier. On algorithmic verification methods for probabilistic systems, 1998. Habilitation thesis, Fakultät für Mathematik & Informatik, Universität Mannheim.