Interactive Realtime Multimedia Applications on Service Oriented Infrastructures

ICT FP7-214777

WP 5 – Framework Services

D5.1.1 Models of Real-time Applications on Service Oriented Infrastructures

IRMOS_WP5_D5_1_1_IT_Innovation_v1_0

Scheduled Delivery: 31 January 2009
Actual Delivery: 9 February 2009
Version: 1.0

Project co-funded by the European Commission within the 7th Framework Programme		
Dissemination Level		
PU	Public	X
PP	Restricted to other programme participants (including the Commission)	
RE	Restricted to a group specified by the consortium (including the Commission)	
CO	Confidential, only for members of the consortium (including the Commission)	

Responsible Partner: IT Innovation Centre (IT Inn)

Revision History:

Date	Editor	Status	Version	Changes
9 February 2009	Matthew Addis	Final	1.0	Ready for submission

Authors

Bill Mitchell , Zlatko Zlatev , Matthew Addis (IT Innovation); Tor Neple (Sintef); Kleopatra Konstanteli, George Kousiouris, Spyridon Gogouvitis, Dimosthenis Kyriazis, Gregory Katsaros (NTUA); Ralf Einhorn (DTO); Thomas Voith (ALUD); Eduardo Oliveros (TID)

Internal Reviewers

Reviewer 1: Giorgio Buttazzo (SSSA); Reviewer 2: Marcus Kessler (ALUD); Consolidating Reviewer: Uwe Wössner (USTUTT)

Copyright

Acknowledgements

The research leading to these results has received funding from the EC Seventh Framework Programme FP7/2007-2011 under grant agreement n° 214777

More information

The most recent version of this document and all other public deliverables of IRMOS can be found at http://www.irmosproject.eu

Glossary of Acronyms

Acronym	Definition
AC	Application Component
ACC	Application Client Component
ASC	Application Service Component
ASCD	Application Service Component Description
BPEL	Business Process Execution Language
BPMN	Business Process Modelling Notation
CDF	Cumulative Distribution Function
CTMC	Continuous Time Markov Chain
D	Deliverable
DAS	Direct Attached Storage
EC	European Commission
HRTS	Hard Real-time System
IRMOS	Interactive Realtime Multimedia Applications on Service Oriented Infrastructures
ISONI	Intelligent Service Oriented Network Infrastructure
LRT	Long Running Transaction
MAP	Modelling Analysis and Planning
MARTE	OMG UML profile for Modelling and Analysis of Real-time and Embedded Systems
MDP	Markov Decision Process
MFLOPs	Mega Floating Point Operations
MFLOPS	Mega Floating Point Operations per Second
MIPS	Millions Instructions Per Second
NAS	Network Attached Storage
NFL	Non Functional Property
OMG	Object Management Group
OSD	Object Storage Device
PCA	Principal Component Analysis
PDF	Probability Density Function
QoS	Quality of Service
SAN	Storage Area Network
SC	Service Component
SLA	Service Level Agreement
SOA	Service Oriented Architecture
SRTS	Soft Real-time System
UML	Unified Modelling Language
VSL	Value Specification Language
VSN	Virtual Service Network
VMU	Virtual Machine Unit
WAN	Wide Area Network
WP	Work Package
WCET	Worst Case Execution Time
WSRF	Web Service Resource Framework
YAWL	Yet Another Workflow Language

Table of Contents

List of Figures

List of Tables

1. Executive Summary

The IRMOS project is developing tools and techniques for modelling, simulating, analysing, and planning 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.

The report considers the value-chain for real-time applications hosted by third-party service providers. In the context of this value chain, we 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 presented for building models of real-time applications including the use of stochastic process algebras, finite state automata, workflow models (e.g. BPEL, BPMN and YAWL) and specification languages (e.g. UML MARTE). The report discusses how these models need to be supported by estimation of application resource consumption, e.g. through benchmarking and fitting, and how mapping techniques allow models to be built for different actors in the value chain, e.g. for application providers and infrastructure providers. Tool support is also discussed e.g. PRISM for probabilistic model checking and Visual Service Composition Studio for service-oriented modelling.

A detailed and specific real-time application scenario is included and modelled to allow the various techniques presented in this document to be demonstrated and quantitatively evaluated. The scenario also reveals the level of detail needed in order for meaningful modelling to be achieved in practice. The modelling scenario has been carefully engineered to be as representative as possible of the broad range of application characteristics encountered in the three reference IRMOS applications (film post-production, eLearning, virtual and augmented reality).

The modelling techniques discussed and then demonstrated in this report include identifying what resources are necessary to support an application, when those resources will be required during the application workflow, what performance is needed from them (i.e. QoS) and what will happen to the application if the required performance is not delivered. All of these are essential when developing and then agreeing service level agreements between the various entities in a service oriented infrastructure.

2. Introduction

This report is a product of IRMOS WP5 Task 5.1. The purpose of this task is to develop tools that allow the execution requirements of real-time applications to be externalised in the form of models and specifications. The purpose of D5.1.1 is to provide models and techniques to allow these execution requirements to be estimated for real-time applications. This addresses objective O1 of WP5, namely to "produce models and descriptions of the requirements of real-time applications when deployed and executed on SOIs".

Execution requirements include resource requirements (e.g. memory, storage, processing, networking), coordination/orchestration requirements (e.g. workflow, messaging, synchronisation, timing), and business requirements (e.g. how contractual relationships are captured and embodied in service level agreements).

The models and execution requirements covered by this deliverable cover the actors involved in the application value chain, the corresponding business relationships between them, and the time varying aspects including application workflows and temporal profiles, including: (a) data and control aspects of the application workflow, including the synchronisation of users and resources during the interactive workflow activities; (b) temporal profiles of how requirements are likely to change as a function of, for example capturing when users are anticipated to join or leave session with an online collaborative environment; and (c) the specification of aspects of the corresponding business level SLAs, for example temporal deadlines and QoS parameters including performance and any allowed variations

Models of real-time applications have use in various stages of the lifecycle of designing, developing and using real-time applications. Figure 1 shows where models are used with respect to the IRMOS application phase diagram.

Workflow modelling is required in order for an application developer to define the workflow of an application. A workflow includes what activities need to be done, what causal dependencies exist between the activities, and any constraints there are on QoS. The workflow is necessary in order to create an application from elementary service components. Examples of workflow modelling during the application design phase are described in Section 5.

New OMG standards, such as MARTE and SoaML, mean that now UML allows real-time aspects of streaming multimedia applications to be incorporated within a MDE design methodology. This style of UML is particularly relevant during the application service design phase. UML models can then be used to define application service capabilities and interfaces between architecture components. Examples of these models are described in Section 6.

Models in IRMOS are also used so service providers can may define the application requirements for the IRMOS execution platform and the specific resources required by the application during runtime. Here, models help to determine the final amount of resource that will need to be reserved during the application concretion phase and the following negotiation and reservation steps. These reservations will be determined with respect to the cost model and the probability of success that the service provider decides upon according to their business plan. Examples of these models are presented in Section 7.

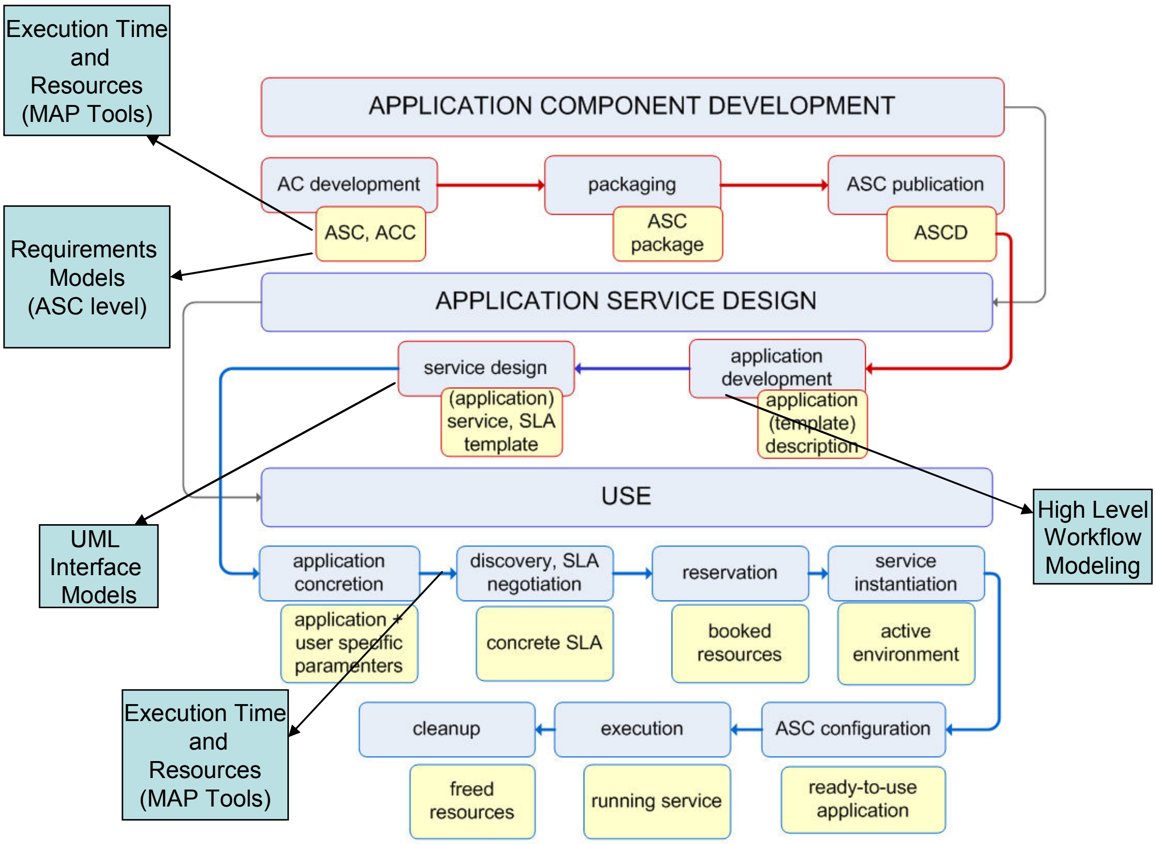


Figure 1: where models are used in IRMOS

Finally, it is also important to note that another task in IRMOS WP5 (T5.2) covers how to specify networks of services (i.e. VSN descriptions when reserving resources in ISONI). Therefore, D5.1.1 does not go as far as how to specify execution requirements in terms of a VSN specification. However, D5.1.1 does discuss how models/specifications can be created to describe execution requirements for a real-time application. This separation should mean that the results of this report are of use beyond the project when others seek to design and develop their own applications and infrastructures for real-time applications.

2.1. Deliverable Structure

The main body of the deliverable consists of two main parts.

Section 3 provides a discussion on why modelling is important, what problems modelling can solve, who would use the models and when they would be used in the lifecycle of designing, developing and using real-time applications. These aspects of modelling are presented as a set of modelling facets which capture the 'who', 'why', 'when', 'where' and 'what' of real-time application models.

Section 4 to Section 7 provide a tangible and practical example of modelling in the form of a case study. This includes a series of implemented models for the various aspects of the case study, e.g. model of the high-level workflow and model of resource needs for executing real-time applications in the workflow. Each case study section explicitly describes which modelling facets are being addressed.

Section 8 then provides a conclusion summarising the main findings of these two parts of the report.

The main body of the report is followed by Annex A to Annex C which present the techniques used to create the models in the case study, along with a review of the state of the art in modelling techniques.

Finally, Annex D describes the Quality Assurance metrics used as part of the IRMOS Internal Review process applied to this report before submission to the Commission.

3. The need for models of real-time applications

The IRMOS project will develop tools and techniques for modelling, simulating, analysing, and planning IRMOS real-time applications. These tools and techniques are intended to support some of the processes involved in designing, developing, deploying and executing applications on service oriented infrastructures which support guaranteed QoS for resources. In this context, this section addresses the following questions:

- Why are these models needed?
- Who might use these models?
- What will they do with them?
- How will they benefit?
- When are the models used during the application lifecycle?

The section concludes by presenting a hypothetical application scenario. This scenario is representative of the broad range of application characteristics encountered in the three IRMOS application scenarios (film post-production, eLearning, virtual and augmented reality). The scenario is important as it is used in this report to allow the various simulation and modelling techniques presented in this document to be demonstrated and quantitatively evaluated.

3.1. Overview of IRMOS real-time applications

The IRMOS project has three exemplar real-time application scenarios. A description of these scenarios can be found in IRMOS Deliverable D2.1.1 and a detailed definition of the use cases and resource parameters for these scenarios can be found in IRMOS Deliverable D4.1.1. A short overview of the scenarios is included below for convenience and to provide a lead in to the more detailed characterisation of these scenarios later in the document.

Film post production

The film post production scenario centres on collaborative working between a group of highly talented artists, colourists, editors, VFX and sound operators, located in different countries when working together during the hot phase of post-production of an international Sci-Fi movie co-production.

In the scenario, early in the morning (CET) they log into the IRMOS platform to jointly review the shooting result of yesterday (which partly has been ingested, edited, colour corrected and sound synched during the night) and to prepare today's work together with the director, the director of photography (DOP) and the producer.

The logical architecture of the scenario is shown in Figure 2.

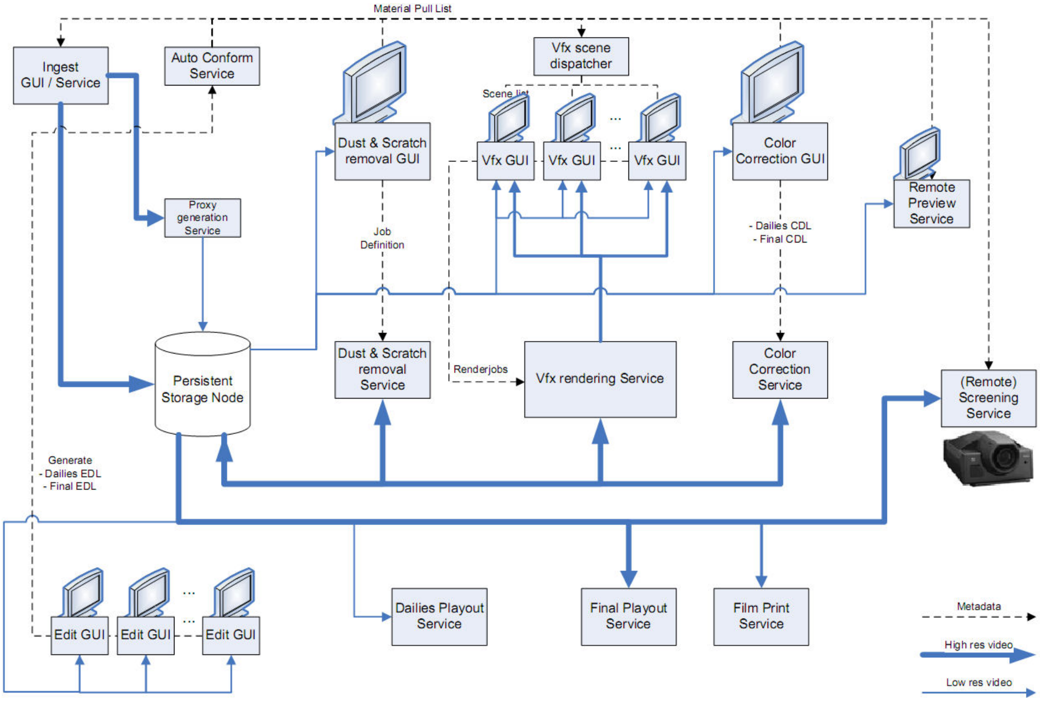


Figure 2: Film Postproduction Use Case

The digital dailies workflow involves ten major activities that start with the already illuminated negative film from a mechanical film camera or the image files generated by a digital film camera all the way through to the final deliverables of the process, for example a film master ready to be copied for Cinema release or the master files to be used for DVD duplication. This workflow is shown in Figure 3.

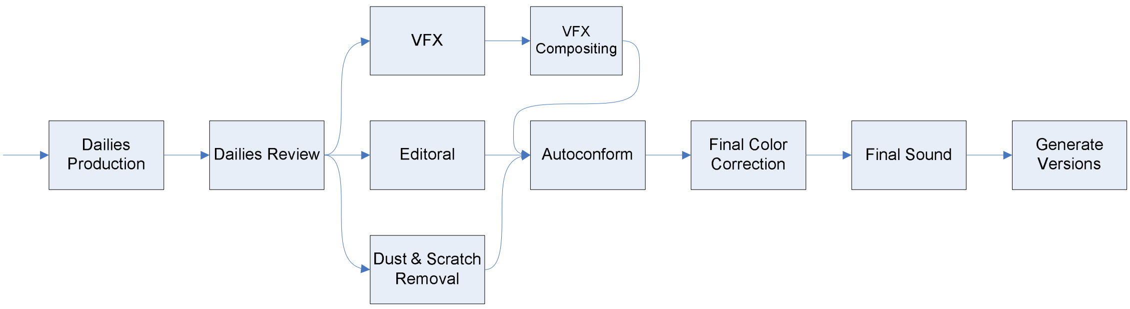


Figure 3: Film-Postproduction Workflow (© Thomson Systems Germany)

Virtual and Augmented Reality

This scenario concerns modern product design processes in engineering that involve virtual and augmented reality for optimisation of pre-product prototypes. For such tasks software suites like COVISE are utilised which allow to create a workflow based on the designer's or engineer's needs incorporating previously executed simulation calculations on dedicated hardware. COVISE offers different 3D visualisation environments from desktop to fully immersive 3D visualisation environments. Combination of data input modules, data post-processing, data colouring, iso-surface and particle tracer modules provide the best fitted visualisation for the individual scenario. Augmented reality comes from combining the results of real physical experiments with that of software-based simulation. Individuals using COVISE can work on their own or in collaborative working sessions involving teams of people distributed across the world. The use of IRMOS to support COVISE allows collaborative working to include augmented reality, for example experiments done on a real-world prototype can be done with simulation results overlayed at the same time. An example would be visualising the results of a computational fluid dynamics simulation of car aerodynamic properties overlaid on top of real-world video from smoke trails over a prototype car in a wind tunnel as shown in Figure 4. In the collaborative scenario, all partners can influence the visualisation either through adjusting parameters directly or by communicating changes to a partner who has the opportunity to apply such changes.

The wind example is elaborated below. An engineering company in Munich wants to evaluate the performance considering aerodynamic drag of a car model in a wind tunnel. A university institute at the University of Stuttgart provides the wind tunnel facility. The car model has been deployed into that wind tunnel and the experiment is started with a smoke probe for visualising the air flow as smoke is a rather mass-less particle. Therefore behaviour compared to real weather-based conditions like rain-drops is different. A computer simulation is run in parallel to simulate the same setup with mass-based particles, e.g. like rain-drops to simulate the effect on contamination of the rear mirror at the driver's door. With a camera and a marker the physical experiment is recorded and streamed in the visualisation software where the video image is overlaid with the simulation results.

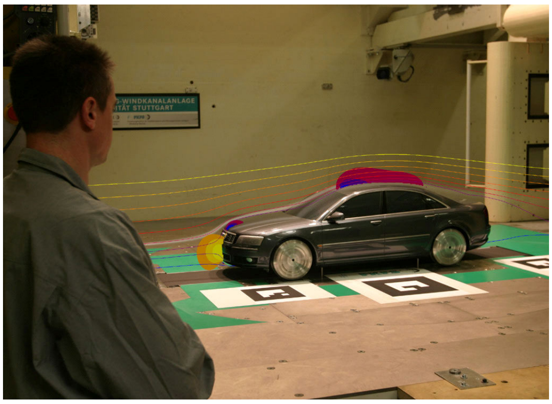


Figure 4: Augmented Reality for car aerodynamics analysis

eLearning

The IRMOS eLearning scenario centres on visitors to museums that can receive information on their portable electronic devices (e.g. mobile phone or PDA) about the museum artefacts around them or interact with each other and a teacher through a 3D virtual world. So, for example, in the 'indoor' part of the scenario, a museum visitor might be walking around the museum rooms with his mobile device WIFI enabled. The museum is equipped with WIFI position-detecting system. When the visitor goes nearby a work of art, he obtains learning contents shown on his mobile device. In the 'virtual world' part of the scenario, a teacher might be walking around the museum rooms with his mobile device. The museum is equipped with an indoor positioning system. Some students are at home (or other remote location) browsing an internet page where the museum is reconstructed as a Virtual World. The teacher movement is reconstructed in real-time into the virtual world and the students can see his avatar moving. The teacher cannot see the virtual world. The Teacher and students can interact for asking questions, receiving answers, discussing. The interaction may be both written and vocal. When the teacher goes nearby a work of art, he can invite the students (via chat or voice) to start a lesson. Each student plays his lesson by himself, but can continue communicating via chat. The lesson is overlapped on the virtual world.

The IRMOS eLearning scenario forms part of a set of possible wider applications of IRMOS to social networking and interactive communication involving media exchange as shown in Figure 5.

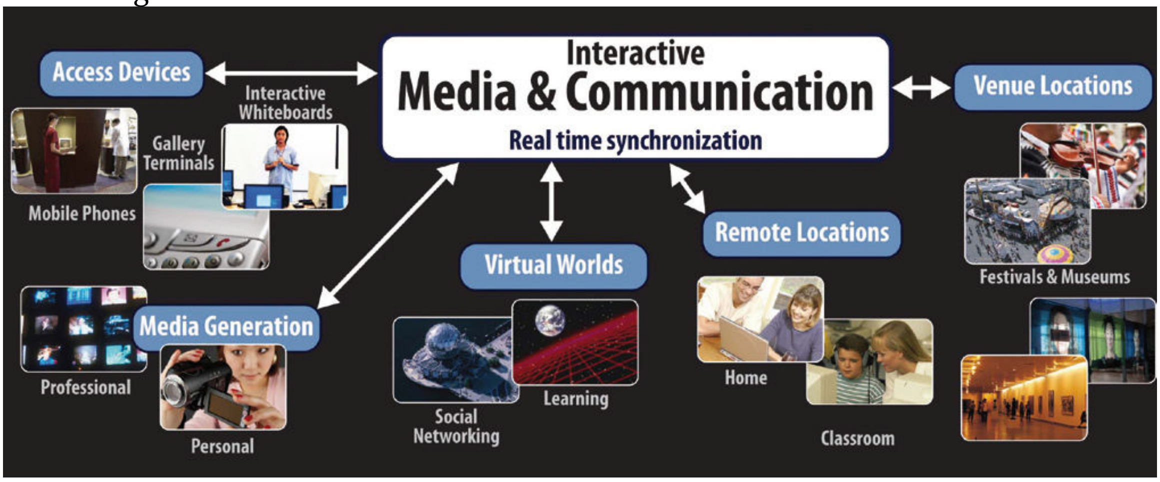


Figure 5: IRMOS for social networking and learning

3.2. Overview of the IRMOS value chain

It is important to consider the use of real-time application models in different parts of the value chain. Full details can be found in IRMOS Deliverable ID2.1.2 -*Analysis of business Models*. The generic IRMOS value chain in Figure 6 is reproduced from ID2.1.2 below.

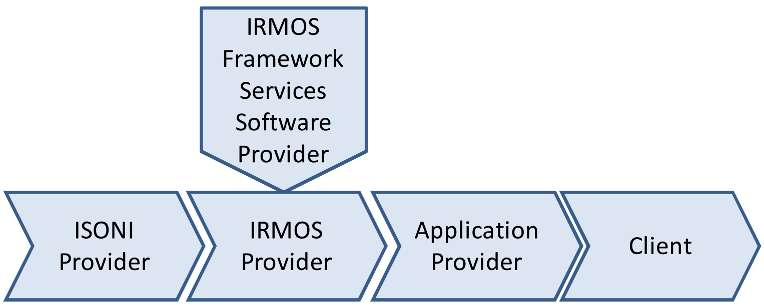


Figure 6: Initial Value Chain for IRMOS

The actors in the value chain can be grouped into three different categories:

- Providers of the basic infrastructure and resources that will be used by application providers and their clients. The infrastructure provides the basic capacities (storage, networking, processing) needed by application providers. The application providers can then in turn offer applications with guaranteed QoS that meet the real-time requirements of their clients.

- The Application Providers create and provide the applications that will be used by the clients. This is done in collaboration with application software developers and vendors along with any content providers, depending on the specific application.
- The client is the entity that uses the application. Internally this entity can be considered to consist of a Customer, who pays for the service and has the ability of sign the SLA with the Application Provider, and a Consumer, who is the person that actually uses the application.

An analogous value chain that already exists is Amazon Web Service [1], that provides storage (S3) and processing (EC2) resources that are used by a range of application providers (e.g. Mailtrust, Alexa, SmugMug etc. [2]) to host and deliver their applications as web-accessible services to members of the public or businesses.

From a real-time application modelling perspective, the important aspects of this value chain is the separation of the resource-level (storage, processing, networking) from the application-level (e.g. a software tool for video effects processing provided as a service) and how these two different levels are provided by different actors in the value chain. This results in two levels of SLA as shown in Figure 7 which in turn means two levels of QoS specification and two levels of monitoring/reporting.

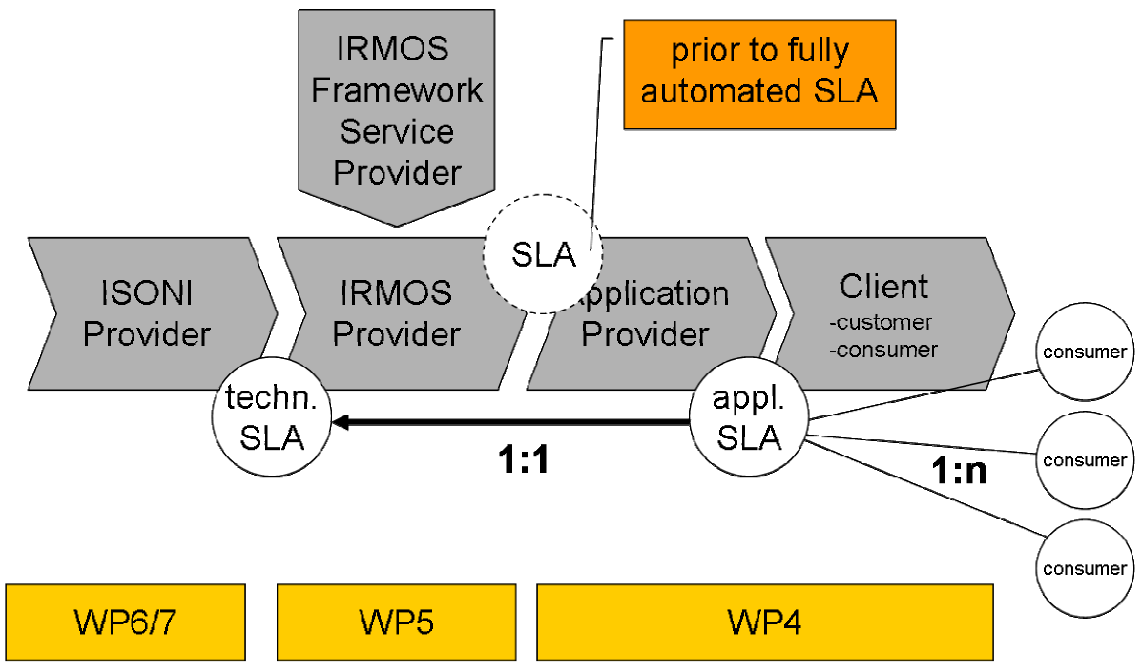


Figure 7: Simplified value chain and use of SLAs in IRMOS

- The Application SLA is an agreement established between the Client and the Application Provider. This SLA contains the high level QoS parameters of the application as defined by the user.

- The Technical SLA is an agreement between the ISONI Provider and the IRMOS Provider. This agreement contains low level QoS parameters associated with the infrastructure.

There is another SLA established between Application Provider and the IRMOS Provider, but this is a long-term relationship that is previous to the automated SLA negotiation process and that does not have to be negotiated every time a new customer uses an application over the IRMOS platform. Therefore, during the SLA negotiation phase this SLA is considered as fixed.

The Application Provider and IRMOS Provider need between them to map the application-level QoS terms in the application SLA to/from the infrastructure-level QoS terms in the technical SLA.

The models described in this report are primarily targeted for use by the Client, Application Provider and IRMOS Provider, i.e. they provide a description of the execution requirements of the application in a form that can be used in a specification of a technical SLA. The models do not go into the detail of how the ISONI Provider provisions a particular set of resources for a given application – this is work done elsewhere in IRMOS within WP6 - *Execution Environment* and WP7- *Intelligent Networking*.

In the rest of this document we use the term 'Infrastructure Provider' to refer to any actor that provides low-level resources as a service (storage, bandwidth, processing) with some form of QoS guarantees allowing the resource to be used for real-time applications. ISONI is one example of an Infrastructure Provider, although we do not restrict this report to ISONI as we need to ensure the models and techniques proposed are general purpose and not overly constrained by any specific features of ISONI.

3.3. Why are models needed?

With reference to the IRMOS value chain, at one end there is a client's desire to use real-time applications in a high-level workflow and at the other end there is a corresponding set of services deployed at an Infrastructure Provider. There are many stages involved achieving the optimum configuration of services and resources in the context of the various actors in this value chain. This section looks at why modelling is useful in this process.

Modelling Facet	Title	Description
MFWHY1	The workflows and applications in IRMOS typically involve multiple people in multiple locations interacting with each other and various hardware/software applications in different ways	Understanding and planning what applications are needed, when they are needed and who will use them can be a complicated activity involving various uncertainties, e.g. how long a task will take to complete or how many people might be involved. In some cases, the workflow evolves at 'execution time' and it is not possible to define the steps or paths through the workflow until at least some of the steps have been executed.
MFWHY2	Resource needs for an application typically depend strongly on exactly how that application is used	The amount of resources needed by an application is typically a complicated function of the data and control information that is input into the application both before and during execution. For example a user might adjust application settings during application execution (e.g. changing the parameters of a video effect or deciding on the number of frames to be rendered). These adjustments can in some cases dramatically change the application's resource needs.[1]
MFWHY3	Mapping from application needs to resource requirements is generally a difficult task.	Mapping (e.g. converting the need to sustain a particular video frame rate and video quality into network performance in terms of bandwidth, jitter, packet loss) is not a simple analytical process that yields precise results. Mapping will at best produce a rough estimate of resources needed. This is typically expressed as a likelihood of resource consumption being between upper and lower bounds rather than a single number. For complex applications, e.g. video effects rendering or engineering simulations, if the actual resource consumed is within say 50% of the resource predicted, then the prediction can be considered a good one!

[1] The conventional approach is to plan for the worst case or not allow any changes to be made at 'run time'. Whilst this guarantees the application will complete given the available resources, this approach can be inefficient/expensive e.g., due to over-provisioning or inflexible as no alterations can be made.

MFWHY4	Infrastructure Providers do not in general guarantee with 100% certainty the performance of resources when an application is executed[2].	Even if the resource requirements of an application can be determined precisely in advance, Infrastructure Providers will only guarantee that the performance of resources delivered for the application will statistically be between certain limits, e.g. an upper and lower bound, i.e. guarantees are expressed in terms of a probability not a certainty[3] Whilst probabilities can be quite high for individual resources used for specific steps in a workflow, when multiplied together across a set of resources and workflow activities the compound probability of having high enough performance can be dramatically less and can result in application failure[4].
MFWHY5	Trade-offs between time, effort, money and resources are often needed in the real world.	There are often trade-offs to make in order to achieve the most cost effective solution. A balancing act[5] is often needed between factors, e.g. how many concurrent users to support, what video quality to use, the likelihood of disrupted or degraded service, and of course the cost. The relative cost[6] of different types of resource (storage, computation and networking) also plays a part in optimising an application, e.g. by trading extra computation cost of data transcoding for reduced network cost of transfer. Many of these factors influence each other and understanding the options and choosing the best one can be difficult
MFWHY6	Interactions between the various types of uncertainty (workflow, mapping, QoS etc.) are not necessarily linear.	Systematic propagation, combinatorial effects, critical points of failure and other effects are likely to be important and require careful analysis when predicting application requirements. For example, you can't just add up a set of probabilities to produce an overall estimate of whether a workflow or application is likely to perform as expected.

Table 1 Modelling facets that describe why models are needed.

[2]It is important to distinguish between performance and availability. Availability of a resource can be very high, e.g. 5 nines (probability of being able to access a network is 99.999%), but performance can still be very variable (e.g. bandwidth on the network is anywhere between 100kBit/s and 100MBit/s). Online resources often have very high availability, but it is the guarantees of performance that matter for real-time applications.

[3] The probability of the resource being between limits will of course depend on how stringent these limits are, e.g. it will typically be much easier for a resource provider to deliver against very wide bounds than very narrow ones. The probability of staying with bounds will depend on the separation of the bounds and the overall performance needed, and could be quite high if the customer is prepared to pay enough (e.g. 98% of the time jitter will be less than 1ms) but this is a probability nonetheless.

[4] There is also the associated issue of being able to differentiate between a failure caused by lack of performance at the resource level and other failures that may occur at the application level, e.g. due to a bug or incorrect user input.

[5] Ultimately this balancing act is a choice for the client or application provider to make, e.g. whether to use lower quality video in order to support more concurrent users over a network connection that has limited bandwidth. The Client or Application Provider will need suitable tools to support this decision making process and the development of these tools is one of the objectives of IRMOS.

[6] Amazon S3 is a good example. The cost of transferring data in and out of S3 just once is the same as the cost of storing it on S3 for almost two months, i.e. network costs are relatively high compared to storage costs. http://aws.amazon.com/**s3**

In summary, the issues are:

(a) complexity of the applications that IRMOS aims to support

(b) uncertainties in workflows, application needs and resource requirements

(c) making trade-offs in order to find the most cost effective solution

If resources are underestimated then it is likely that the application or workflow will fail or degrade unacceptably. If resources are over-estimated, then the consequent charging by Infrastructure Providers for the reserved resources, rather than the used resources, could cause costs to be uneconomic[7].

The challenge is to identify the range of resource requirements that bound application execution and the likelihood that the actual resource usage will be within this range. The decision is then one of cost of provisioning for this range (and the cost incurred for deviations from the range) against the risk (probability and impact) that execution deviates.

3.4. What needs to be modelled?

This section elaborates on the different characteristics and parameters that need to be modelled

[7] This is an issue of business model and other ways in which the resources could be used. For example, 'spare' capacity could be used as and when available to service 'best efforts' jobs. This would ensure a high overall level of resource usage and could allow, for example, an Infrastructure Provider to 'refund' customers if they do not use all of their reservation. This model is common in other sectors, e.g. the airline industry.

Modelling Facet	Characteristic	Description	Notes
MFWHAT1	Workflow	The series of activities involved in an application and their causal relationship, e.g. review, visual effects, colour correction and transcoding as part of a post-production workflow.	The workflow we describe here is at the application level. There may be finer-grained workflows that take place at the service component level, but these are the scope of future deliverables.
MFWHAT2	Temporal constraints	The absolute or relative timing constraints for the activities in a workflow, e.g. a particular activity has to start at t0 and end at t1.	Some temporal constraints will be hard, e.g. 'this task must complete before 10:00am or the workflow is considered failed' and some will be soft, e.g. 'this task should ideally complete within 3 hours, but variation of up to ½ hour either way is possible'.
MFWHAT3	Users and their interactions.	The number of users, how many interact with the application concurrently, what level of interactivity is required, and what dependencies exist on them.	When users interact with applications, e.g. by participating in a virtual world, deciding to view a video stream or adjusting an application at run time, they generate a varying workload on the application. Unless a worst-case provisioning strategy is used, modelling is needed to optimise resource reservations. Furthermore, if users play an active role in the transitions and timing of a workflow, e.g. because it contains decision points or human tasks, then this needs to be modelled as part of the workflow and temporal constraints.
MFWHAT4	Data exchange	The frequency, volume and type of data exchanged between components of the application running on the infrastructure and between the application and the external environment.	Examples include upload of digital video, positioning data for mobile users interacting with location based services, synchronisation data for avatars in a virtual world, video and audio streams, or control commands sent by the operator of an augmented reality visualisation. Extremes are high frequency exchange of small messages (e.g. syncronisation data) through to low frequency exchange of very large datasets (e.g. digitised film).
MFWHAT5	Principal application parameters	The main application-level parameters that determine the resources needed to execute the application, e.g. number of frames to be colour corrected in post-production. These are the application parameters that need to be mapped to infrastructure parameters (see resource needs and performance below).	At the time of writing, there is relatively little information on the main application parameters for the three IRMOS reference scenarios, although many can be easily deduced, e.g. the volume of film in the post-production scenario or the number of concurrent users in the eLearning virtual world scenario.
MFWHAT6	Resource needs	The storage, processing, memory and networking resources needed to fulfil an application.	These parameters are enumerated in detail in IRMOS Deliverable D4.1.1

MFWHAT7	Performance	The bounds of performance for the resources needed, e.g. acceptable bounds for the jitter, latency and bandwidth of a network connection.	These parameters are enumerated in detail in IRMOS Deliverable D4.1.1
MFWHAT8	Sensitivity	How application performance or success varies with a change to the resources available to the application or to uncertainties in inputs.	Given the various uncertainties in the modelling process, mappings, input data to an application and the QoS of resources, it is essential to know how sensitive the application or models are to these uncertainties.
MFWHAT9	Cost	Measure(s) of cost that allow cost-benefit analysis to be done. For example, to allow the cost of provisioning resources for an application to be compared with the risk that the resources selected will not be sufficient.	Without a cost model, the logical conclusion of the modelling process is to use as much resource as physically available since this will always increase the probability of the application being successful. This is of course not a realistic option, so cost models are needed to allow real-world trade-offs to be explored.
MFWHAT9	Lifetime/longevity	The lifecycle for different parts of an application or the data used within it.	This applies particularly to persistent data and the services used to store and provide access to it. These services and data may need to live beyond the lifetime of the specific workflow activities or applications that process that data.

Table 2 Modelling facets that describe what models are needed.

The resource needs of IRMOS applications have received detailed analysis in Deliverable D4.1.1. In this deliverable, each of the three application scenarios were analysed in terms of their resource-level parameters, with a focus on performance parameters.

D4.1.1 groups the parameters into the following categories: volatile memory, volatile storage, persistent storage, processing and network. The storage, memory and network categories share a set of I/O parameters (of varying relative importance depending on the category):

I/O	
Bandwidth[8]	The data flow rate, e.g. number of Gigabytes that a network channel has the capacity to transfer each second.
Jitter	Unwanted variation in a periodic signal, e.g. variation in the inter-arrival time for packets in a real-time stream.
Latency	Delay between something being initiated and it starting to happen, e.g. the time between a request for data and the start of delivery of the data.
Packet-loss	Failure of one or more packets of data to reach their destination.
End-to-end delay	Delay between request and response including all steps in-between.
Concurrency	Number of concurrent connections that can be supported.

Table 3. I/O parameters

Memory and storage (persistent or volatile) have the additional parameters of data volume (i.e. how much data can be stored), random-access time (time taken to access an arbitrary bit of the data selected at random) and integrity (correctness of the data, which is something that can never be absolutely guaranteed) and longevity (how long the data needs to be kept for). Processing has the additional parameter of processing power (the rate at which processing can be performed, e.g. as measured in floating point operations performed per second). Network, storage and processing all have the additional non-functional parameter of availability (percentage of the time that the resource is available for use, typically expressed in terms of a 'number of nines', e.g. 5 nines meaning 99.999%).

3.5. Who needs to use the models?

This section describes who in the IRMOS value chain is expected to use models of real-time applications. Actors include those who design, develop, deploy, provide and use IRMOS applications. Different actors will use different models at different stages of the application lifecycle and for different reasons.

[8] Care is needed with terminology for network parameters, e.g. Bandwidth and Throughput are not the same thing (Bandwidth is a channel's capacity to transfer data at a particular rate whereas throughput is the number of messages correctly delivered per unit time over that channel).

Modelling Facet	Who might use the models?	What will they use models for?	What inputs need to be provided to the model?	Outputs created by the model
MFWHO1	Client and/or Application Providers	Estimate resource needs of an application	• Values/ranges of application parameters (e.g. for video streaming this might be frame rate, resolution, colour depth) • Mapping between application parameters and resource parameters (e.g. an empirical model, a function derived from knowledge of the code, statistics on previous executions).	Estimation of resource requirements. This includes: • What resources are needed (CPU, disk, I/O etc.) • Bounds on expected resource usage, e.g. upper and lower limits • Level of confidence that resource usage will be within these bounds.
MFWHO2	Client and/or Application Providers	Specify a resource utilisation profile for a given application or workflow.	• Workflow description (if appropriate) including steps, dependencies, participants, locations, applications, and constraints (e.g. deadlines). • For each application in the workflow, the inputs and outputs as described in ID1 above will also be needed	• Description of the resource needs of the application including what resource is needed, when it is needed, the QoS required, and any networking needed to access the application(s) deployed on this resource.
MFWHO3	Application Providers	Analyse workflows to understand their likely behaviour when using an Infrastructure Provider	• Workflow description including steps, dependencies, participants, locations, applications, and constraints (e.g. deadlines). • For each application, inputs and outputs as described in ID1 above are required • Infrastructure QoS guarantees including probabilities of meeting the agreed levels of performance	• Sensitivity to variations in resource QoS. • Sensitivity to variations in users, applications, or order of activities. • Identification of critical activities, interdependencies or resources that could lead to bottlenecks or failures. • Probability of meeting deadlines or other constraints • Design of workflows/services that have measurable and predictable behaviour

MFWHO4	Application Designer/ Developer	Plan how to develop and deploy an application as a set of application service components at an Infrastructure Provider	• Understanding of how the application works, e.g. through code analysis or benchmarking • Performance classes and cost models for Infrastructure resources, e.g. to understand the different options for deployment.	Better understanding of the ways in which the application could be split into various pieces (e.g. functions) to be deployed as ASCs in a way that achieves the best balance between use of resources, scalability, flexibility, robustness etc.
MFWHO5	Client and/or Application Providers	Make trade-offs and optimise an application/workflow against available resources	• Resourcing descriptions as described in ID2 above • Response from an Infrastructure Provider on what resources are actually available and the QoS that can be provided	Understanding of what application compromises to make that result in a best fit to available resources
MFWHO6	Application Provider	Design application SLAs	• Inputs as described in ID1, ID2, ID3	• Understanding of what SLAs are possible that can be met with confidence given available resources. This includes QoS specified at an application level. The SLAs might take the form of 'classes of service' at an application level much as an Infrastructure Provider might do at the resource level. • Estimate and optimise resource provisioning necessary to enact a workflow according to one or more SLA classes or template
MFWHO7	Application Provider	Optimise and manage application SLAs with multiple customers	• Inputs as described in ID1, ID2, ID3 for each customer	• Estimate and optimise resource provisioning needs necessary to enact applications/workflows according to SLA guarantees • Analysis of what would happen if the AP decides to break one or more SLAs for business reasons.

Table 4. Modelling facets for who needs to use models.

Modelling facets MFWHO1,2,3,5,6,7 target the Client or Application Provider and hence support the IRMOS value chain as presented in Section 3.2.

Modelling use case 4 targets application designers and developers (potentially working in partnership with application providers). This is particularly interesting as conventional application development tends to be separate from deployment of an application as a service. However, application design decisions have the potential to be influenced quite significant by considering upfront the eventual deployment of the application on distributed resources with guaranteed QoS. Furthermore, applications can be developed with monitoring and control in mind so they can be easily deployed as a service that can be actively and efficiently managed by an Application Provider.

Modelling use case 7 also merits further comment. First consider the value delivered to a Client by an Application Provider, which includes:

- **Abstraction from application resourcing**. The Application Provider provides a service in terms that the Client understands (e.g. video rendering at a particular frame rate). The Client need not worry about how this service is resourced.
- **Risk when provisioning an application**. The Application Provider uses resources operated by one or more Infrastructure Providers. The behaviour and execution time of applications and workflows is often unpredictable, which 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 Infrastructure Providers.

Now consider that the challenge is how much abstraction can be sustained by the Application Provider? The larger the abstraction then the more uncertainty and hence the more risk the Application Provider takes when delivering guaranteed application QoS. Clients want Application Providers to speak their language but an Application Provider needs to balance how much consumer language is necessary because moving too far into the consumers space increases the risk.

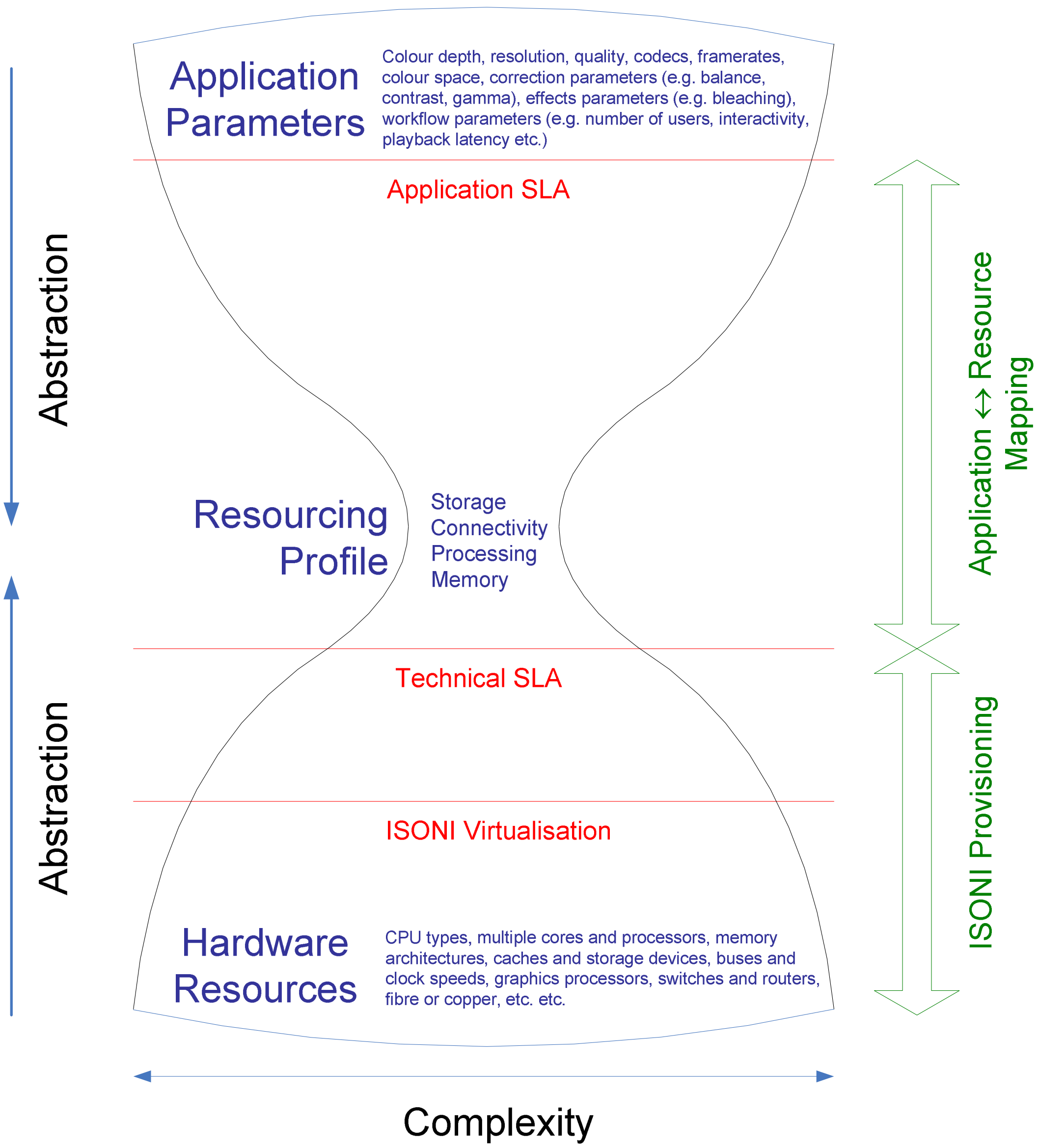


Figure 8: Mapping, abstraction and complexity

An 'hourglass' model of abstraction is shown in Figure 8.

At the application level, the parameter space is large and complex, even for simple applications. For example, as shown in the diagram, 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 (resolution, colour depth, colour space, number of frames etc.), the corrections to be applied (balance, contrast, gamma and special effects e.g. bleaching), and the use of colour correction within a workflow (number of users, interaction required, framerate etc.). The Application Provider uses mappings (e.g. based on the models in this report) to abstract this application-level complexity and characterise the application execution requirements in terms of its need for storage, processing, memory and networking. In effect, the Application Provider is abstracting application complexity when one or more Infrastructure Providers deliver the necessary resources.

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 (e.g. SMP or MPP), networking (e.g. Ethernet or fibre channel), processors (e.g. Intel or AMD, multiprocessor or multicore), storage (e.g. the world of NAS, SAN, DAS, OSD and other storage technologies). Details of the hardware level as used inside ISONI can be found in D6.1.1 and D6.4.1. Other Infrastructure Providers might of course use different hardware infrastructures.

The hardware details are typically abstracted by Infrastructure Provider using virtualisation techniques. The Infrastructure Provider then describes the virtualised resources they make available by using technical SLAs. This further abstracts the resources and describes them in terms of relatively generic storage, processing, networking and memory parameters. For example, processing might be specified in terms of CPU speed and CPU numbers, but not in terms of exactly what CPUs are being used at the physical level (in IRMOS, ISONI does also make this information available). Likewise, I/O (networking or to storage) might be specified in terms of latency, jitter, bandwidth and loss rate without going into details about the physical level, e.g. whether the link is a local Ethernet connection or whether it uses IP over a WAN.

Some form of mapping is needed to convert between characterisation of the application (Application SLA) and the resources needed/available to run that application (technical SLA). Mapping can be difficult and if done by the application provider then it presents a risk to the application provider (e.g. they do not reserve enough resource to service a particular customer, or they overprovision and waste money). Taking on this risk and charging for this as part of the service provided is where the Application Provider delivers value. However, to ensure customer satisfaction and a profitable service, it is important for the Application Provider to quantifiably estimate the uncertainty in mapping and the consequences of getting it wrong.

The intermediate level of generic resource parameters (memory, storage, networking, processing) is very attractive because of the relatively small number of parameters and hence compact and manageable specifications. This makes for a simple and clean separation between the application-level and the hardware-level with the consequent improvement of manageability. However, in the real-world life is never quite this easy. The precise behaviour of a particular application on a particular resource depends on exactly how that application has been implemented and optimised. As a simple example, execution time can be different on two processors with the same clock speed but different architectures. Execution time can also be different for the same processor and clock speed, but different cache sizes. Likewise, the execution time on multiple processors is very rarely inversely proportional to the number or processors and depends strongly on the parallelisation strategy used by the application and any resource management tools.

Therefore, these real world subtleties need to be kept in mind because they are on one hand potentially quite significant but on the other hand very hard to address. In the extreme, a very complex model could be build that accurately reflects the real world, but it could also take longer to execute and consume far more resources than the applications it models!

Further considerations arise for the Application Provider when faced with the decision of how best to service multiple Clients. The SLA with each client will specify the performance of the service (QoS) and any penalties incurred if either party to do not fulfil their obligations, e.g. the Application Provider does not adhere to the QoS specification. The SLA is made with a 'customer' within the client organisation, i.e. someone who is empowered to make contractual agreements. However, the service itself may be used not by the customer but by one or more 'consumers' within the client organisation. Therefore, it is important for the Application Provider to consider how many consumers exist for the service as well as the contractual obligation to the customer.

Keeping within the terms of the SLA incurs a cost to the Application Provider due to the need to use resources from one or more Infrastructure Providers. Given the various levels of uncertainty that exist as discussed in previous sections, the most common way for the Application Provider to be sure of meeting its obligations to its Clients is through simple over-provisioning of resources, i.e. booking resources that it will have to pay for but might never get used.[9] This can be expensive. On the other hand, if the Application Provider reduces its cost by reducing the resources reserved or used from Infrastructure Providers then it increases the risk of not meeting obligations to its Clients and hence incurring penalties. The extreme of this is severe under-provisioning where many of the SLAs with the Clients will be broken. Somewhere in between is an optimum[10]. This optimum will depend on many factors, some of which are not technical e.g. if SLAs are persistently broken then there will be damage to customer relationships and customers may never return to use a service provider again. The optimisation problem is shown in Figure 9.

Real-time application models are potentially very useful to the Application Provider in identifying where this optimum position exists, especially its sensitivity to the uncertainties in the mapping process.

[9] Dynamically adjusting the allocation of resources to the application at run time is an alternative bur relies on having these resources available if required.

[10] For some of the IRMOS use cases, this optimisation may not apply, e.g. because worst-case provisioning is used and there has to be a strong guarantee that an SLA will not be broken otherwise the customer will sue or will never use the service again. However, in the general case, knowing which SLAs might be broken and the impact is an important factor in business decision making.

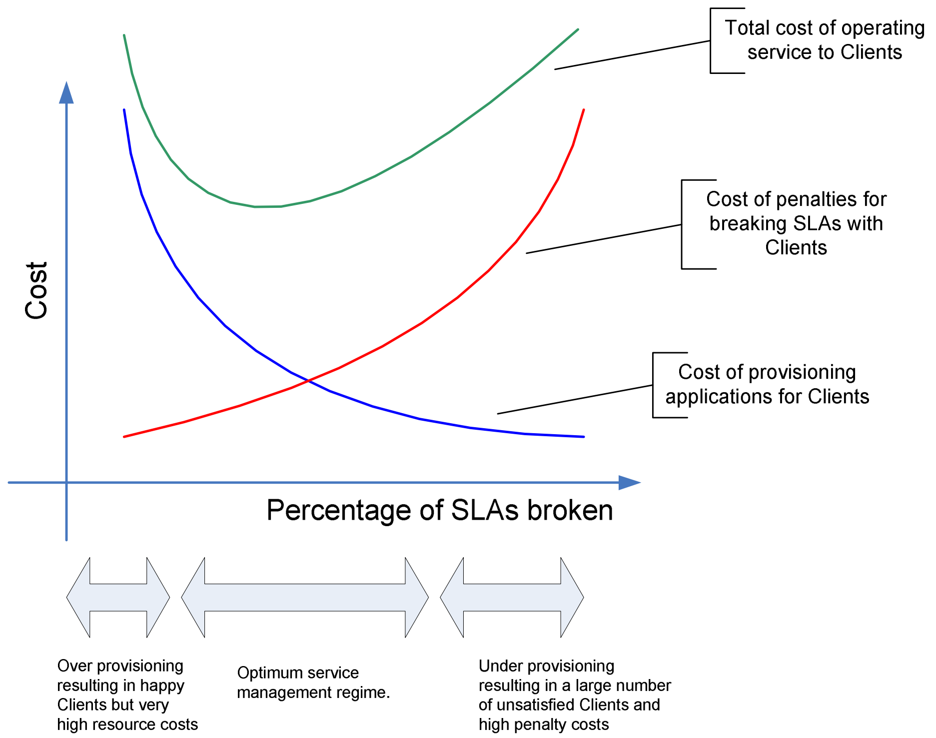


Figure 9: SLAs and provisioning model for Application Provider

3.6. When in the lifecycle are models needed?

The lifecycle of an application typically involves two stages. Firstly, there is a ‘preparation’ stage during which the application is designed, developed, installed and tested at one of more Infrastructure Providers. In IRMOS, these ‘preparation’ activities do not happen in real-time. Secondly, there is an ‘execution’ stage during which the application is executed, monitored and managed according to the SLAs between the parties in the value chain. These ‘execution’ activities happen in real-time.

In IRMOS, for ISONI as an Infrastructure Provider, the preparation stage is not a real-time activity. Furthermore, in IRMOS there is no scope for renegotiation of SLAs after the application has started to execute, i.e. SLAs are static.

However, in general, this need not be the case and SLA renegotiation and evolution can at least be imagined, i.e. SLAs are dynamic. Depending on the capabilities and needs of the Infrastructure Provider and other actors, the activities of discovering, negotiating and agreeing SLAs could be done in real-time.

Modelling Facet	Lifecycle stage	Description
MFWHEN1	Preparation	Models are useful in the initial 'preparation' stages of estimating resource needs in advance of provisioning or executing an application. Models are also useful in the design and development of new applications or the adaptation of existing ones. These use cases have been described in detail in Section 3.5 and are not elaborated further here.
MFWHEN2	Execution	Resource consumption during application execution could be monitored and compared with the predicted consumption. Deviations could then be identified and the application 'adjusted' in some way using a control mechanism to ensure that it still 'completes' in some sense within the constraints of resource availability.[11] The SLAs with Application Provider and Infrastructure Providers could be renegotiated at run time with consequent change in the resources made available for an application[12]. The models would be used to identify the changes necessary to the technical SLA to meet the observed demands of the application. Doing this requires support in the negotiation process and business models as well as the ability for the Infrastructure Provider to adjust provisioning at run-time

Table 5. Modelling facets that describe when in the lifecycle models are needed.

3.7. Real-time Application Scenario

The rest of this report uses a case study to show in a practical way how modelling can be done to address each of the modelling facets listed above.

A hypothetical application scenario is used in the case study and has been designed specifically for evaluating the range of simulation and modelling techniques and tools proposed in this report. The scenario includes many of the characteristics seen in IRMOS applications. The hypothetical scenario is based very loosely on the IRMOS digital film postproduction use case described in D4.1.1.

The models created for this case study are presented in Section 4 along with links back to this section for specific modelling facets addressed.

In the scenario, a film director is working with two post production houses on the application of special effects to a film that was shot on set the previous day. The scenario involves transfer of digital video from a telecine to IRMOS, review of this film by the director and post house staff within a collaborative and interactive session, the application of both video and audio manipulations to the film and then delivery of the results back to the director for final sign off.

[11] For example, a video processing application might be observed to consume more resource than expected and hence not process all the frames needed before resources are expended. In this case the application might then adapt by lowering quality, resolution or some other application parameter to reduce resource consumption.

[12] In IRMOS this is not something currently supported by ISONI

4. Case Study

This section shows how the various techniques described in this document can be applied in practice.

The scenario is only for post-production. The scenario has been carefully engineered to cover a wide range of characteristics, but inevitably the other IRMOS exemplar applications may have some aspects that are not covered, e.g. in eLearning we might need to model the effect of having a large and variable number of users.

The scenario is not intended to be realistic – many activities are clearly wrong - the purpose is only to represent the types of things that need to be modelled and not to create a perfect mirror of the real world.

The steps in the scenario typically have considerable detail. A lot of information is needed before worthwhile modelling can be done. If a scenario is too vague or the values of parameters are not known then it is either impossible to create a model or the outputs of the model are meaningless.

The steps in the scenario include a description of what could go wrong at each stage, e.g. if not enough resource is available. This is important as it captures the negative consequences of not meeting the application requirements. This allows models to be developed that include exception handling or use sensitivity and critical path analysis to target limited resources at the most important steps of a workflow.

The scenario includes a cost model. This is important because when using models for optimisation it is important to know the trade-off between cost and probability of an application executing successfully. Otherwise, the answer to 'how much resource do I need' will always be 'use all the resources available' as this will minimise the probability of failure.

The scenario includes a very simple mapping between application parameters (e.g. frame rate, resolution and colour depth) and resource parameters (e.g. storage and processing). Real-world models are unlikely to be analytical and will involve considerable uncertainty.

4.1. Description of Case Study

A film director is working with two post production houses on the application of special effects to film that was shot on set the previous day.

Activity	Description
A1	Film from yesterdays shooting has been developed overnight and is uploaded to a IRMOS storage service first thing in the morning. Uploading involves an operator using a telecine at the film studio to digitise the film. The output of the telecine is transferred directly over the network to an IRMOS storage service. There are 4 hours of programme material to upload. The maximum digitisation rate of the telecine is 2x real-time, i.e. it is possible to upload the 4hrs of film in 2hrs elapsed time if desired. The telecine can run slower if required so data transfer can take longer. When the telecine is running, a constant bandwidth connection is required. For the purposes of this scenario there is no buffering of telecine output before it is sent over the network.13 If available bandwidth drops by more than 2% then the transfer will stall. If the transfer stalls then its needs to be restarted which takes up to 10 minutes as restarting may require a physical process of rewinding film. Restarting does not require any extra resource, it only introduces a delay. If there are several restarts so that the transfer is delayed beyond the window of network reservation, then transfer is not completed and the transfer is considered to have failed. If not enough storage is available on IRMOS then the transfer will fail completely. The film resolution is 2k, the frame rate is 24fps and the colour depth is 16bit RGB linear.
A2	The film director reviews the uploaded film alongside the post production special effects teams in the two post houses that the director is working with. The director and post production teams are not in the same physical location. The objective of the review is to decide exactly which shots to apply special effects to. Each organisation involved in the review (film studio plus the two post houses) needs to be able to view the footage in real-time, i.e. at 24fps and at the same time (i.e. they each see exactly the same thing as each other). Each organisation needs its own network connection to IRMOS so the total bandwidth needed is 3x the bandwidth needed to transfer the footage at 24fps to one organisation. If the bandwidth available is not enough and causes the frame rate to drop by more than 5%, then viewing has to be restarted. Restarting will typically mean viewing the last 5 minutes of footage again. This of course increases the overall amount of time spent viewing. Whilst review needs real-time transfer to the director and post houses, not every single frame will be reviewed (e.g. only the first few seconds of a scene might be viewed before a decision is made on whether effects are needed for that scene). Only 20% of the footage is likely to be viewed. However, there is a 10% chance that as much as 30% of the footage will need viewing.
A3	Post house 1 does colour correction on some selected shots. Suppose that the number of shots needing colour correction cannot be determined in advance14 (it is decided as part of A2), however it is anticipated that effects will be applied to approximately 35minutes of footage. However, there is a chance that substantially more or substantially less footage will need

[13] In a real world scenario, some form of buffering would take place between telecine and the network, e.g. using memory or disk, so that short-term drops in network performance would not cause a transfer to stop. However, for this scenario we consider the case of no buffering as it allows us to explore how to model consequences of network performance variations.

[14] In the real world for colour correction, probably all the frames will need correction. However, this isn't the case for video manipulation in general, so for this scenario we consider the case where not all frames need correction so we can investigate how to model the general case.

	effects work. The probability that effects will be needed outside of the range of 30-40minutes of footage is 20%. If not enough processing is made available from IRMOS then the colour correction doesn't fail – it just means that not all of the frames will get corrected. Extra storage is needed during this activity for temporary files created during colour correction. This storage is 2x the storage needed for the footage being processed. This extra storage is not needed beyond the end of the activity as the original footage is replaced by the corrected footage and none of the temporary files are kept15. The post house needs to view the corrected footage on average twice whilst colour correction settings are optimised. Viewing is done using real-time streaming rather than downloading, i.e. the bandwidth requirement is fixed. As with A2, if bandwidth available from IRMOS drops too much then viewing will need to be repeated for part of the footage. If colour correction for A3 is not completed by the deadline then the workflow can be considered as failed.
A4	Post house 2 does audio correction on some selected shots. The number of shots needing audio correction cannot be determined in advance (it is decided as part of A2), however it is anticipated that effects will be applied to approximately 120minutes of footage +/- 10 minutes. Extra IRMOS storage is needed during this activity for temporary files. This storage is 0.5x the storage needed for the footage being processed. This extra storage is not needed beyond the end of the activity as the original footage is replaced by the corrected footage and none of the temporary files are kept. The post house needs to listen to the audio at least 5 times and as much as 10 times when developing and applying audio correction settings. This is done by downloading the processed soundtrack rather than streaming, i.e. downloading can be at any rate so the bandwidth requirement is variable. The soundtrack is 10% of the data volume of the digitised film. If audio correction for A4 is not completed by the deadline then the workflow can be considered as failed.
A5	Results from A3 (video) and A4 (audio) are multiplexed into a single AV file and sent to the director along with the original footage which has been transcoded (A6). This is sent as a single bundle that is downloaded to the director as a file rather than streamed, i.e. bandwidth can be varied provided that the download completes in time. If the download doesn't complete by the deadline for A5 then the workflow is considered failed.
A6	The original footage is transcoded into a compressed form to be sent to the director (A5) along with the version which has had the effects applied. This is so the director can do a simple before/after comparison of the original footage with the corrected and edited footage. Transcoding is used to reduce the data volume (compression) and hence speed up/lower the cost of network transmission. However, transcoding requires processing power. The higher the compression, the more processing resource is needed. The maximum compression that can be achieved is 8x. If at least 98% of the original footage is transcoded then A6 is considered successful, although ideally 100% should be done.

Table 6. Description of activities in the case study.

[15] In a real world scenario the original footage and perhaps intermediate edits will often be kept for as long as possible and may even be archived. However, for this scenario we take the simple case of long-term storage not being required.

4.1.1. Workflow diagram

The dependencies between activities in the workflow are shown in Figure 10. The semantics of the arrows are that an activity A cannot start until all other activities with arrows that lead into A are all complete. So, for example, A5 cannot start until A3,A4 and A6 have all finished.

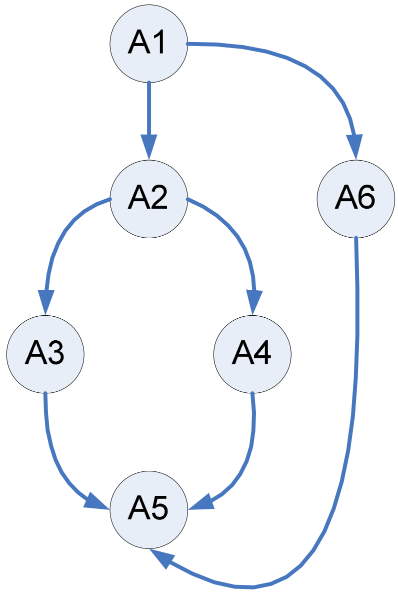


Figure 10: Activity dependencies

4.1.2. Input constraints

An initial set of constraints for the workflow might look like this:

Activity	t start	t finish	Min duration	Max duration	Comment
A1	7:00		2:00		7:00 is when the telecine operator is first available for upload of yesterday's film. There are 4 hours of film to upload and the maximum rate the telecine can digitise the film is 2x real-time, i.e. the minimum upload time is 2 hrs.
A2	10:30	12:00	1:00		This is when the Hollywood director is available
A3	13:00	15:00	2:00		This is when the visual effects team is available.
A4	13:00	15:00	2:00		This is when the audio effects team is available.
A5		18:00	2:30		The end time of the task is a hard deadline on when the edited film must be sent back to the Director
A6			3:00		

Table 7. Initial constraints for the workflow in the case study.

The constraints are a result of external factors, e.g. because a certain set of people are needed to perform an activity in the workflow and they are only available at certain times.

t_{start} and t_{finish} are hard constraints. The semantics are that the activity must start at t_{start} and no later and must finish at t_{finish} and no earlier. If either t_{start} or t_{finish} are not specified then the time that the activity can start or finish is flexible.

The min and max durations for the activity are estimates of how long the task is likely to take even if unlimited IRMOS resources are available. So, for example, if a team is reviewing the film from yesterday's shoot this might be estimated to take at least 30minutes.

Note that these constraints are not at this stage a result of any limitation due to resources available from IRMOS.

Given the entries above, it is possible to deduce further entries in the table given the dependencies between the tasks.

Activity	t start	t finish	Min duration	Max duration	Comment
A1	9:00		0:30	1:30	9:00 is when the team is first assembled for the start of the dailies review
A2	10:30	12:00	1:00	1:30	This is when the Hollywood director is available
A3	13:00	15:00	2:00	2:00	This is when the visual effects team are available.
A4	13:00	15:00	2:00	2:00	This is when the audio effects team are available.
A5		18:00	2:30	3:00	The end time of the task is a hard deadline for when the edited film should be sent back to the Director
A6			3:00	6:00	

Table 8. Initial and derived constraints for the workflow in the case study.

For example, we know that A6 can't start until A1 is complete. The earliest A1 can complete is 9:30. A6 needs to be complete before A5 can start since the outputs of A6, A3 and A4 are bundled together and sent to the Director in A5. The latest A5 can start is 15:30 as it takes a minimum of 2:30 to perform. Therefore, the maximum duration for A6 is 6 hrs.

4.1.3. Mapping

For the purposes of this scenario '2k' resolution film means 1998 ×1080 pixel resolution when digitised, i.e. approx 2Mpixels per frame. There are several definitions of 2k[16] and the one chosen here is arbitrary and only made to make the arithmetic simple!

The data volume of digitised film is:

Volume = framerate * colourdepth * resolution * duration

Framerate is in frames per second (fps)

Colour depth is in bits per pixel per channel (3 channels for RGB)[17].

Resolution is in pixels per frame

Duration is in seconds

Volume is in bits (divide by 8 to get bytes)

So, for example, 20minutes of 2k film at 24fps and 16bit colour depth requires:

24 * 16 * *3 * 2 * 20 * 60 = 2764800 Mbit = 345.6GByte of storage

The processing that is needed when applying special effects to the film is:

Processing = algorithm factor * number of frames * resolution of each frame

Resolution is in Mpixels per frame

Number of frames = duration * framerate

Algorithm factors:

1.5 for colour correction

2.5 for audio correction

[16] 2k resolution includes: 2048x858 (aspect ratio of 2.39:1, known as CineScope or 'scope for short as well as 'anamorphic' although strictly speaking this refers to lenses that are not spherical and are used to 'squeeze widescreen shots onto conventional 35mm film), 1998x1080 (aspect ratio 1.85:1) and 2048x1080 (2k container within which both 2048x858 and 1998x1080 will fit). See Digital Cinema specifications for details http://www.dcimovies.com/DCIDigitalCinemaSystemSpecv1_2.pdf

[17] 16 bits per colour channel with RGB colour space is just one of many ways in which colour information might be encoded. Things get a lot more complicated when chroma subsampling is used to give luminance information a larger share of available bandwidth than colour information (e.g. 4:2:2 as used by various HD cameras). See http://en.wikipedia.org/wiki/Chroma_subsampling

0.5 for transcoding with 2x compression

1.0 for transcoding with 4x compression

3.0 for transcoding with 8x compression

Processing is in MFLOPS (mega flops)

So, for example, colour correction of 30minutes of 24fps 2k film will require:

1.5 * 24 * 30 * 60 * 2 = 130 GFLOPS (giga flops) of processing resource

The processing time needed to apply effects is the processing needed divided by the rate at which IRMOS can do processing (measured in MFLOPS per second).

The IRMOS resource reservation for processing is expressed in terms of processing rate required * duration of processing, e.g. a reservation might be 10 MFLOPS per second for 1 hr, which equates to 36 GFLOPS of processing in total.

The time taken to transfer data over the network is:

Transfer time = Data volume/Bandwidth

Volume is in Mbit

Bandwidth is in Mbit/s

Time is in seconds

So, for example, 50GByte transferred over a connection with 10Mbit bandwidth will take just over 11hrs.

If digitised film needs to be transferred ‘in real-time’ this means that transfer of 1 hr of digitised film should take 1 hr of elapsed time.

4.1.4. QoS model

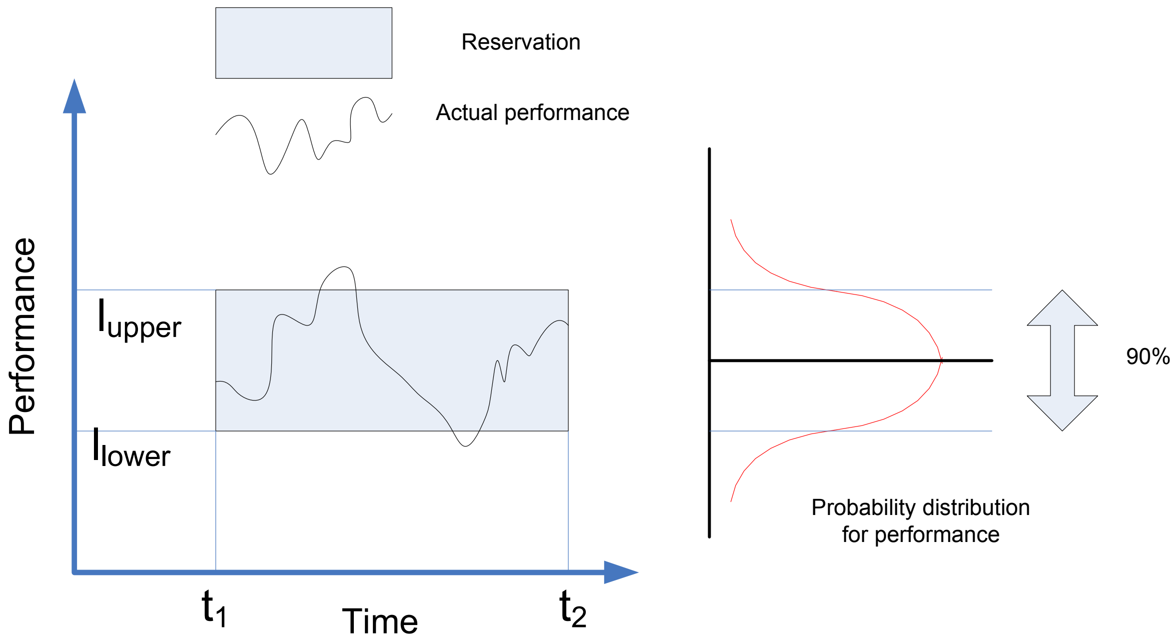


Figure 11. QoS model used in the case study.

The quality of service model for the case study is simple. The resource offered by a service provider is guaranteed to be between limits l_{upper} and l_{lower} for percentage p of the time. For example, bandwidth available on a network connection might be guaranteed to be between 90 and 110 MBit/s for 95% of the time that the user has reserved the connection.

4.1.5. Cost models

In the case study, the assumption is that a user of IRMOS pays for what they reserve, not what they use. This of course depends on the business model, but considering all the variations of charging models possible is outside of the scope of this report.

Suppose the cost of making a reservation for bandwidth B for time t is:

> *Cost bandwidth = 0.001 * B^2 * t where t is in seconds, B is in Gbit/s and c is in pounds sterling.*

So, for example, a connection of 2Gbit/s for 1 hour will cost £14 and is sufficient to transfer 900GB of data. However, transferring the same amount of data in half the time will cost nearly £60. In other words, to get the same amount of data transferred over the network it is cheaper to use less bandwidth for a longer period of time than it is to use a high level of bandwidth, but for a shorter length of time. We deliberately introduce non-linear scaling of cost with bandwidth in the scenario to avoid unrealistic situation where there is no difference in cost between having a slow connection for a long time and an extremely fast connection for a short time.

Suppose the cost of making a reservation for processing video data is:

> *Cost of processing = 0.5 * processing needed, where processing needed is in gigaflops and the cost is in pounds sterling.*

So, colour correction of 30minutes of 24fps 2k film will cost £65. Note that the reservation itself is expressed as processing rate * duration of the processing.

Suppose the cost of storing video data is:

> *Cost of storage = 0.001 * data volume * t, where data volume is in GB, t is in hours, and cost is in pounds sterling.*

So, storage of 900GB of data for 10 hours will cost £9.

The business SLA from the service provider provides two levels of possible QoS. One of the purposes of modelling is to allow service consumers to perform a cost benefit analysis, hence variation in such terms should be considered within the scenario. The probability that the resource IRMOS provides will deviate outside of the range l_{upper} and l_{lower} can be halved if desired, but this doubles the cost. In other words, if it costs X for a resource reservation where IRMOS has a 94% probability that the resource delivered, then it will costs 2X for the same reservation but with a 97% probability and likewise 4X for a 98.5% probability etc.

QoS terms from the provider can be represented in the form of probabilities. It may be the provider does not explicitly provide these kinds of probabilities, but it is assumed they can be calculated from the SLA. The probabilities associated with each of the cost models above are:

Bandwidth	94%
Storage	100%
Processing	98%

The probabilities are specified relative to some predefined time interval, which ensures that metering data can be used consistently when monitoring against an SLA. For simplicity we will assume this interval in an hour. The probabilities are the chance that the delivered resource will stay within the limits specified for each hour that the resource is reserved. So, for example, if a networking resource is booked for 2 hours then there is 94% chance that the agreed bandwidth will be provided continuously for all of the first hour. There is then again another 94% chance that it will be provided continuously for the second hour. This obviously means that the overall probability of remaining with the limits specified will go down as the duration goes up. In other words, the longer the resource is used for, the higher the overall chance that there will be at least one deviation within the period.

So, for example, the processing resource needed for colour correction of 30minutes of 24fps 2k film will cost £65 where there is a 98% probability of that processing actually being delivered by IRMOS. The same processing with 99% probability would cost £130.

Notes:

When considering storage of the uploaded film in Activity1 note that the film is needed in subsequent activities. Therefore, storage resources will be needed in the workflow beyond the activity in which they might be first. So, for activity A1, storage resource is needed until the end of A6 whilst the bandwidth resource to do the upload to IRMOS is only needed for the duration of A1.

Network costs are only incurred when data is transferred to/from IRMOS, e.g. uploading digitised film or viewing the results of colour correction. This may mean that if a resource is not used at all then a refund is available. We assume for this scenario that if part of a resource is used then the user will be charged the full cost of that resource irrespective of whether the full amount is used. Network costs are not incurred for operations local to IRMOS, e.g. when a colour correction application needs to load in film data that is on IRMOS storage.

Some activities don't instantaneously fail if IRMOS resource delivery deviates outside of the agreed bounds. In these cases, it can be that the work takes a bit longer to complete but can still be done if there is still time before the hard deadline for the overall activity. One option is to reduce the chance of the activity failing is therefore to plan for the best case of the activity completing a little earlier than absolutely necessary and reserving a bit of extra resource up to the hard deadline so that the work can overrun in the worst case of resource not being delivered by IRMOS as expected. The other option would be to pay more to reduce the probability of deviation. Small deviations may cause failure if the reservation times are not given some 'padding'. How much padding is needed will be dependant on what the users believe their content is worth compared to the price needed to incorporate such padding. Models should help users to determine what the likelihood of failure is given their choices in over-provisioning.

4.2. Simulation and modelling outputs

The point of models and simulations built to describe the case study is to allow exploration of how the application will behave under a range of different circumstances.

The outputs of the models and simulations should allow a reservation plan for IRMOS to be constructed, i.e. what resources are needed and when, along with a breakdown of the cost of each activity and the likelihood that the activity might fail (not complete by its scheduled end time). So, for example, it might be that the plan says that resources will be reserved for A1 so it starts at t1, ends at t2, has bandwidth b, storage s, and that this will cost X and has a probability of being successful of p.

The total cost and the overall probability of failure should also be calculated and it should be possible to investigate how these vary when individual parameters are varied (within the constraints) to allow the trade-offs between cost and probability to be investigated, e.g. how much can the chance of success be increased by spending more money on resources?

5. Case study: models of high level workflow

At the highest level the scenario describes a workflow indicating a set of different roles and task, the tasks are carried out using tools that are built from services running on the IRMOS platform. The following sections describe the scenario using different description languages. At this level the real-time aspects that are relevant are those related to schedules and deadlines for the start and end of the different tasks. For further details of these languages see Annex A.1.

5.1. YAWL

Figure 12 shows the scenario, at a high level, described using YAWL. The YAWL editor (the tool used to draw the diagram) does not allow for task to role allocations or the timers for the tasks. This information can be added to the model, and will then be used by the YAWL workflow engine when the workflow is executed. Since the allocation of tasks to roles is not visible in the diagram, it is hard to denote that the "Review Film" task is actually performed by three actors at the same time. Regarding timing of tasks, YAWL supports defining exact start times, and timeouts (time from the task starts until it has to be finished).

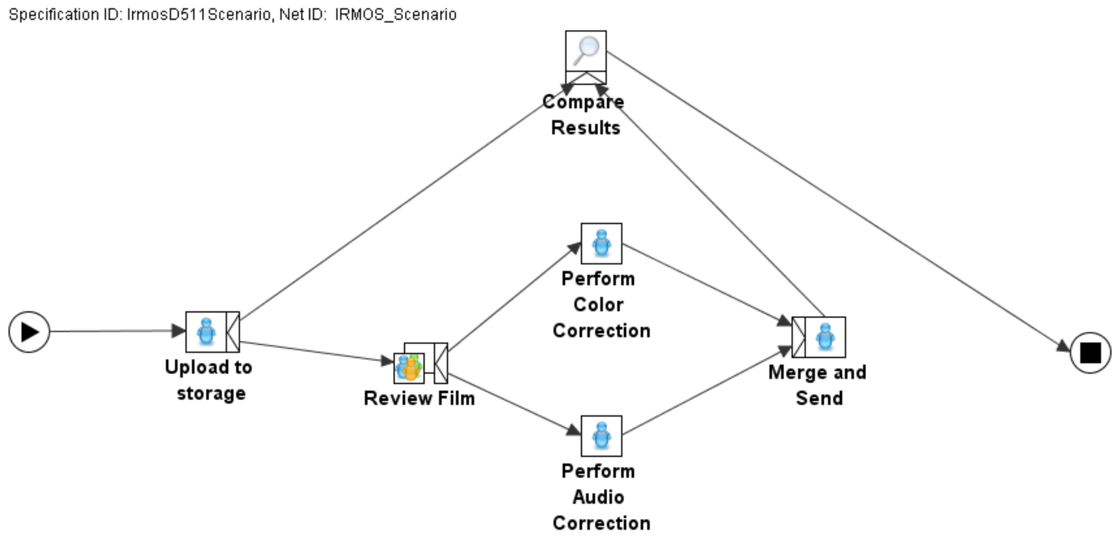


Figure 12: YAWL diagram of scenario

Each of the tasks in the diagram above can be detailed further in new nets of tasks.

5.2. BPMN

Using the Business Process Modeling Notation (BPMN) to describe the same scenario (diagram rendered in Figure 13), it is possible to indicate what roles perform what tasks using the swimlane construct. The parallel review of the film by the three actors can thus be visualized (albeit using a comment to emphasise the "real" parallelism). A timeout of a task can be indicated using a Timer event, as exemplified for the "Upload Film" task, which has to be completed in N hours from start.

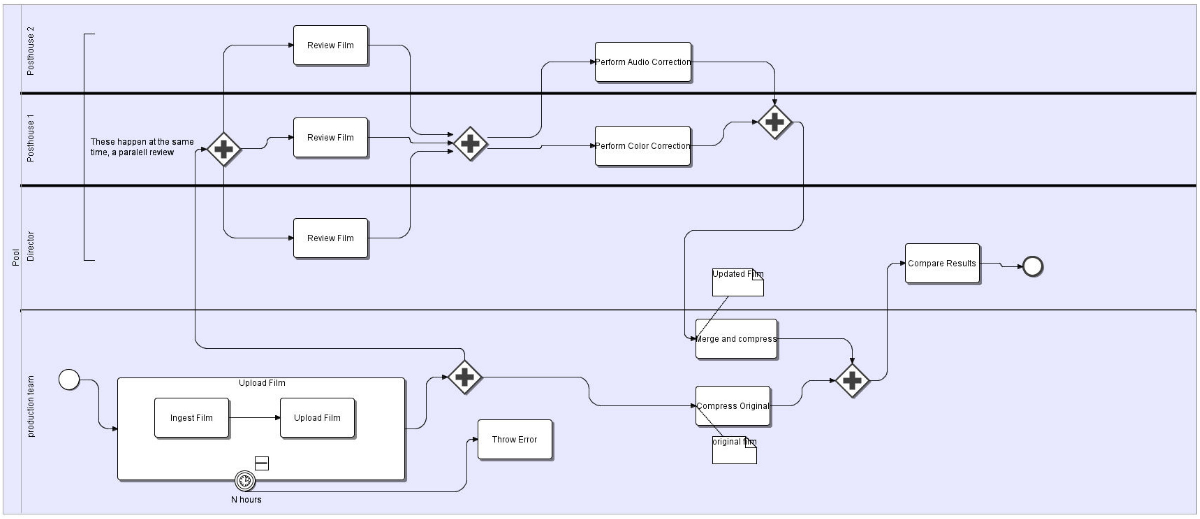


Figure 13: BPMN diagram of the scenario

5.3. BPEL

Business Process Execution Language (BPEL), which is an XML-based language for specifying interactions with Web Services (WS). It functions as a WS orchestrator for the implementation of business processes by using exclusively WS interfaces. As a common programming language, BPEL supports all necessary commands like *while* and *for loops*, *conditions* as well as structures for task sequencing, correlation of messages and process instances, fault handling and timed events. More details concerning BPEL can be found in paragraph 9.4. Figure 37 shows the scenario modelled using Netbeans 6.5 which complies with specification 2.0 of BPEL.

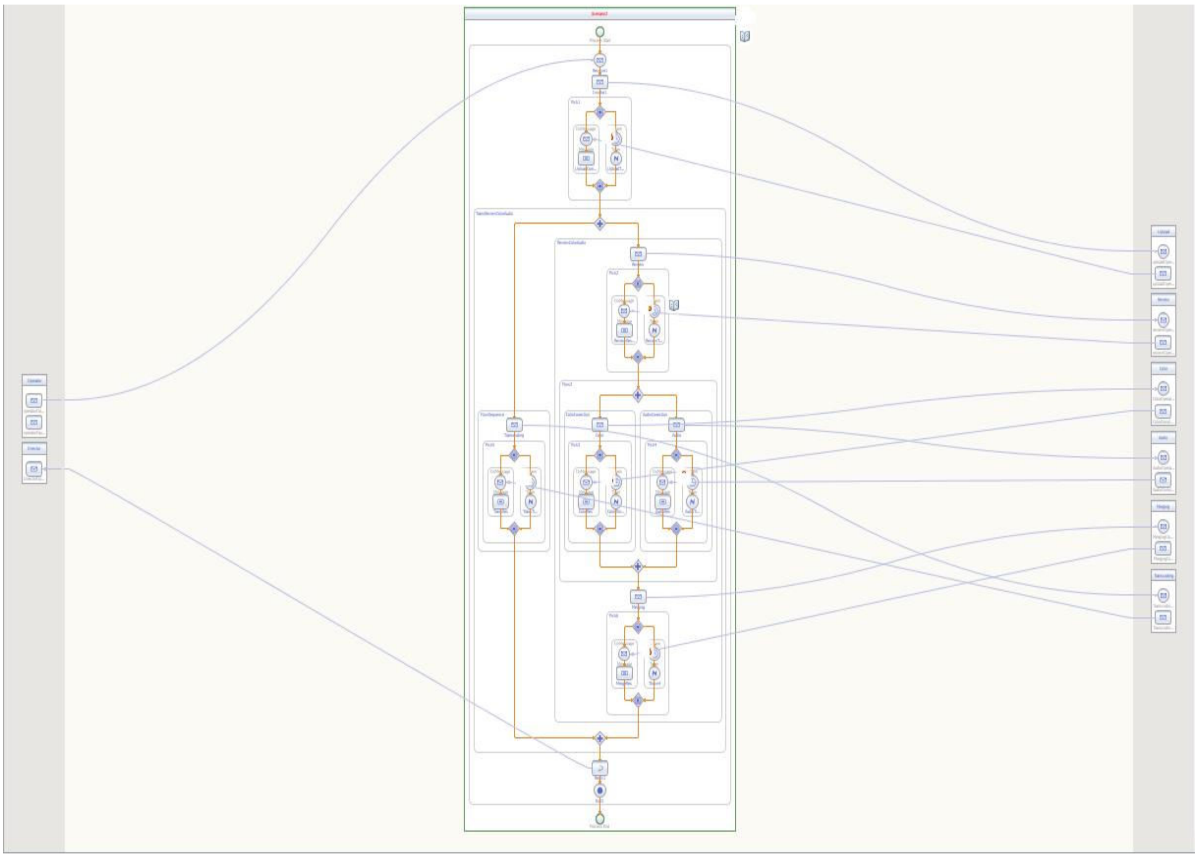


Figure 14: Services Interactions – BPEL modelling

Following the SOA approach all tasks/actions of the proposed scenario are modelled as web services and described by a WSDL file. The following services have been identified:

- *Operator.* This is the service that represents the operator of the application as mentioned in the scenario
- *Upload.* This is the service that is responsible for the uploading of the digitized film to the storage.
- *Review*. This is the service that is responsible for streaming the film to the three locations (director, colour and audio correction post houses). Although this could be considered as a more complex task, we have chosen to depict it as a single service for simplicity reasons.
- *Colour and Audio correction*. These services are responsible for the color and audio correction done in the two post houses.
- *Merging.* This service conducts the merging of the color and audio correction services.
- *Transcoding.* This service is responsible for the compression of the original film.
- *Director.* This represents the actions that need to be taken by the director.

Since the services listed above are not yet implemented they can be considered as abstract services. The process defines the sequence in which the services are invoked and their results, if any, are used by the following services.

The BPEL structures that are used are:

- *Receive*. This is used to wait for an incoming message from a web service.
- *Invoke.* This is used to send a message to a web service so as to invoke a certain operation on it.
- *Pick.* This is used to wait for a message or a timed event. In more detail, this structure is used in order to wait for a message from a web service that notifies of the completion of the given task. If the message is not received during a predefined time period a fault is thrown. This is used in order to model the timing requirements that are described in the scenario.
- *Flow.* This structure is used in order for different tasks to be executed concurrently.
- *Assign.* This is used in order to copy data between variables so that the output of a service can be passed as input to another service.

The process is made clearer in figures 38-41.

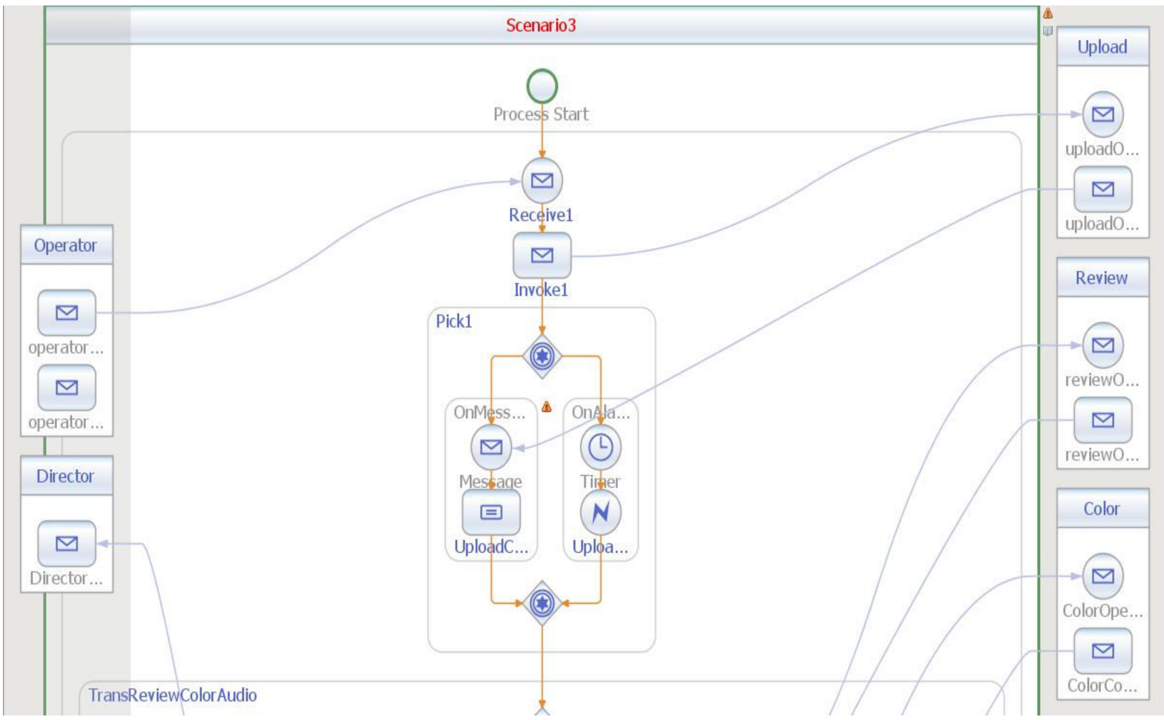


Figure 15: Process start

In figure 38 the start of the process is depicted. The operator service initiates the workflow by sending a message that is received by the enactment engine. Following this, the upload operation of the upload service is invoked. After this the pick structure is used. This contains two branches: the OnMessage branch and the OnAlarm branch. The first one waits for a message by the Upload service by which it is notified of the correct completion of the service and receives its output. The OnAlarm branch has a timer function which contains a time period for which the process awaits for a message by the upload service. If a message is not received within this time frame a fault is raised. It is therefore clear that timing constraints can be modelled by an enactment engine based on BPEL. The only constraint is that the services need to implement an operation that will be responsible for notifying the engine of their correct completion.

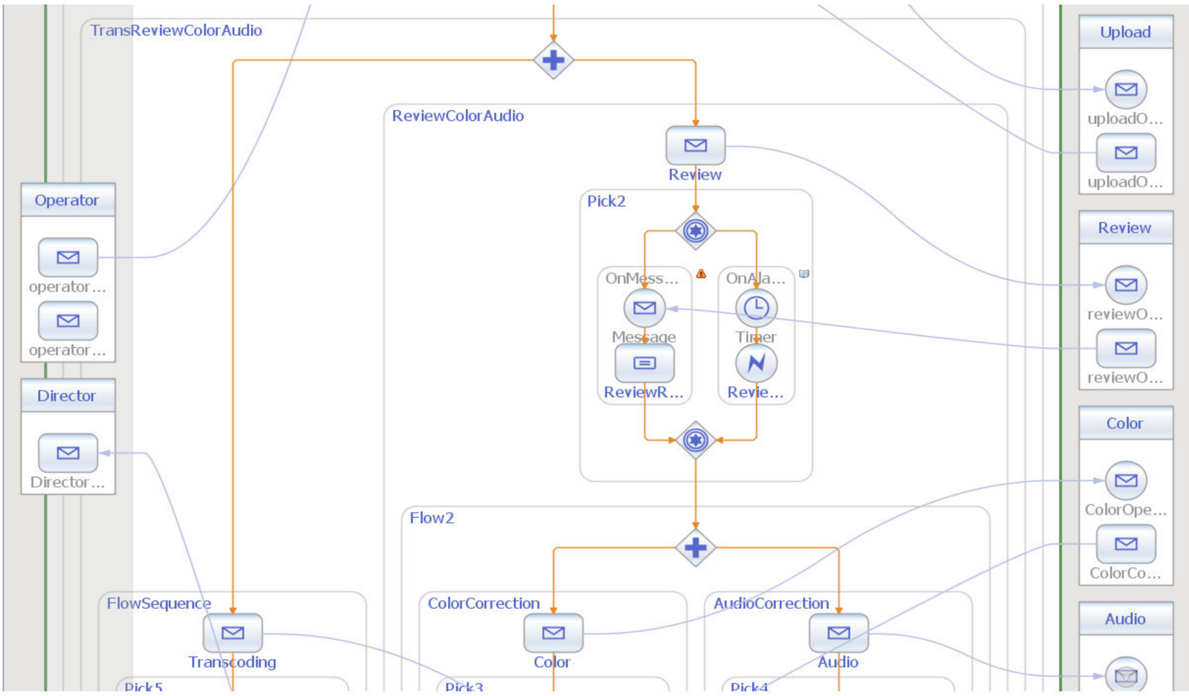


Figure 16: Review task

In figure 39 the use of a flow structure is visible, by which the concurrent execution of tasks is made possible. One branch of the structure contains the transcoding task, while the other is consisted of the review task, followed by the concurrent execution of the colour and audio correction tasks. Again, pick structures are used to model the timing constraints of each task. Figures 40-41 show the rest of the process and are included for completeness.

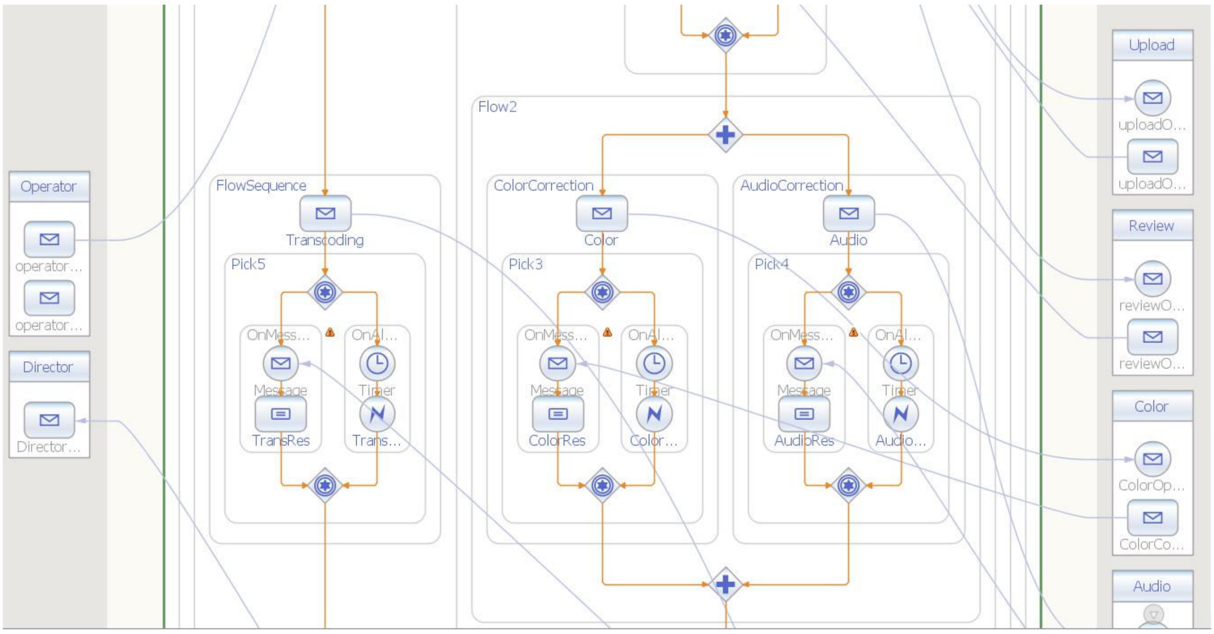


Figure 17: Transcoding, Colour and Audio Correction

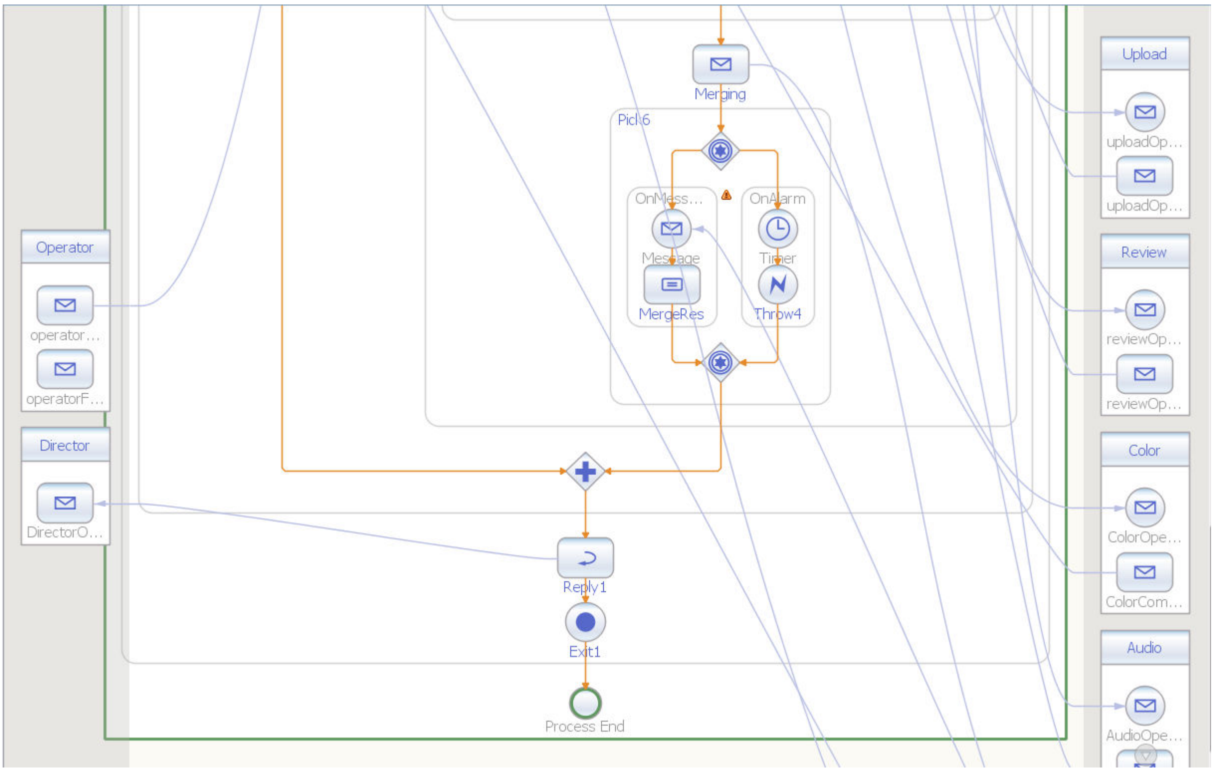


Figure 18: Merging and Process End

5.3.1. Extensions used to model QoS constraints

From the above we can safely say that BPEL can be used as the basis for the enactment engine of IRMOS for it is capable of specifying time restrictions and fault handling, through the use of the pick structure and fault and event handlers respectively. It should be mentioned that the current version of BPEL4WS does not support the modelling of other real-time or QoS constraints that could be needed for the orchestration of real-time critical workflows as the BPEL4WS specification does not include elements for describing QoS constraints neither for the overall workflow nor for the individual web services that comprise the workflow. In order to take full advantage of the BPEL4WS language and avoid defining a new language for QoS-aware workflows the following techniques could be used:

1. A separate document is used to describe the QoS requirements.
In order to facilitate real-time QoS awareness, a secondary language could be developed. This allows using the standard BPEL4WS language along with a secondary document that includes QoS requirements related to elements within the standard BPEL document. This approach has been used successfully to support QoS-aware workflow management in GridCC project [31]. The latter includes a QoS-aware workflow management system (WfMS) for orchestrating Web services with real-time QoS requirements, which builds heavily on BPEL and XPath notation [32][33]. In more detail, a secondary document is composed of sets of QoS constraints, which are linked to separate BPEL4WS activities that could be both basic activities and structure activities using the following basic structure:

```
< ?xml version = ”1.0” encoding = ”UTF−8”? >
< QoSRequirements >
      < QoSConstraint >
            < XpathReference >
                  “XPath to Service 1 in BPEL document”
            < /XpathReference >
            < Resources >
                  < CPU Speed > 2 GHz < /CPU Speed >
            < /Resources >
            < MaxDurationTime > 100 secs< /MaxDurationTime >
            < Reliability > 100 < /Reliability >
      < /QoSConstraint >
      < QoSConstraint >
            < XpathReference >
                  “XPath to Service 2 in BPEL document”
            < /XpathReference >
            ...
      < /QoSConstraint >
      ...
< /QoSRequirements >
```

In more detail, the above example specifies several QoS constraints elements, which are independent of each other in a QoS document, and there is a different XpathReference pointing to different services of the standard BPEL document and consequently each of these services must match the corresponding QoS requirements. In this case of Service

1, the overall CPU capacity should be 2GHz on fully reliable resources, mapping to overall time should be less than 100 seconds.

The approach of using a separate document for specifying QoS constraints is versatile and flexible enough to facilitate single service, multiple service as well as generic workflow requirements both high level and low level ones. Also, as the proposed structure is based on XML it could be extended to facilitate a range of new QoS constraints that could be required for supporting QoS-aware workflow management within the IRMOS framework such as the notion of reservation duration, execution duration, relaxed start time, multithreaded services, multicore CPUs, exception handing and user-defined exception handling, etc.

It should also be noted that the usage of an accompanying document for specifying QoS constraints in conjunction with XPath-based notation, promotes loose-coupling between the basic workflow language and the secondary structure and makes this approach adoptable to any XML-based languages and a good candidate for a SOA-oriented system such as the IRMOS platform.

2. Extension of standard BPEL4WS.

Rather than using a different document, as discussed above, an extension could be embedded inside the standard BPEL4WS document. To this end, the standard BPEL language is extended to express QoS and QoS-based service composition. For dealing with some of real-time aspects of BPEL, additional constraints on the services are imposed by existing approaches [34][35][36][37]. In more detail QoS-related attributes are introduced into the basic structure of BPEL4WS with regards to the execution time, cost, availability and reliability among others. In most of the existing approaches the BPEL activity that is being extended to facilitate real-time restrictions is the <invoke> activity because it is the one that is actually used during the selection of candidate locations for execution. A good example of such an extension is the one that follows as presented in [38]:

```
<process>
    <!-- details ...-->
      <invoke>
          <QoS-based>
                <QoS-constraints>
                      <QoS-Constraint>
                            <QoS-Name> QoS name </QoS-Name>
                            <QoS-Op> Relational Operator </QoS-Op>
                             <QoS-Value> Constraint value of QoS </QoS-Value>
                      </QoS-Constraint>
                      ...
                </QoS-Constraints>
                  <QoS-Selection>
                      <QoS-WeightTerm>
                         <QoS-Name>QoS name</QoS-Name>
                         <QoS-Weight>Weighted value of QoS</QoS-Weight>
```

```
                    </QoS-WeightTerm>
                    ...
                </QoS-Selection>
            </QoS-Based>
        </invoke>
        <!-- details ... -->
</process>
```

This structure introduces a new child element inside the <invoke> element, named <QoS-Based>. This new element has two child elements:

(1) <QoS-Constraints> element, which is used for facilitating any QoS constraints for the service and may contain multiple <QoS-Constraint> child elements for each of the required constraints. Each individual QoS constraint is described by three attributes: <QoS-Name> which is the name of the constraint, <QoS-Op> which is the relational operator (i.e. <, >, =, etc) and <QoS-Value> which is the value of the given constraint.
(2) <QoS-Selection> element, which is used for expressing any preferences by assigning a weigh of significance to each constraint included in the <QoS-Constraints> element and it may contain multiple <QoS-WeightTerm> tags. Each of these <QoS-WeightTerm> tag has two sub- tags: <QoS-Name> which is the same as the name of constraint and <QoS-Weight> which is the value of the weight and therefore the sum of all weight values should be 1.

Such a generic structure could be easily extended to facilitate theoretically any range of new QoS constraints and preferences that could be required by simply adding new records in the pool of possible values that the defined tags could take. However, compared to using a separate document for maintaining QoS-related information, although this approach works well on single-service level, it is actually less flexible because multiple service and/or overall workflow requirements, need the introduction of new elements in different locations of the standard BPEL document and consequently the extension of many more BPEL elements. Another disadvantage of this approach is that it is strictly depended on the XML-based language that is being extended each time, and therefore does not promote loose coupling. However, this approach could be faster in practice, because it allows for faster processing without having one XML document referencing others during manipulation.

Both the aforementioned techniques can be used to incorporate MARTE elements to BPEL. For example, the NFPs that MARTE proposes can be seen as QoS parameters described above. For example, let us consider the proposed resource types, as proposed by MARTE in [39] and demonstrated in the following figure:

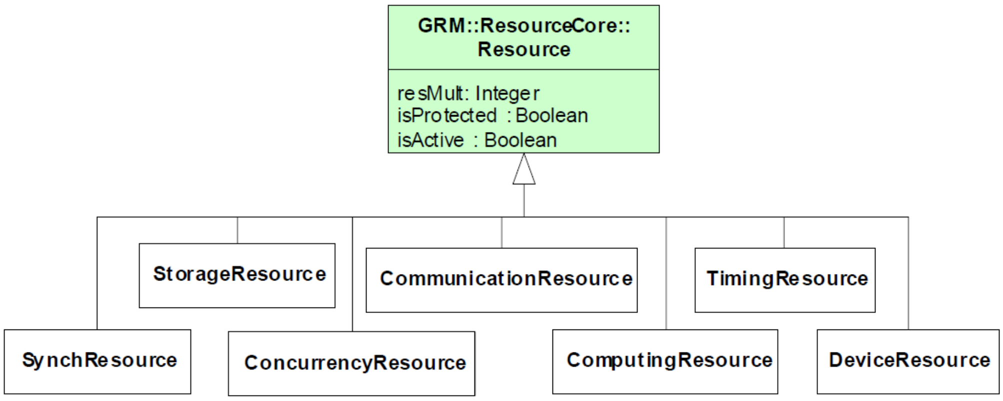


Figure 19: ResourceTypes package of MARTE [39]

In the special case of MARTE's StorageResource resource type, the following XML-based description could be used:

```
<StorageResource>
	<resMult>1</resMult>
	<isProtected>false</isProtected>
	<isActive>true</isActive>
	<size>
		<value>120000</value>
		<type>KB</type>
	</size>
	<refClock>
		<value>5</value>
		<type>ms</type>
	</refClock>
</StorageResource>
```

Where the <resMult> tag is used to express the limited nature of an aggregated multi elementary resource and when used, it indicates the maximum number of elementary units' instances of a particular type of resource that are available through its corresponding service. The <isProtected> and <isActive> tags have boolean values with the <isProtected> tag implying the necessity to arbitrate access to the resource or its services, while <isActive> means that it has its own course of action. The <size> tag has two child elements: <value> which defines the size of the resource and <type> which defines the metric used. Finally, the <refClock> corresponds to the pace at which data is updated.

In the case of using the secondary document as Table X, the <StorageResource> element as described above fits well as a child element of the <Resources> element, whereas in the case on extending the standard BPEL description, although the proposed extension in Table XX cannot used as it is, it could be changed as to include more attributes in the

<QoS-Constraint> element as to cover every aspect of the proposed StorageResource type and all MARTE's resource types in general.

5.4. Summary

Modelling Facet	Specific Facets addressed	Description
What is being modelled	MFWHAT1 to MFWHAT4	The workflow including: • What services are part of the workflow • How services take part in the workflow – Control flow • What messages are exchanged between the services – Data flow • Timing constraints of when each service needs to have finished • Fault handling
Who will use this model	MFWHO2 and MFWHO3	The application provider when constructing the abstract workflow The client when defining the QoS constraints of the abstract workflow
When will they use it	MFWHEN1 and MFWHEN2	The application provider when constructing the application The client when defining the QoS constraints of the abstract workflow When the application is executed the workflow is used by the workflow execution engine
Why is it useful	MFWHY1	The workflow is used as one input in constructing the final VSN description. The workflow is also useful in describing tasks where ACs are used by different roles to perform a business process. The workflow needs to schedule activities in a precise manner with respect to low level resources. Scheduling can be derived directly from the workflow model.

Table 9. Modelling facets addressed by workflow model

Technique used	Why this technique?	Further information
YAWL	• YAWL has one of the most complete set of workflow patterns amongst currently available tools. • The YAWL graphical language is very expressive with respect to workflow composition	A.1.3

BPMN	Gives the possibility of describing allocation of tasks to roles or actors in the diagrams. Therefore providing the possibility of defining models applicable to multiple actors.	A.1.3
WSBPEL	• Fast becoming the industry norm and extensively used as a workflow execution language • Uses exclusively WS interfaces • Can be extended in a standard manner to include timing constraints • Fault handling can be defined • Can be extended in a standard manner to incorporate QoS	A.1.4 and A.1.3

Table 10. Techniques used for workflow model

6. Case study: UML models of services

This section describes the scenario using UML and a set of UML profiles. The base of the service modelling is done using the SoaML [52] profile that is under standardisation at the OMG. For further details on UML profiles see Annex A.6.

6.1. Capabilities

SoaML allows for defining the capabilities that are needed within a specific service architecture. This allows for defining the functionality needed without having to specify who provides the functionality and how it is used. An analysis of the scenario at hand can result in the following capabilities diagram; the dotted lines indicate a dependency between capabilities:

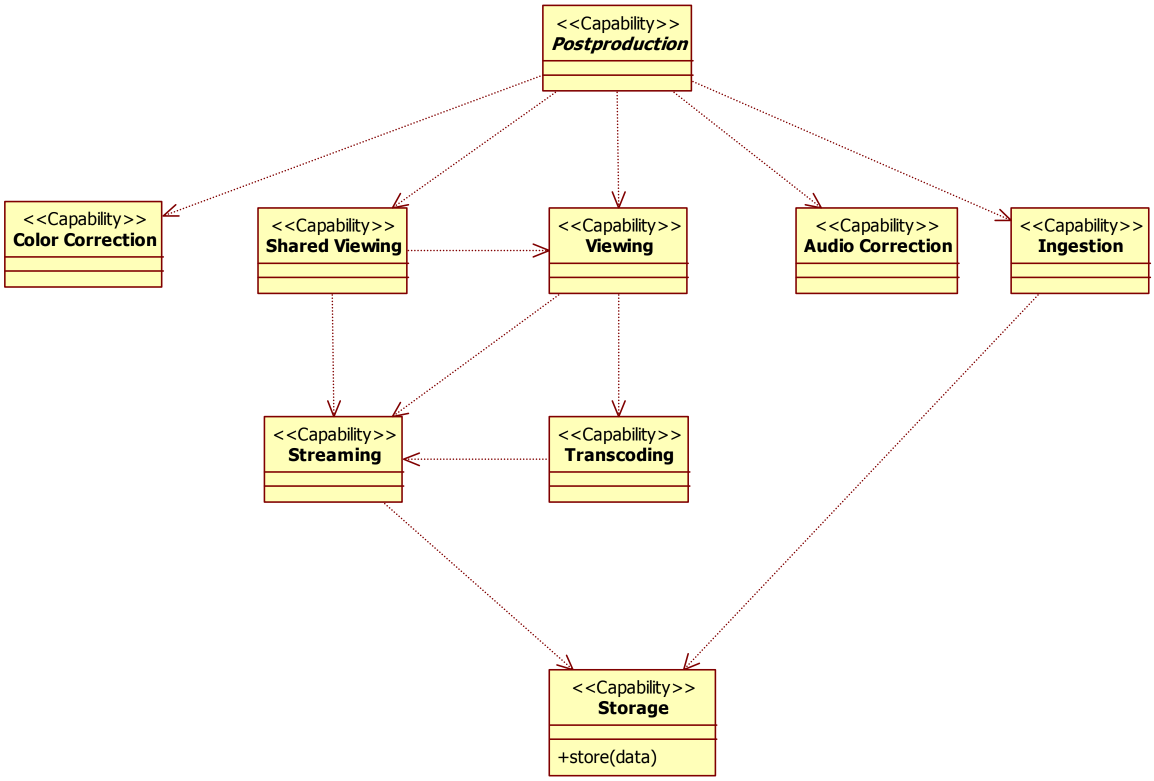


Figure 20: Capabilities of the post production scenario

6.2. The service architecture

The purpose of A SoaML Services Architecture is to illustrate how different entities work together for some purpose. These descriptions are based on the roles that are involved in the collaboration. Figure 21 shows an example of this based on the involved services to allow for shared viewing in the scenario of this document (note that the collaboration should have the <<ServicesArchitecture>> visible above the title, but the used tool did not support this).

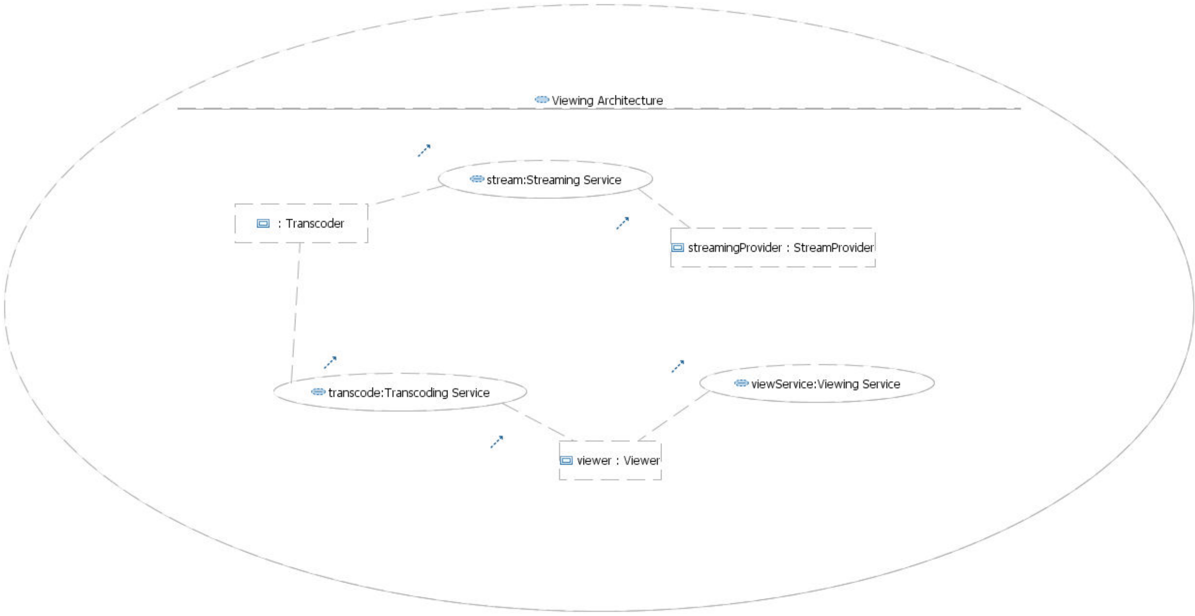


Figure 21: Example services architecture

The squares in the diagram above represent roles in the service architecture. These roles can be typed using the capabilities that are defined for the architecture (as described in the previous chapter).

The ovals inside the large oval represent service contracts that define how a service is to be used. Taking the Streaming Services as an example, it is defined as shown in Figure 22 using yet another UML Collaboration.

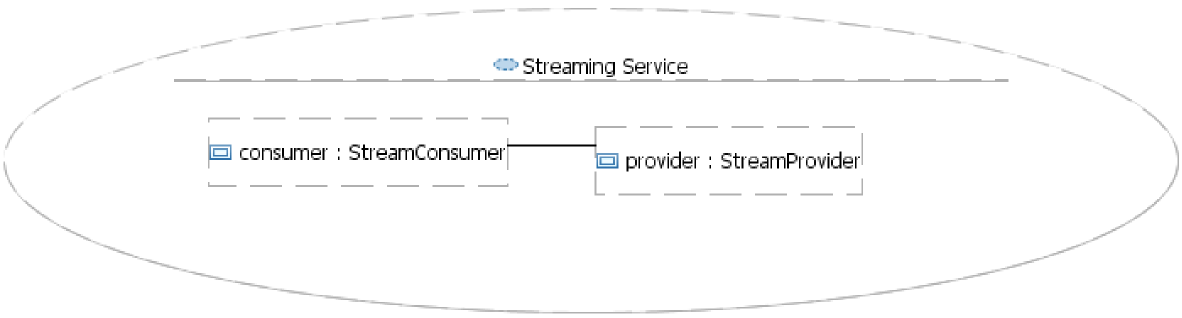


Figure 22: Example ServiceContract

Again, the stereotype <<ServiceContract>> should have been visualized at the top of the oval. This defines that there are two roles for the streaming service, one provider and one consumer. The interaction between these roles can then be further detailed using a UML behaviour such as a sequence diagram.

6.3. The services

In describing a service it is essential to describe the interface of the service and how this interface is to be used, the latter often described as the protocol of the service. SoaML uses a standard UML interface to describe the interface part, and adds a UML class with stereotype <<ServiceInterface>> that can have an attached description of how the services are used by consumers. Figure 23 shows a simplified description of the Transcoding service.

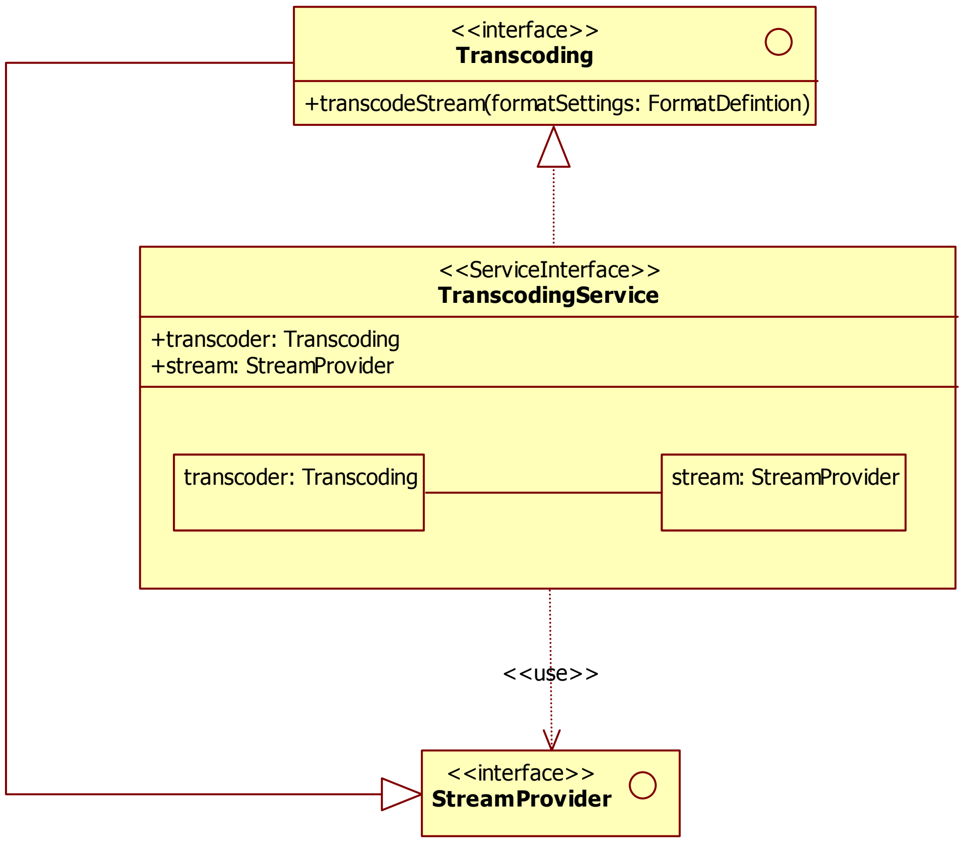


Figure 23: Simplified description of transcoding service

The model describes that the transcoding services provides the Transcoding interface (using the realization dependency link) and requires the StreamProvider interface. The standard UML mechanism for extension is used to denote that the transcoding interface also provides a stream as output. The lower compartment of the TranscodingService class shows the two parts that are needed to provide this service, the transcoding interface and the StreamProvider interface. In order to specify behaviour, or the protocol, between these a Collaboration can be added and described further using for instance a sequence diagram. A simple example of such is provided in Figure 24.

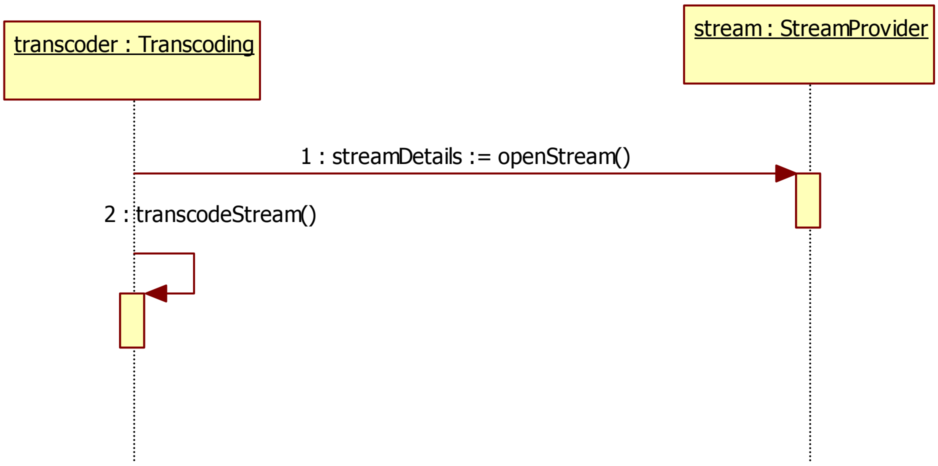


Figure 24: Interaction between transcoding and streaming

Services are provided by Participants, playing the roles defined by the services they provide. Participants expose their required and provided interfaces using ports typed by the relevant ServiceInterface classes. The provided services are offered through ports decorated with the <<ServicePoint>> stereotype, while the required are denoted with <<RequestPoint>>. Figure 25 shows the TranscodingProvider where the additional stereotype <<FlowPort>> has been added in order to describe that this port allows for stream based communications. This stereotype is taken from the MARTE UML profile (and is also part of the SysML profile).

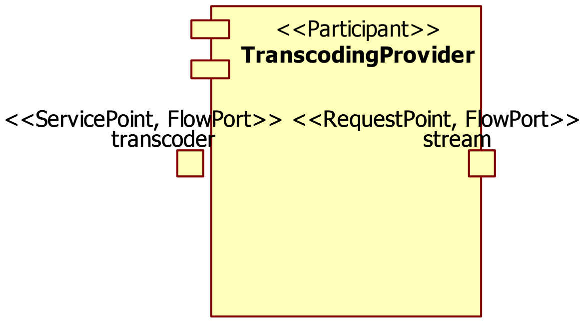


Figure 25: Description of the TranscodingProvider participant

The ports can then be connected using UML connectors, indicating a communication path between the components, an example of which is shown in Figure 26.

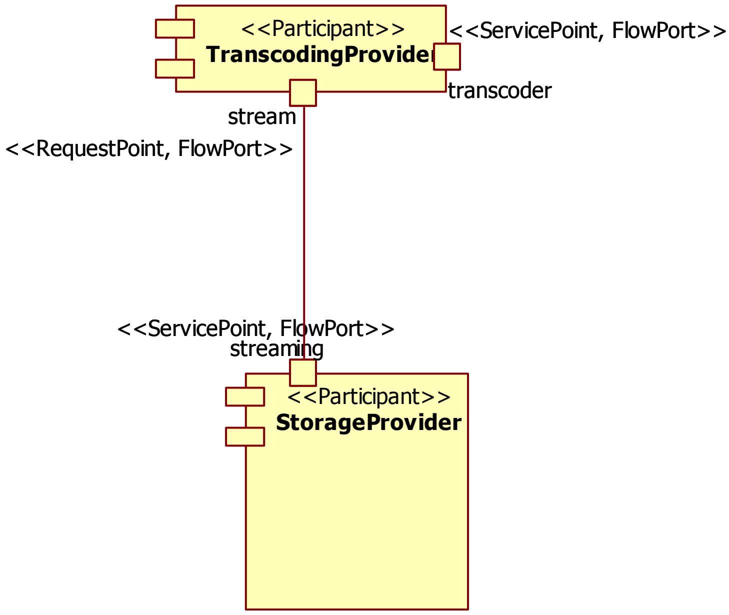


Figure 26: Example of communication path

6.4. Non-functional aspects

The SoaML profile it self does not provide anything directly related to the description of real-time or QoS aspects. Since multiple UML profiles can be applied to UML models such aspects can be added to the models using the MARTE or the UML Profile for Modeling Quality of Service and Fault Tolerance Characteristics and Mechanisms Specification. The SENSORIA project (IST-016004) has done work on how to apply non-functional properties to UML models of services based on the latter UML profile in [46]. The profile supports the definition of a contract between a requester and a provider and thus allows for definition of SLAs. An example of non functional properties for the StreamProvider and Transcoding interfaces is shown in Figure 27.

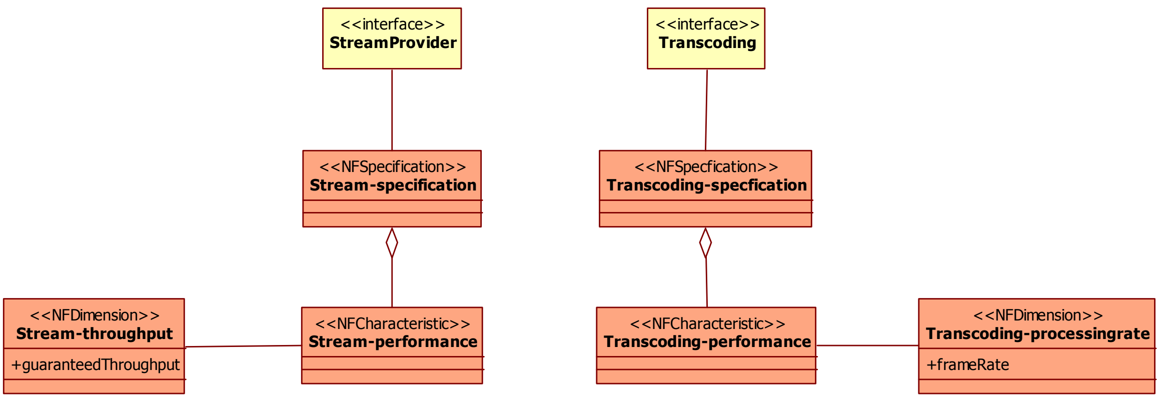


Figure 27: Non-functional properties for interfaces

The SENSORIA work also provides ways of defining contracts between services in a similar manner to the above, by defining the stereotype NFContract, that in turn has specifications characteristics and dimensions. In addition monitors can be modelled for the contracts. A simple contract between the transcoding service and the streaming service is provided in Figure 28.

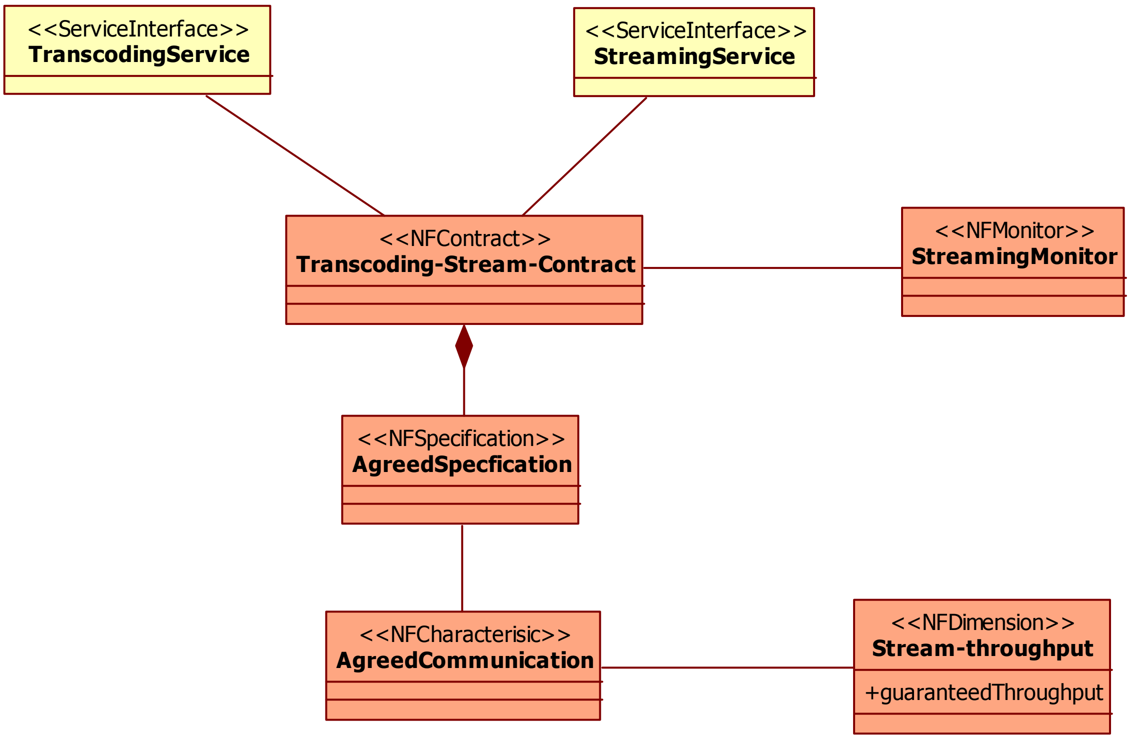


Figure 28: Example contract definition

In modelling services using SoaML the communication channels them selves are not really modelled. In order to describe requirements to the virtual networks in IRMOS some mechanism is needed for this. The above example adds communication related non-functional properties to the streaming service, but the communication channel it self is not defined. In the previous example diagrams UML connectors have been used to show connection paths between services, these could be annotated using the same mechanisms as described here.

6.5. Concept model outline

One of the goals of IRMOS is that services should be able to provide an agreed quality of service. This agreement is defined by a Service Level Agreement (SLA). In modelling of QoS using UML one has typically annotated the model it self with QoS values (including offers and agreements), in IRMOS many of these parameters will vary depending on what virtual hardware the service is deployed to and what virtual network is set up for the service. This in turn will be defined by SLAs and will therefore not be defined in a general model of the services. It is however necessary for the different services to define what QoS parameters they can handle, and how these can be monitored by the outside world. This is along the lines of the work of SENSORIA referenced in the previous chapter.

Figure 29 shows an outline of a model of the key concepts related to the IRMOS architecture, where Client Components and Service Components form an Application component that provides useful functionality to a user. Service Components have a SLA template that indicates what parameters can be part of an agreement of service. On the technical side a Service Component has a description of its interface(s) and a defined list of monitorable parameters. Such parameters are a representation of possibly application specific indicators of the performance of a Service Component, for instance frames per second for a transcoding service. During runtime these parameters can be used to monitor performance toward the defined SLA for the current execution. It is probable that most of the monitorable parameters will be part of the SLA template.

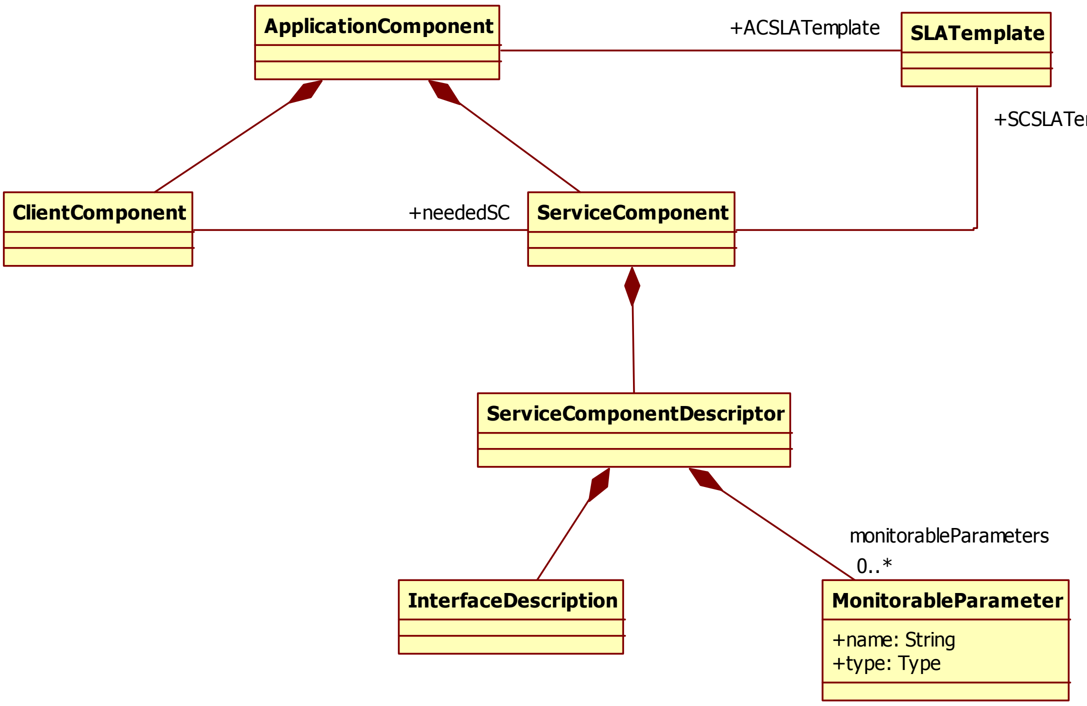


Figure 29: Intial outline of concept model

The reason for mentioning this here is that IRMOS will have to provide mechanisms (in form of a language) to enable service component developers to provide the Service Component Descriptors. This initial metamodel can be seen as a first step in the direction of an abstract syntax for this language. The metamodel will be refined and extended in subsequent WP5 deliverables.

6.6. Summary

Modelling Facet	Specific Facets addressed	Description
What is being modelled	MFWHAT10	The services, their interfaces and interconnections are described. In addition QoS attributes and contracts can be added to the models
Who will use this model	MFWHO4	Such models will be created by the service developer/designer
When will they use it	MFWHEN1	The model will be created during development of a service, but will be used in development of dependent services, or when composing services into applications.
Why is it useful		• The structural aspects of the services are useful when generating implementation and deployment artefacts. This information is also needed for subsequent composition of the services. • As the services are described using UML this connects well with the use of UML state machines for other aspects of the modelling.

Table 11. Modelling facets addressed by the UML model

Technique used	Why this technique?	Further information
SoaML	• It permits UML based technique for modelling different aspects of services • Provides enough information in order to generate of other artefacts such as code and parts of Service Component Descriptor	A.6.2.3
MARTE	Provides standardised ways of augmenting UML models with QoS information	A.6.2.1

Table 12. Techniques used for UML model

7. Case Study: execution time and resources

In the following sections we describe stochastic timed finite state automata that define the behaviour of the individual application components given in the case study. Some of the automata are described separately in the Annexes since those particular models do not provide further insight than the ones included here. The additional models can be found in Annex C.4 and C.5. The models are based only on the behavioural characteristics defined by the conceptual descriptions in the scenario. Even so, they can describe how completion time is sensitive to factors such as bandwidth or Mflops. The estimates produced from the models are mostly produced by analytical means; only two models use simulation techniques for some calculations. Overall the examples we consider show that:

- It is possible to produce meaningful behavioural models that can analytically estimate performance values from conceptual descriptions in many cases.
- The models can successfully be used to perform sensitivity analysis.
- The models can be successfully used to compare different levels of resource provisioning.
- The models allow experimentation with different composition of components without requiring any further benchmarking.

The performance and sensitivity results in this section were derived with the PRISM model checker, using the semantics defined in Annex C.2. Further details of the PRISM tool and stochastic timed automata can be found in Annexes C.3 and C.1.

7.1. Telecine Application

In this Section we describe stochastic timed finite state automata that model the ingestion process via a telecine device. Uploading onto the IRMOS platform involves an operator using a telecine device at the film studio to digitise the film. The most important part of the model is to describe the interdependencies between the various factors that influence performance so that variation in behaviour can be estimated. This section will focus on modelling these interdependencies of the case study for the telecine device and bandwidth.

Since this activity is the initial one in the case study we will assume the amount of film to be ingested in known exactly. Undoubtedly in a real application there would be several other known factors that would be able to interrupt or affect the telecine device and not just the bandwidth QoS. For any known interaction between factors that are suitably described it is possible to include these into the model. For this example in order to make the exposition concise we are only considering bandwidth variation. The telecine device is known to work at a constant rate, except when interrupted due to bandwidth dipping below a known QoS level.

Figure 30 shows a UML state machine defining a timed finite state automaton for the telecine device. This automaton is not in fact stochastic since the input data is assumed to be known exactly. Variation in behaviour for this automaton will occur through interaction with a second automaton that defines variation in the bandwidth QoS. The second automaton will model bandwidth variation stochastically. With the semantics defined in Annex C.2.2 It is necessary that the automaton is always able to perform some transitions that describe time passing. Otherwise when composed in parallel with a clock automaton the resulting system may deadlock. Hence, the first state is an idle state that simply marks time passing whilst waiting for a synchronous event `start_Telecine` that will trigger the automaton to start.

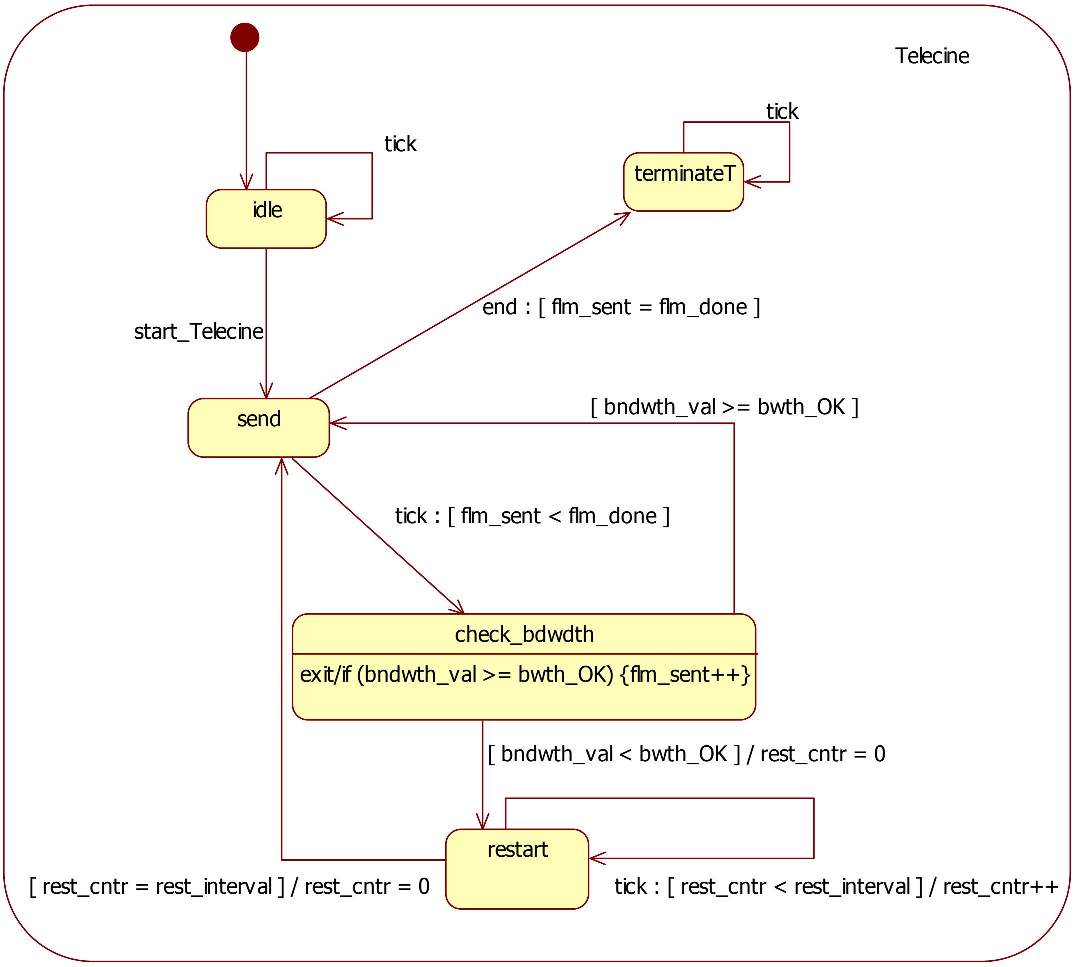


Figure 30: Stochastic Timed Finite State Automaton for Telecine Device

The rest of the automaton models transmission of ingested data at a constant rate when bandwidth QoS is within acceptable bounds. The `send` state abstracts the process of data being sent via some data link. It abstracts away all details of any output buffer or interface behaviour between the telecine and such a buffer. Since the model is solely concerned with the interaction of the telecine and bandwidth QoS none of these kinds of detail are required in the model.

From the `send` state the automaton transitions to the `check_bdwth` state if there is still some film to be ingested. Otherwise the Telecine automaton terminates by moving to the `terminateT` state (which is an abbreviation for terminate Telecine). The transition must occur synchronously with a clock tick event. That is the transition will be a timed event that occurs synchronously with a tick transition in a clock automaton. Variable flim_sent is used to track the amount of film sent so far. The constant film_done is the total amount of film to be transmitted.

In the model it is assumed that the bandwidth QoS is represented by a real number. Typically this would represent bytes sent per time unit. For example perhaps acceptable bandwidth is at least 100Mb/sec. It is also possible that the model combines bandwidth levels for the data link together with information about buffer size. For example, it may be from experience with a particular buffer size it is known that as long as the available bandwidth is at least 50Mb/s over a one minute period then the telecine device will work as expected. That would mean if the average bandwidth over a one minute period is at least 50Mb/s then the telecine device will work as expected.

In the telecine automaton we have assumed that the bandwidth level must be above some amount bwth_OK at each clock tick. That is, the average bandwidth between two consecutive clock ticks is above this amount. It is straightforward to modify the automaton so that the bandwidth level must be above bwth_OK for a longer continuous sequence of clock ticks before the next portion of film is transmitted. However, for the purpose of this example to do so would not give any particular additional insight into the modelling process.

In the check_bdwth state should bandwidth drop below the bwth_OK level then the telecine device has to perform a reset. The exact details of how that occurs are not important for the model, only how long that takes and the fact that no data is transmitted during that time. If bandwidth is above the bwth_OK level then the automaton models the successful transmission of data by increasing flim_sent and transitioning back to the send state.

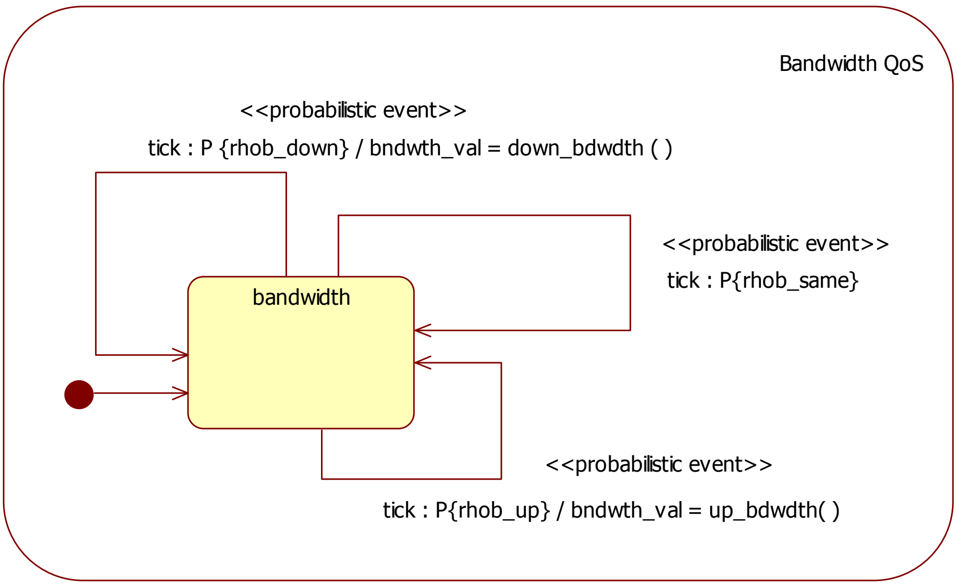


Figure 31: Bandwidth Stochastic Timed Finite State Automaton

With no knowledge of the mechanics of the data link or of data traffic it is only possible to model bandwidth variation as a stochastic automaton where the amount of variation is constrained to be within QoS guarantees. For the IRMOS platform that is appropriate since the exact nature of possible variation within such guarantees will be known. Figure 31 describes a UML state machine defining a very elementary automaton that describes bandwidth variation in terms of transitions whose effect is to increase or decrease the amount of bandwidth. For the telecine example we define a bandwidth variation model where the value can change incrementally from one clock tick to the next. In later examples we will look at more extreme variation where bandwidth can go from acceptable to unacceptable in a single clock tick and vice-versa.

As mentioned previously UML state machines do not provide stochastic transitions. To depict these within a UML state machine we annotate transitions with the <<probabilistic event>> stereotype. This stereotype has the Markov decision process (MDP) semantics defined in Annex C.2.2 The bandwidth automaton contains only a single state. Each transition is chosen stochastically. The sum of probabilities for all three transitions is one, so that the probabilities represent a distribution ensuring that some transition will always occur. In the automaton there are functions `down_bdwdth()` and `up_bdwdth()` that define the variation in the bandwidth for particular transitions. These functions are defined in such a way that they only change bandwidth if the result will be within maximum and minimum bounds. The other transition, if it occurs, leaves the bandwidth level unchanged. Thus at each clock tick bandwidth may go up, down or stay the same and which occurs is determined purely stochastically.

The probability that the bandwidth will increase is defined to be `rhob_up`. The probability that the bandwidth will decrease is defined to be `rhob_down`. The probability that the bandwidth will be unchanged is defined to be `rhob_same`. For this example we choose an arbitrary relationship between the three probabilities given by

```
rhob_down = 2*rhob_same, rhob_same = (1 - rhob_up)/3
```

For the example we assume that the bandwidth QoS is a real number between 0 and 10. That is we rescale the bandwidth parameter for average bandwidth per time interval so that it lies between 0 and 10. Thus a required QoS of 9 is interpreted as meaning that the bandwidth must be above 90% of the maximum permitted bandwidth for each time unit.

In order to analytically calculate performance estimates from the above automaton a clock is required of the form defined in Section C.3.1.1.

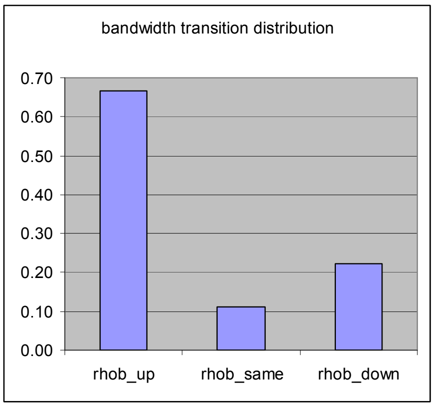


Figure 32: Example transition distribution for bandwidth

As a first calculation it is informative to see how completion time varies when the time taken to reset the telecine varies. In this calculation we take `rhob_up` = 2/3 and consider only the first twenty minutes of material to be ingested. There is a high probability at each time unit that bandwidth will increase or stay the same. We use this initial amount in order to highlight the non-linear increase in estimated completion time. This gives the distribution shown in Figure 32.

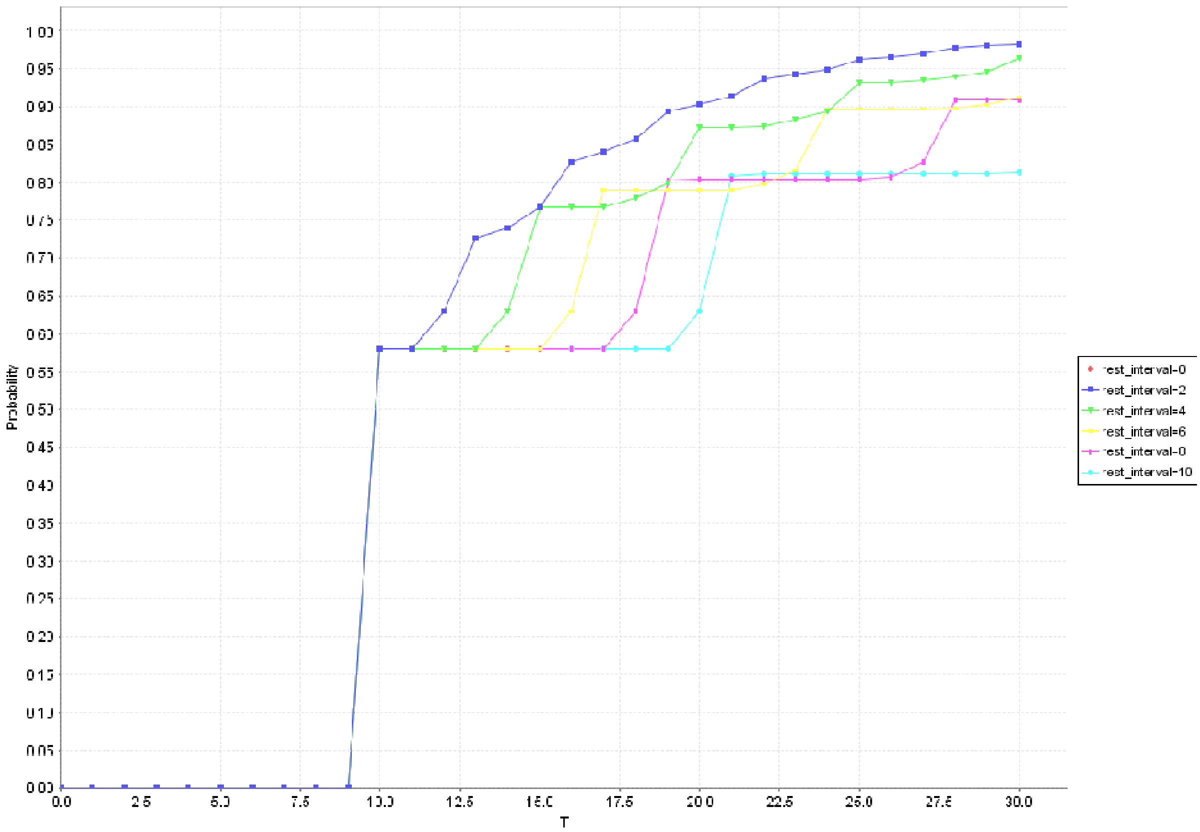


Figure 33: Estimated ingestion for first twenty minutes of video by time T

Figure 33 shows the estimated completion by time T of the telecine device for different reset values. From the data it is very clear that the completion time estimates have sudden discontinuous step changes. This illustrates how an automata based model of behaviour can give a more representative description of performance.

For the next set of calculations we keep the reset interval fixed at ten time units. Figure 34 shows the estimated execution time where the required QoS value and the probability for `rhob_up` vary. In this run of the model we have assumed the amount of material to be ingested is 180 minutes. That can be ingested in 90 minutes if the telecine device can run at full speed. The time interval for this model was taken to be the reset value, which in this case is assumed to be ten minutes. The time axis for Figure 34 therefore represents tens of minutes.

The probability shown is the probability that termination will occur by time T. From the graph we can see that both the level of QoS required and the likelihood of bandwidth increasing are significant in affecting the execution time. The graph shows quite a large sensitivity to both of these variables. In particular this shows that the termination time will vary greatly with respect to the behaviour of the bandwidth. What is also noteworthy is that the various curves cross over each other. This shows that elementary comparison between parameter sets is not simple.

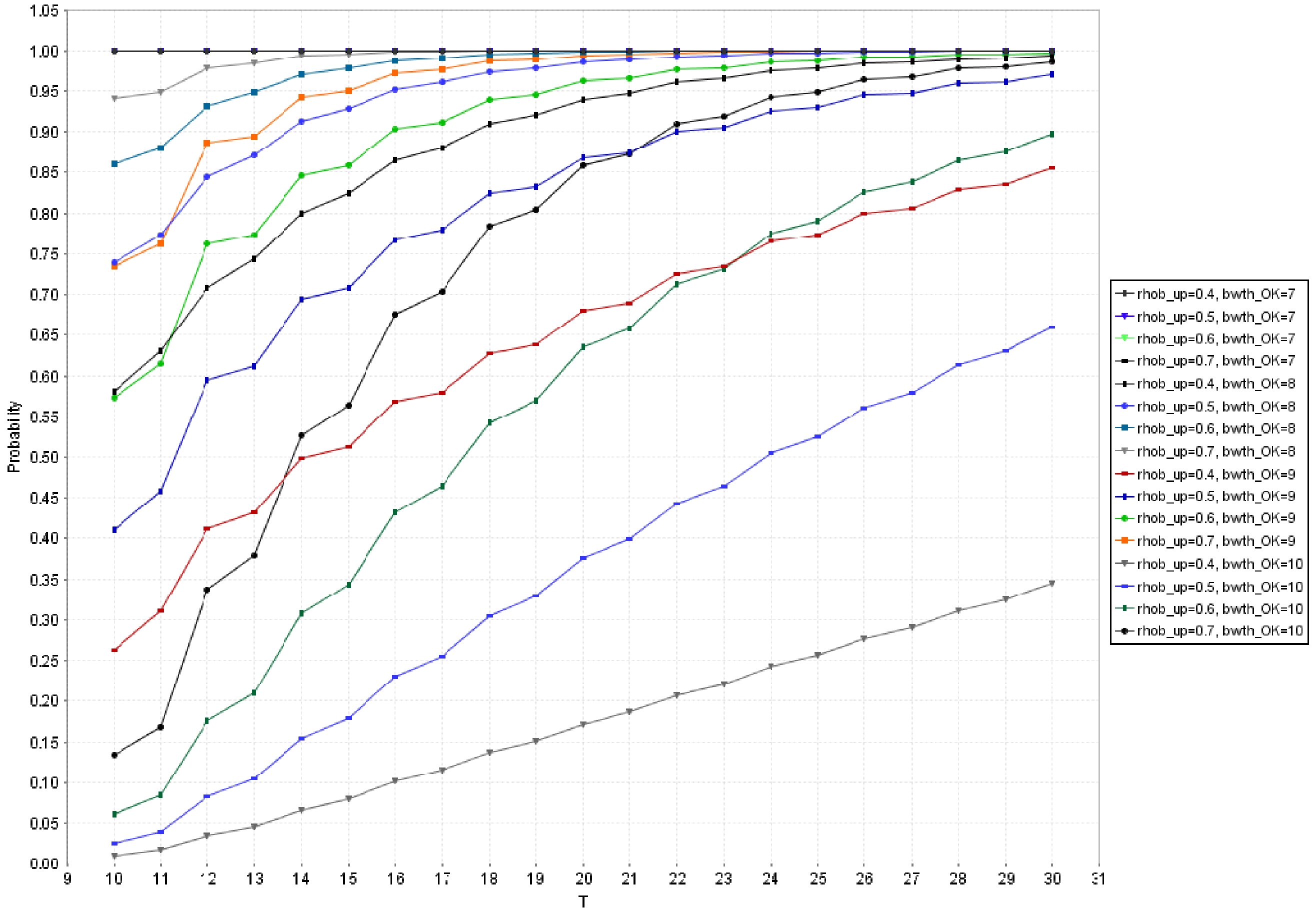


Figure 34: Estimated completion by time T, where T is in tens of minutes.

The model used for this example has assumed that the available amount of bandwidth varies stochastically. This assumption may not be correct. What if the telecine device transmitting data causes traffic management problems that cause a reduction in the bandwidth available? Then the affect of the telecine device entering a reset period should also have a direct effect on the level of bandwidth available, since the traffic generated by the telecine device is no longer present. It is possible to modify the above automata to represent an approximation to such behaviour. The new automaton causes the `bndwth_val` to have a higher chance of increasing with each clock tick while in state `restart`. Figure 35 shows the difference in termination time with and without this interaction between reset and bandwidth level. The probability shown is the probability that termination will occur by time T. The red curve shows the estimated termination time when a reset significantly reduces traffic. The blue curve shows the estimated termination when this is not a significant effect.

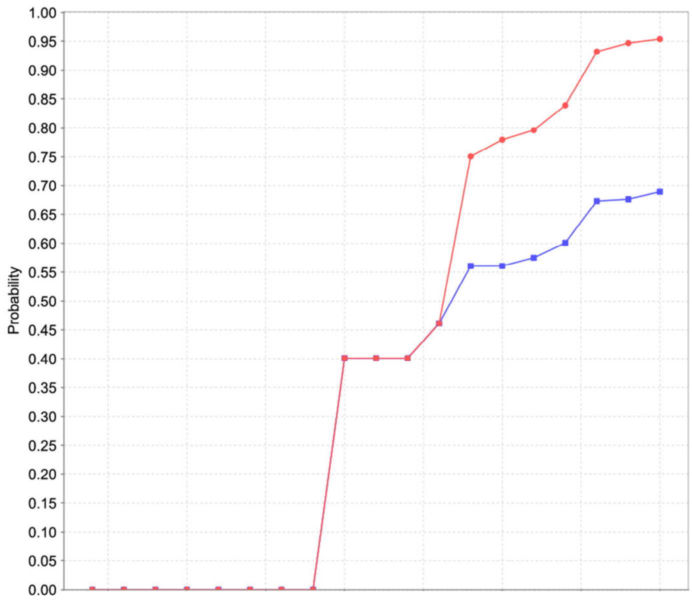


Figure 35: Completion by time T when reset affects available bandwidth

This refinement of the model demonstrates the importance of interactions between the different components being modelled. The initial assumption that available bandwidth is independent of the telecine device transmitting data has significant effects on the termination time. The first model is conservative in estimating termination and therefore represents a safe estimate where there is no other information about possible interactions.

This example has shown how sensitive completion time for a simple application component such as a telecine device can be with respect to bandwidth variation. The analytical calculations also highlighted how discontinuous jumps in completion time estimates can occur. Finally the model illustrated how different parameter value sets do not have a simple relationship with one another. This was illustrated in the graphs where completion time curves for different parameter sets cross.

7.2. Dailies Review

In the scenario the film will be reviewed by the director in conjunction with two post houses. Based on prior knowledge it is known that only between 20% and 30% of the film material available will be reviewed. Assuming input film of length three hours that means a minimum of 36 minutes will be reviewed and a maximum of 54 minutes. The probability of between 37 minutes and 54 minutes of material needing reviewing is known to be 10% and the probability of 36 minutes is 90%. We assume these probabilities are defined by a negative exponential distribution.

A stochastic timed automaton that abstracts the review process is shown in Figure 36. In this case we suppose the data being reviewed is stochastic. That is modelled by a time varying probability on transitions. The probability of completion increases each clock tick, where the increase is given by the equations defined in SectionC.3.1.1. Recall that the notation for a probabilistic transition it is possible to include a Boolean guard in addition to a probability. The transition can only occur when that Boolean guard evaluates to true. Then the transition can occur with probability given by the probabilistic value annotating the transition. The Boolean guard is enclosed by square brackets on the transition.

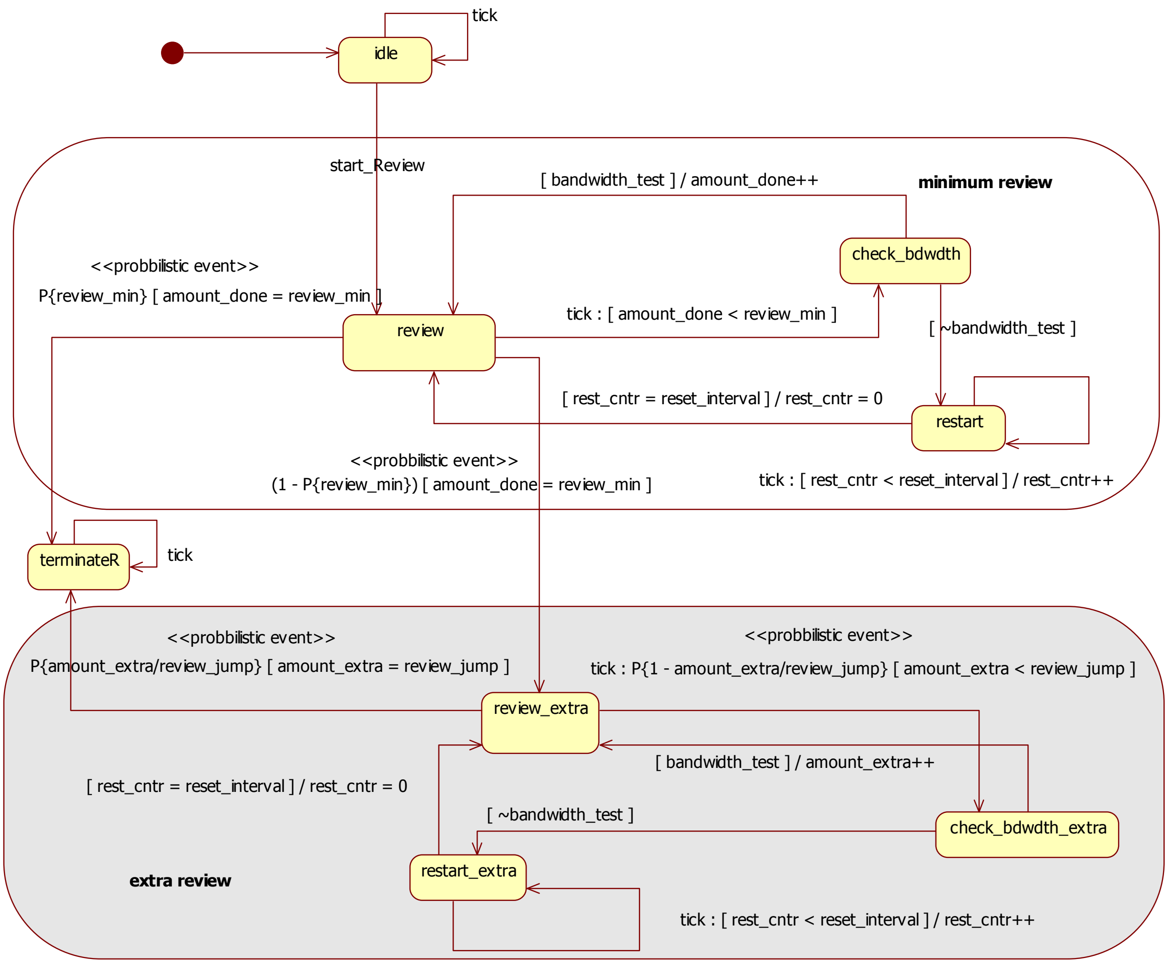


Figure 36: Director review of dailies

For this example we are interested in the affect of bandwidth and amount of review film on the completion time of the review task. The automaton uses a variable `amount_done` to store the amount of material reviewed at each clock tick. `review_jump` is a constant that defines the extra amount of material that might have to be reviewed. `reset_interval` defines the number of clock ticks that a stream reset will require if the video stream has to be rewound. `amount_extra` is a variable used to record the additional amount of material reviewed beyond the minimum 36 minutes. `bandwidth_test` is a boolean valued test that is true when the bandwidth level over the last time unit is above the minimum QoS level required for all three sites involved in the review process.

Once the minimum amount of material is reviewed there is a probability of `review_min` that the review will terminate. Otherwise additional material will be reviewed with a gradually increasing probability at each clock tick that the review process has terminated.

As in the telecine example, if the level of bandwidth for any of the sites viewing the dailies falls too low then a reset will have to occur. That will involve rewinding the data stream a fixed amount and retransmitting the data again. If any one of the sites suffers a reset then all of them will since they are viewing the material synchronously.

In the automaton before a state change can occur that represents an increase in the amount of material being reviewed, first the automaton transitions to the `check_bdwdth` state. As in the telecine example, this state abstracts the concept of checking the bandwidth level over the last time period to see if a reset has been triggered at any of the reviewing sites.

The automaton in Figure 36 describes the review process for the combination of the director and the two production sites. We can therefore represent the bandwidth variation of the three streaming data links for the reviewers as a single automaton, as shown in Figure 37.

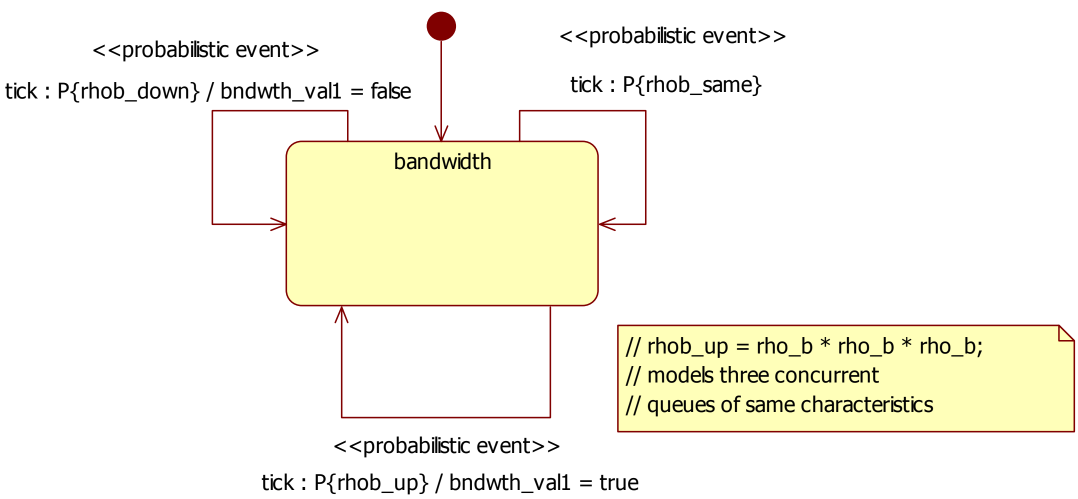


Figure 37: Automaton combining bandwidth variation for three links

This automaton uses a Boolean value to represent whether the amount of bandwidth is above the specified QoS level during one time unit. This defines a much more compact abstraction for bandwidth variation than the previous model used for the telecine device. In this example we have abstracted away from representing the level of bandwidth available. This model can better represent wildly fluctuating bandwidth, which therefore has a greater potential to model bursty traffic. In the automaton we assume each data link between the reviewing sites have the same characteristics. Let the probability that over one time unit the bandwidth level is sufficient for each site be `rho_b`, then the probability that the bandwidth level is sufficient for all three sites is simply (`rho_b`)^3. We denote this probability as `rhob_up`.

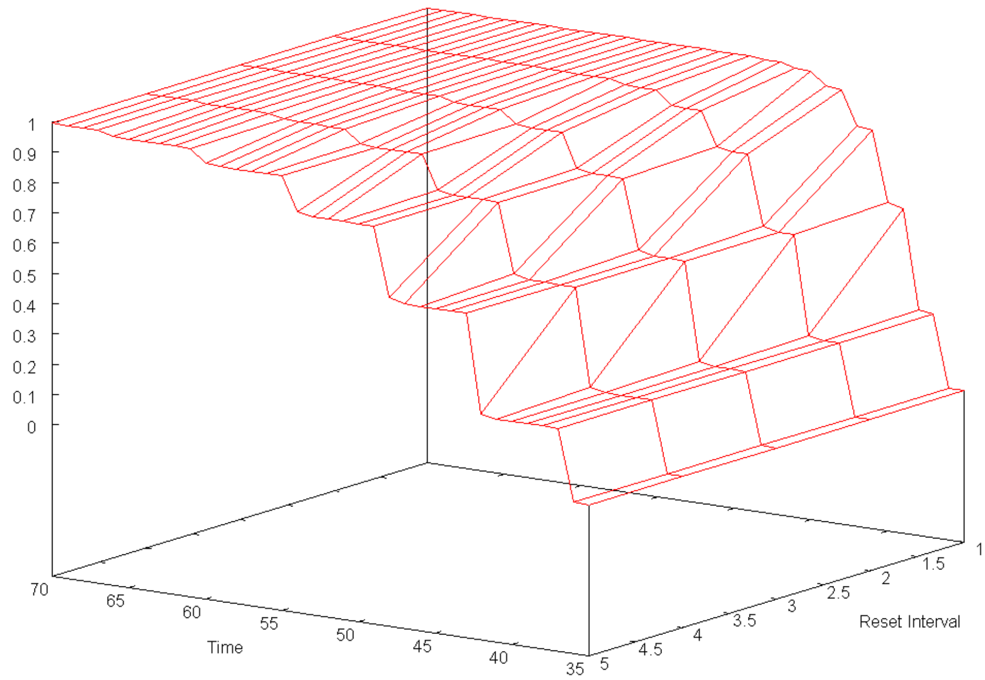


Figure 38: estimated review time for various reset intervals

Figure 38 gives estimated completion time for the review process that are analytically derived from the automata described above when run concurrently with a discrete clock. In these calculations the value of `rho_b` is 98%. That is for 98% of the time each data link will have sufficient bandwidth. The probability level shown is the probability that termination will occur by time T. This is why the probability increase towards one as time increases. The graph is shown as a two dimensional surface to illustrate how sharply a change in reset interval affects the probability of a given termination time.

The graph illustrates the variation in completion time as the reset interval increases from one time unit to five time units. In this example a time unit corresponds to one minute. When the reset interval is only one minute then there is virtual certainty that the review process will complete in 48 minutes. The same data is shown in Figure 39 as a collection of curves. This illustrates better how termination time grows with increase in reset value.

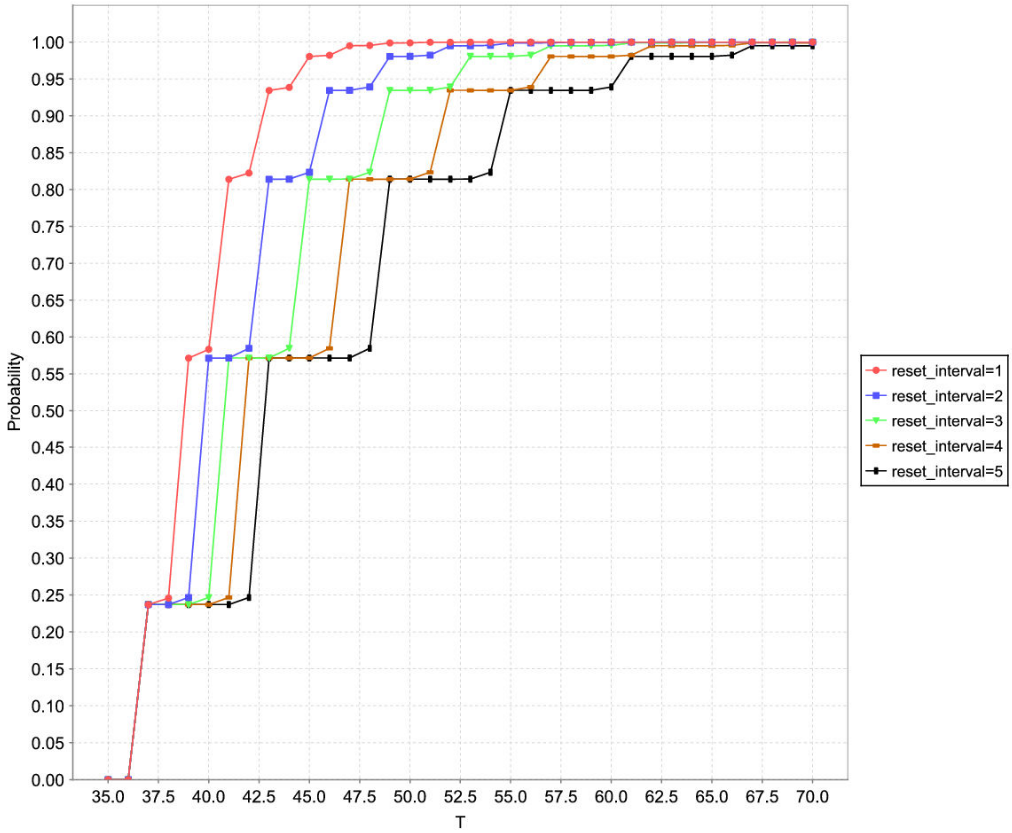


Figure 39: Review reset variation

For example when the reset value is one there is a 95% probability of finishing in 42 minutes. Whereas, when the reset value is two there is a 95% probability of finishing in 46 minutes. When the reset value is 5 there is only a 95% probability of finishing after 55 minutes.

At first glance the worst case analysis for this example would seem to suggest that review must surely continue way beyond the 54 minute value. That appeals to common sense since it seems as if the review will end up looking at much more than the minimum amount 36 minutes of material and it there is bound to be some resets resulting from low bandwidth in that period. However, incorporating known stochastic distributions of review material and bandwidth levels shows this is very excessive in most cases. The figures show that in cases where a high level of confidence in completion is sufficient, rather than absolute certainty, then completion time is very likely to be much lower than this worst case suggests.

The dailies example illustrated how the bandwidth model can be abstracted to a very high level and still gives valuable insight into the completion time for the review process. This model again illustrates the discontinuous jumps in completion time that are caused by bandwidth QoS changes. The example also illustrates how a behavioural model can describe three data link bandwidth levels within a single automaton. The automaton for the review process also shows how a complex distribution of input data can be dealt with by dividing up the automata into the composition of two parts. This compositional aspect of modelling will be further considered in the colour correction section.

7.3. Audio Correction

All of the examples so far have focused on interactions between the data links and application execution time. This allowed general attributes of the models to be described without concern for the other factors that also affect performance. This section now considers how parameters other than just bandwidth affect estimates. For example, how parameters such as audio quality and available Mflops affect performance.

This Section describes a detailed model of the Audio Correction activity of the film post-production case study. The activity is analysed in isolation of the other activities in the case study. The analysis results, e.g. the activity completion time distribution function, are to be used in setting up the activity parameters for the application, e.g. likelihood of timely completion of the activity given a pre-determined deadline.

Working scenario

A post-production house performs audio correction on some selected sequences of shots. The number of shots needing audio correction cannot be determined in advance (it is decided as part of another activity), however it is anticipated that audio correction and effects will be applied to approximately 30 minutes of footage +/- 10 minutes.

It is needed to listen to the audio at least 3 times and as much as 6 times when developing and applying audio correction settings. This is done by downloading the already digitised soundtrack rather than streaming it, i.e. downloading can be at any rate so the bandwidth requirement is flexible. The soundtrack is 10% of the data volume of the digitised film footage.

Scenario analysis and further assumptions

The audio correction activity consists of three steps: download the digitised footage audio, listen to the audio in order to determine the settings for the audio correction process and finally process the audio in order to apply the necessary corrections and effects.

In this case study we are interested in the time that it takes to complete the audio correction activity. Because the time length of the audio (T_{Audio}) to be processed is not known in advance, but for simplicity we assume it to be uniformly distributed between a lower value (T^l_{Audio}) and an upper value (T^u_{Audio}) respectively of 20 minutes to 40 minutes, the completion time of each of the above activity steps will be characterised by a probability distribution and the sum of these completion times will be the overall activity completion time, which will be characterised by another probability distribution. We assume uniform distribution for T_{Audio} for simplification of the calculations; actually any distribution can be approximated to an arbitrary accuracy by an appropriate number of scaled uniform distributions but this will introduce complexity unnecessary at present.

To model the audio downloading time we need to know the volume of the audio in terms of memory and the bandwidth of the transfer link used. Because the audio is downloaded and not streamed in real-time only the average link bandwidth (S_{link}) is important and we assume it to be 50 Mb/sec (megabits per second). The audio volume is assumed to be 10% of the total volume of the digitised film footage, which is the sum of the volume of the video and the volume of the audio, so the following formula gives the relation between the video volume (V_{Video}) and the audio volume (V_{Audio}): $V_{Audio} = V_{Video}*(0.1/0.9)$. **(5.3.1-1)**

The volume of the video can be calculated from the frame rate (*rate*), the colour depth (*depth*), frame resolution (*resolution*) and video duration (*duration*) according to the following formula: $V_{Video} = rate*depth*resolution*duration$. **(5.3.1-2)**

We assume the following values for these attributes: video frame rate of 24frames/sec, colour depth of 16 bit, frame resolution of 2 MPixels and video time duration the same as the duration of the audio.

The volume of the audio, V_{Audio}, will be uniformly distributed between a lower value (V^l_{Audio}) and an upper value (V^u_{Audio}), calculated according to **(5.3.1-1)** and **(5.3.1-2)** where the value for the time duration is respectively T^l_{Audio} and T^u_{Audio} of the audio duration distribution. The time for downloading the audio (T_d) will be calculated as: $T_d = V_{Audio}/S_{link}$ (5.3.1-3)

The downloading time of the audio will be uniformly distributed between a lower value (T^l_d) and an upper value (T^u_d), calculated according to (5.3.1-3) where the value for the audio volume is respectively V^l_{Audio} and V^u_{Audio}. This distribution will be used in the state model for the audio downloading to calculate the state transition probabilities.

For the audio listening model, the total listening time is the joint probability distribution of the number of listening iterations (N_l) and the listening time per iteration, T_{Audio}. In the current scenario N_l is uniformly distributed between 3 and 6, and T_{Audio} is as described above uniformly distributed between 20 and 40 minutes. These distributions will determine the probabilities of the state transitions of the audio listening model.

The final step of the audio correction activity is the processing of the audio in order to apply the necessary corrections and effects identified during the audio listening step. To model the processing time (T_p) we need to know the amount of processing work (W_p) that needs to be performed and the speed of the processing resource (S_r). The processing work that needs to be done when applying audio corrections is assumed to be calculated from the frame rate (*rate*), frame resolution (*resolution*) and video duration (*duration*) according to the following formula:

$$W_p = algorithm\_factor * rate * duration * resolution. \quad (5.3.1-4)$$

For audio correction the *algorithm_factor* is assumed to have the value of 2.5. The speed of the processing resource, S_r,is assumed to be 100 MFLOPS/sec and the processing time, T_p,is calculated as: s$\boldsymbol{T_p = W_p / S_r}$

(5.3.1-5)

The audio processing step will be modelled by a state machine with transition probabilities determined by the attributes Processing work and the Resource speed.

As *duration* in (4) is uniformly distributed, so is W_p and equally T_p will be uniformly distributed with lower and upper values calculated according to **(5.3.1-4)** and (5.3.1-5). The T_p distribution will be used in the state model for the audio processing to calculate the state transition probabilities.

The full audio correction process is modelled as sequence of three state machines. The next section describes the audio correction model.

Activity model

The audio correction activity model consists of four state machines. Figure 40 depicts the model in the form of UML state machine diagrams. The activity attends sequentially four high level states: *TransferAudio*, *InitiateListening*, *AudioListening* and *AudioProcessing*. At a lower level these higher level states are modelled by (i.e. encapsulate) the four activity state machines. Similarly to all the other activity models, there is a time clock state machine to model the physical time, which executes in parallel with the activity state machines. The time clock state machine is not included in the diagram as the only event we are interested in is the *tick* event which signals the end of each elapsed time unit. Each model state machine is synchronised with the *tick* event of the time state machine in the following way:

- state transitions which are associated with processing over time of same sort, e.g. data transfer, audio listening and audio processing, must be synchronised with the *tick* time clock event.
- instantaneous state transitions, which are associated with control events or decision events, e.g. the event of finishing a download or the decision on how many times to listen to the audio, are not synchronised with the *tick* time clock event.

All the activity state machine models have *tick* loopback transitions for the initial and final states which enable the clock to advance and not to deadlock.

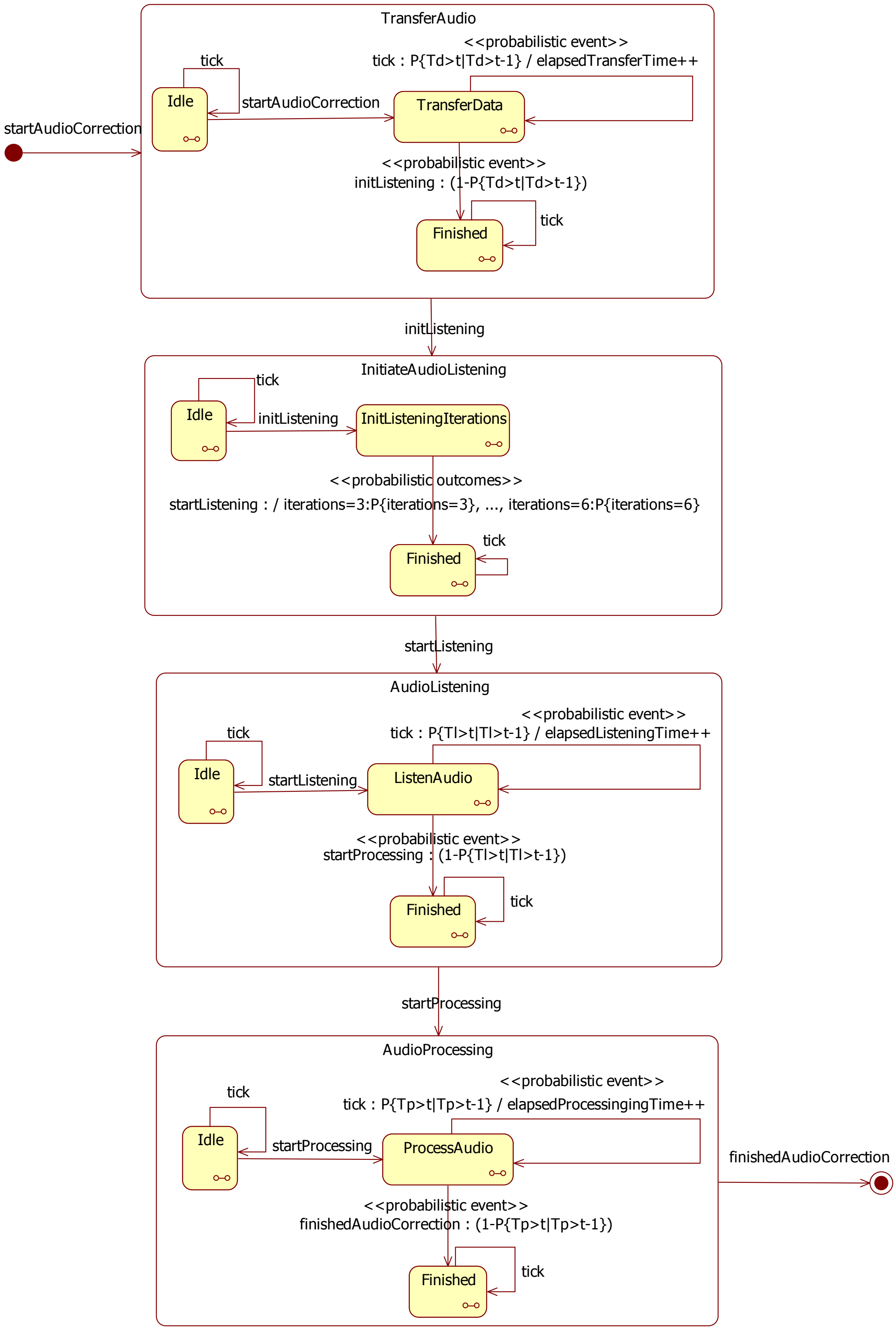


Figure 40: Audio correction activity model.

The first state machine, the submachine for the *TransferAudio* top level activity state, models the audio download. The non-trivial transitions are the *tick* event transition from state *TransferData* to itself and the *initListening* event transition from state *TransferData* to state *Finished*. The first transition models the data download over time and the second models the event of finishing the data download and starting the audio listening step. These are probabilistic transitions with probabilities being a function of the elapsed download time. The probability of the *tick* transition at time step *t* is the probability that we need more time to download all the data given we have been downloading data so far. We denote this probability as $P\{T_d > t| T_d > t\text{-}1\}$. The probability of *initListening* transition is $1\text{-} P\{T_d > t| T_d > t\text{-}1\}$.

The second activity state machine, the submachine for the *InitiateAudioListening* high level activity state, models the probabilistic choice of the number of listening iterations. Transition *startListening* (which occurs with probability 1) has a number of probabilistic outcomes with equal probabilities that sum to 1. The outcomes are the number of listening iterations necessary to optimise the settings for the audio processing step. This transition is not synchronised with the clock as it is treated as an instantaneous decision transition and directly activates the state machine for the audio listening step.

The third activity state machine, the submachine for the *AudioListening* high level activity state, models the process of audio listening during optimisation of the audio correction processing settings. The non-trivial transitions here are the *tick* event transition from state *ListenAudio* to itself and the *startProcessing* event transition from state *ListenAudio* to state *Finished*. The first transition models the audio listening over time and the second models the event of finishing listening and starting the audio processing. Again, these are probabilistic transitions with probabilities being a function of the elapsed listening time. The probability of the *tick* transition at time step *t* is the probability that there is more time to listen to the whole audio given we have been listening so far. We denote this probability as $P\{T_l > t| T_l > t\text{-}1\}$. The probability of the *startProcessing* transition is $1\text{-} P\{T_l > t| T_l > t\text{-}1\}$.

The forth and final activity state machine, the submachine for the *AudioProcessing* high level activity state, models the audio processing stage. The non-trivial transitions here are the *tick* event transition from state *ProcessAudio* to itself and the *finishedAudioCorrection* event transition from state *ProcessAudio* to state *Finished*. The fists transition models the audio processing over time and the second models the event of finishing the audio processing. Same as with the other activity models, these are probabilistic transitions with probabilities being a function of the elapsed listening time. The probability of the *tick* transition at time step *t* is the probability that there is more processing to be done given the audio has being processed so far. We denote this probability as $P\{T_p > t| T_p > t\text{-}1\}$. The probability of the *finishedAudioCorrection* transition is $1\text{-} P\{T_p > t| T_p > t\text{-}1\}$.

The UML diagrams can be directly mapped to the PRISM modelling language, as explained in Annex C.2.2. We have implemented this activity model in the PRISM tool and obtained the activity completion time probability distribution as described in the following section.

Numerical Experiments

We have implemented the model described in the previous section in the PRISM modelling tool and performed a number of experiments to analyse the activity completion time probability distribution. The numerical values used for the experiment are as stated in the previous sections:

- audio time length: uniformly distributed between 20 and 40 minutes.
- video frame rate: 24 frames/sec, colour depth: 16 bit, frame resolution: 2 MPixels.
- average link bandwidth: 50 Mb/sec.
- listening iterations: uniformly distributed between 3 and 6.
- audio volume: 10% of the whole digitised film.
- audio processing algorithm performance factor: 2.5.
- speed of the processing resource: 100 MFLOPS/sec.

Figure 41 shows the PDF of the completion times of the individual stages comprising the audio correction activity. We can immediately see that the longest and the most uncertain stage is the audio listening and settings optimisation stage. If minimising the overall audio correction activity completion time and uncertainty is needed this is the stage which will need the most attention.

The experiment is performed in time steps of 10 minutes in order to constrain the model state space and the overall execution time. Because of this high granularity some of the values shown by the graphs might be less accurate. For example the audio processing completion time (the green graph) should show a uniform distribution with lower and upper values respectively 2.4 and 4.8, but these values fall in between the time step values and so the graph is not completely accurate. Nevertheless, the graph displays the general trend and if more accurate graph is needed the experiment can be redone with smaller time step.

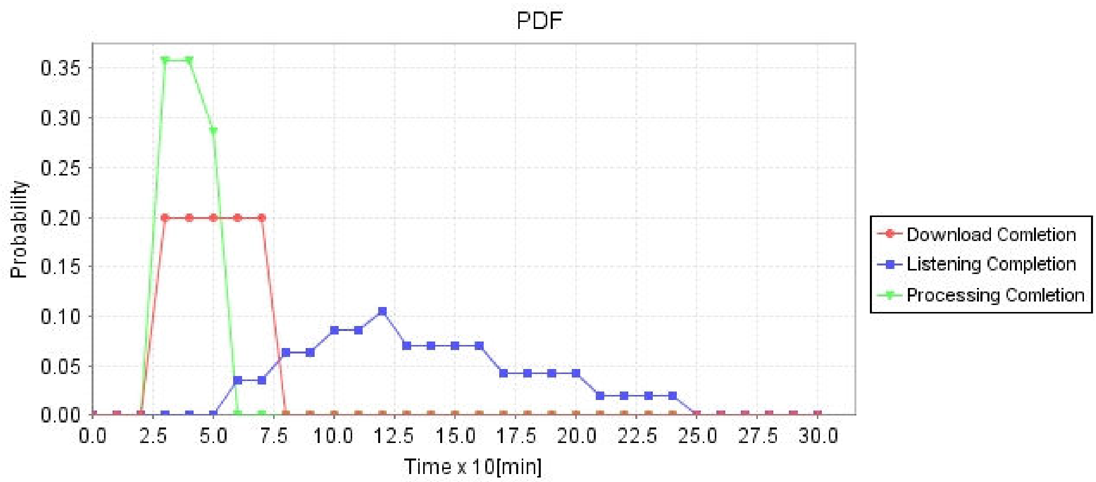


Figure 41: Individual audio correction stages completion times PDF.

Figure 42 and Figure 43 show respectively the PDF and the CDF of the overall audio correction completion time. The PDF shows that the most likely completion time is about 200 minutes. However, the spread of the distribution is quite large, i.e. the uncertainty in the completion time is high; 90% of the distribution is roughly between 150 and 310 minutes.

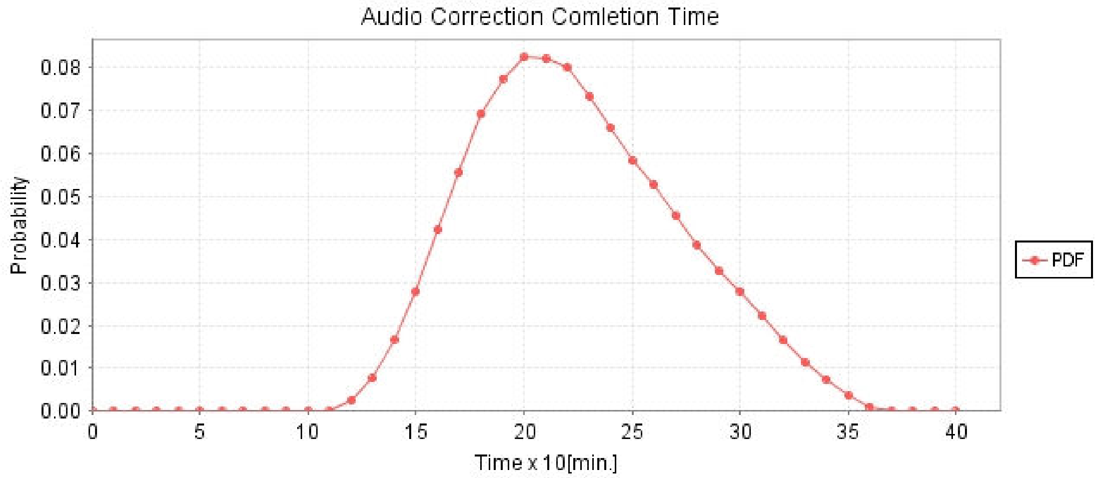


Figure 42: Overall audio correction activity completion time PDF.

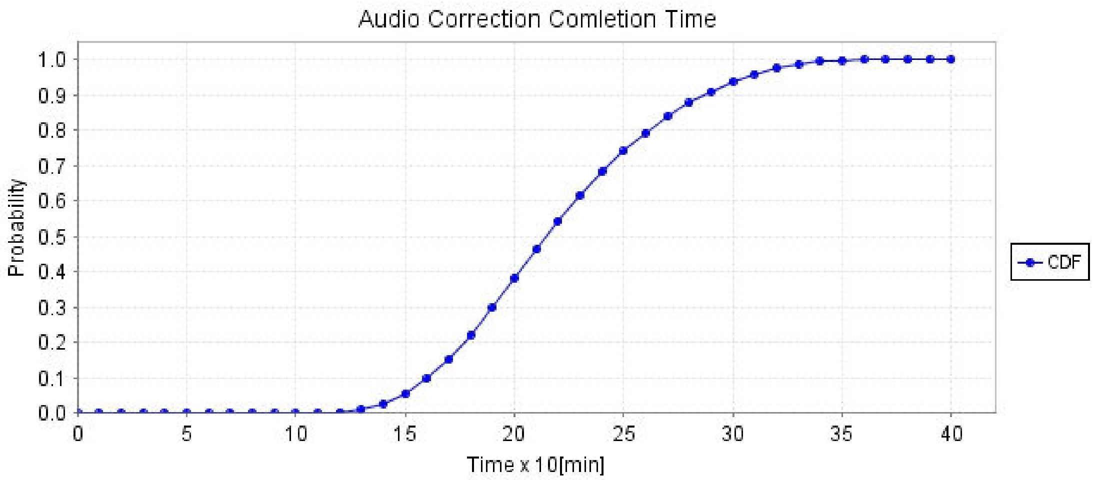


Figure 43: Overall audio correction activity completion time CDF.

The overall activity CDF from Figure 43 can be used to set up the audio correction activity parameters when modelling the whole film production case study, e.g. for a specified deadline, from this CDF we can work out the probability of completing the activity before the deadline without compromising the quality of the activity product.

For this relatively simple scenario the numerical results might seem obvious but they clearly demonstrate the value of modelling. The CDF and PDF graphs for the overall activity directly facilitate risk analysis, e.g. the likelihood of failing to meet a deadline. For example if we have a deadline of 4 hours to do the audio correction we have about 70% chance that we will meet this deadline without compromising the quality of the audio correction. From the PDF graphs for the individual activity stages we can identify the most time consuming and uncertain activity stage and concentrate our resources to improve this stage, e.g. instead of spending money on a network with higher bandwidth or on faster processing resource we might decide on assigning the audio listening and settings optimisation stage to the most experienced specialist that will need less listening iterations.

7.4. Summary

Modelling Facet	Specific Facets addressed	Description
What is being modelled	MFWHAT2 to MFWHAT9	Modelling includes: • How long an activity is likely to take given a particular level of resource available for that activity. • Sensitivity of completion time of an activity to variations in the resource made available to the activity. • Which aspects of an activity have the most influence on completion time.
Who will use this model	MFWHO1 to MFWHO7	• Clients can understand what performance they will likely need and the consequences if it is not achieved. • Application service providers can investigate different provisioning strategies. • Clients/Providers can identify the most critical aspects of provisioning and ensure they have priority.
When will they use it	MFWHEN1, MFWHEN2	• When clients formulate performance requirements for applications. • When clients choose between application service providers. • When application service providers choose levels of service from a resource provider, e.g. ISONI. • At execution time to identify what aspect of application usage to change if resource available is not sufficient, e.g. stream video to less people or reduce resolution.
Why is it useful	MFWHY1 to MFWHY6	• Allows a cost/risk analysis to be done • Avoids expensive overprovisioning

Table 13. Modelling facets addressed by the execution time and resources model.

Tool/Technique used	Why this technique?	Further information
Stochastic models	• Uncertainty can be included in the model. • Detailed knowledge is not needed on how uncertainty is generated (e.g. how a network behaves) only how the uncertainty is distributed. .	C.3, C.4, C.5
PRISM tool	• Freely available tool that can be used to execute models employing finite state automata	C.2.2

Table 14. Techniques used in the execution time and resources model.

8. Conclusion

This report has shown how models can be constructed to describe real-time applications. The techniques used are varied and include the use of stochastic process algebras, finite state automata, workflow models and specification languages to name but a few.

Particular attention has been given to modelling uncertainty surrounding real-time applications. This includes how variation in both the inputs to an application (e.g. data and parameter settings) and performance from the resources used to execute the application can each impact on the probability of the application executing successfully. The techniques used have been demonstrated using a specific application scenario, which has been invaluable when evaluating different approaches and furthermore provides valuable insight into the level of detail needed when developing meaningful models.

Overall, modelling is a complex and difficult business for real-time applications on service oriented infrastructures, in particular for extended value chains where different organisations fulfil roles of application design, development, hosting, consumption and infrastructure provision. There is no single technique that can adequately describe all the aspects and hence a range of approaches need to be combined.

Simulation and modelling has the potential to be an extremely valuable decision support tool for estimating the storage, processing and networking resources needed to execute an application and to assess the risks of either underprovisioning or over provisioning of this resource. Modelling uncertainty is essential to allow sensitivity and robustness analysis to be performed. This allows the impact to be assessed and then managed of any deviations that might occur at execution time, e.g. variation in performance of a resource supplied by an infrastructure provider or changes in how an application is used by its consumers.

Building models of real-time applications is a labour intensive and skilled process and hence not without cost. However, the benefits are being able to go beyond conventional approaches to provisioning real-time applications (planning for the worst case and consequent massive over-provisioning). Instead, more efficient and flexible approaches can be used that centre on the most likely behaviour, the probable deviations and how to managed the consequences of these deviations.

It should be clear from this report that whilst real-time application modelling is certainly possible, good tool support is currently lacking, certainly in terms of tools that ‘business users’ can understand rather than modelling experts or IT professionals. Addressing this problem is now the next step for the modelling work in IRMOS

9. Internal References

[1] WP5PlanningandSimulating_IRMOSRealTimeApplications.v4_notrack.doc
[2] WP5_Mapping_ process_analysis_v0.1_USTUTT_IT_Inn.doc,
[3] WP5_requirements_SC_modelling.doc
[4] Towards T5.1.2/D5.2 – A Tool for specifying real-time applications requirements

10. References

[5] IRMOS_WP2_D2_1_1_v1_0 "Initial version of Requirements Analysis Report" http://www.irmosproject.eu/Deliverables/Download.aspx?ID=22
[6] The Worst-Case Execution Time Problem — Overview of Methods and Survey of Tools, Reinhard Wilhelm, Jakob Engblom, Andreas Ermedahl, Niklas Holsti, Stephan Thesing, David Whalley, Guillem Bernat, Christian Ferdinand, Reinhold Heckmann, Tulika Mitra, Frank Mueller, Isabelle Puaut, Peter Puschner, Jan Staschulat, Per Stenstr¨om, 2008 ACM Transactions on Embedded Computing Systems (TECS), Volume 7 Issue 3
[7] Alan Turing, On computable numbers, with an application to the Entscheidungsproblem, Proceedings of the London Mathematical Society, Series 2, 42 (1936), pp 230-265
[8] Stochastic Process Algebras, Allan Clark, Stephen Gilmore, Jane Hillston, and Mirco Tribastone, in M. Bernardo and J. Hillston (Eds.): SFM 2007, LNCS 4486, pp. 132–179, 2007.
[9] 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
[10] 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.
[11] R. Bruni, H.C. Melgratti, and U. Montanari. Theoretical foundations for compensations in flow composition languages. In POPL, pp. 209–220. ACM, 2005.
[12] 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.
[13] Formalizing languages for Service Oriented Computing, Claudio Guidi, Technical Report, Department of Computer Science, University of Bologna, UBLCS-2007-07, March 2007.
[14] A. Pnueli, The temporal logic of programs, Proc. 18th IEEE FOCS46–57, 1977.
[15] A. Kraiss, F. Schoen, G. Weikum and U. Deppisch, Towards response time guarantees for e-service middleware, IEEE Data Engineering Bulletin 24 (2001) (1), pp. 58–63.
[16] 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

[17] PEPA web site, www.dcs.ed.ac.uk/pepa/

[18] PRISM web site http://www.prismmodelchecker.org

[19] Milner, R.: Communication and Concurrency. Prentice-Hall (1989)

[20] 18. Hoare, C.: Communicating Sequential Processes. Prentice-Hall (1985)

[21] Gotz, N., Herzog, U., Rettelbach, M., TIPP—a language for timed processes and performance evaluation. Technical Report 4/92, IMMD7, University of Erlangen-Nurnberg, Germany, 1992.

[22] Bernardo, M., Gorrieri, R., Donatiello, L., MPA: A Stochastic Process Algebra, Technical Report UBLCS-94-10, Laboratory of Computer Science, University of Bologna, 1994.

[23] Strulo, B., Process Algebra for Discrete Event Simulation, PhD thesis, Imperial College, 1993.

[24] Priami, C., Stochastic π-Calculus. The Computer Journal 38(6), 1996.

[25] 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.

[26] C. Baier and M. Kwiatkowska, Model checking for a probabilistic branching time logic with fairness, Distributed Computing, 11:125–155, 1998.

[27] The Worst-Case Execution Time Problem — Overview of Methods and Survey of Tools, Reinhard Wilhelm, Jakob Engblom, Andreas Ermedahl, Niklas Holsti, Stephan Thesing, David Whalley, Guillem Bernat, Christian Ferdinand, Reinhold Heckmann, Tulika Mitra, Frank Mueller, Isabelle Puaut, Peter Puschner, Jan Staschulat, Per Stenstr¨om, 2008 ACM Transactions on Embedded Computing Systems (TECS), Volume 7 Issue 3.

[28] The Landscape of Parallel Computing Research: AView from Berkeley, Krste Asanovic, Ras Bodik, Bryan Christopher Catanzaro, Joseph James Gebis, Parry Husbands, Kurt Keutzer, David A. Patterson, William Lester Plishker, John Shalf, Samuel Webb Williams, Katherine A. Yelick, Electrical Engineering and Computer Sciences University of California at Berkeley, Technical Report No. UCB/EECS-2006-183
http://www.eecs.berkeley.edu/Pubs/TechRpts/2006/EECS-2006-183.html

[29] M. Ravasi, M. Mattavelli, P. Schumacher, and R. Turney, "High-level algorithmic complexity analysis for the implementation of a motion-JPEG2000 encoder," in *Proceedings of the 13th International Workshop on Integrated Circuit and System Design. Power and Timing Modeling, Optimization and Simulation (PATMOS '03)*, pp. 440–450, Torino, Italy, September 2003.

[30] Image 80486DX2 , GNU Free Documentation License, http://de.wikipedia.org/wiki/Benutzer:Appaloosa

[31] D.J. Colling, L.W. Dickens, T. Ferrari, Y. Hassoun, C.A. Kotsokalis, M. Krznaric, J. Martyniak, A.S. McGough, and E. Ronchieri. "Adding Instruments and Workflow Support to Existing Grid Architectures". Volume 3993 of Lecture Notes in Computer Science, pages 956-963, Reading, UK, April 2006.

[32] L. Guo, A. S. McGough, A. Akram, D. Colling, J. Martyniak, M. Krznaric "Enabling QoS for Service-Oriented Workflow on Grid". The proceedings of IEEE 7th

International Conference on Computer and Information Technology, 16th-19th Oct, 2007, Aizu, Japan.
[33] L. Guo, A. S. McGough, A. Akram, D. Colling, J. Martyniak, M. Krznaric "QoS for Service Based Workflow on Grid". The proceedings of UK e-Science 2007 All Hands Meeting, 11th-13th,Sep, 2007 Nortingham, UK.
[34] F. Baligand, D.L. Botlan, T. Ledoux, and P. Combes, "A language for quality of service requirements specification in web services orchestrations", LNCS 4652, Springer-Verlag, 2007, pp. 38-49.
[35] G. Canfora, M.D. Penta, R. Esposito, F. Perfetto, and M.L. Villani, "Service composition (re)binding driven by application-specific QoS", LNCS 4652, Springer-Verlag, 2007, pp. 141-152.
[36] L. Zeng, B. Benatallah, A. H. Ngu, M. Dumas, J. Kalagnanam, and H. Chang, "QoS-Aware Middleware for Web Services Composition", IEEE Transactions on Software Transactions, vol. 30, no. 5, 2004, pp. 311–327.
[37] M. C. Jaeger, G. Rojec-Goldmann, and G. M¨uhl, "QoS Aggregation for Service Composition using Workflow Patterns", In Proceedings of the 8th International Enterprise Distributed Object Computing Conference (EDOC'04), 2004, pp. 149–159.
[38] J. K. Lee, H. S. Kim, S. H. Kuk, and S. W. Park, "Extended BPEL System for e-Engineering Framework Considering the Characteristics of Mechanical Engineering Services", IEEE International Conference on Services Computing, 2008.
[39] UML Profile for MARTE, Beta 2 (ptc/08-06-09) - June, 2008
[40] B.B. Markelos,' Numerical Methods', Symmetria Publications 1999
[41] ZunZun.com Online Curve Fitting and Surface Fitting Web Site, www.zunzun.com
[42] JMP statistical analysis software, http://www.jmp.com
[43] Mathworks web site, http://www.mathworks.com
[44] GNU Octave web site, http://www.gnu.org/software/octave/
[45] Scilab open source Platform for numerical computation web site, http://www.scilab.org/
[46] SENSORIA public deliverable D1.4a: UML for Service-Oriented Systems, Nora Koch, Philip Mayer, Reiko Heckel, László Gönczy, Carlo Montangero , 2007.
[47] Defined Workflow Patterns, Workflow Patterns Initiative, http://www.workflowpatterns.com/patterns/index.php
[48] Language evaluations, Workflow Patterns Initiative, http://www.workflowpatterns.com/evaluations/index.php
[49] YAWL – Yet Another Workflow Language, http://www.yawl-system.com/
[50] BPMN – Business Process Modeling Notation, http://www.bpmn.org/
[51] Web Services Business Process Execution Language Version 2.0, http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html
[52] Service oriented architecture Modeling Language (SoaML), http://www.omg.org/docs/ad/08-11-01.pdf
[53] www.tomshardware.com

Annex A.

A.1. Workflow

This section will outline the lifecycle of the workflow within the IRMOS framework. During this phase no real-time guarantees are needed, but steps should be taken to enable this during the execution phase. The first part of the section looks at different workflow modelling languages.

For the purposes of this section a model describes the parts of the workflow and how they are dependant, as apposed to performance models that are essentially mathematical representations of the behavioural semantics for the workflow activities. A formal behavioural semantics for workflows together with examples of behavioural models of workflows is given in depth in Annex C.

A.1.1. Workflow description models

According to the IRMOS terminology document a Workflow is defined as follows: “Workflow is the automation of a business process, in whole or part, during which documents, information or tasks are passed from one participant to another for action, according to a set of procedural rules”. A workflow description is thus a model or description of a workflow, in essence a description of the business process. A workflow can be managed by a workflow management system that assists in automating the process.

There exists a multitude of languages for describing workflows, and a corresponding set of workflow engines to enact the defined processes. These range from the high level descriptions more geared toward the human understanding of processes, to the technically oriented approaches focused on the actual automation of the processes.

In the definition of workflows involving or describing real-time systems the issue of time and timely allocation of necessary resources (both human and technical) is naturally important. For IRMOS it is important that the workflow models provide the necessary information for in later phases of the application lifecycle. This includes simulation, planning and execution of the business processes (IRMOS Applications).

A.1.2. Evaluation criteria

To aid in the selection of a modelling language for IRMOS workflows we make use of the set of defined Workflow Patterns [47]The workflow patterns are partitioned into four perspectives: control-flow, data, resource and exception handling. In addition to the patterns [48] provides an evaluation of different approaches in their support of the patterns. To find the approach most suitable for IRMOS it is natural to select the most important patterns and then make use of the evaluations in order to choose the right technology.

The following list enumerates the patterns (ordered by category) that are most important to real-time applications:

- Control-flow
  - Having as much expressive power as possible is useful when describing control flows. This is no different for real-time systems than for other systems.
- Data
  - There are various patterns related to push and pull patterns and how tasks and workflows communicate internally and externally. For IRMOS the concept of streams of data is important, but there is no pattern that explicitly defines this.
- Resource
  - Direct allocation
  - Role Based allocation
  - Capability Based allocation
- Exception types
  - Deadline Expiry
  - Resource Unavailability
  - Constraint Violation

A.1.3. The contenders

The following languages have been identified as the most appropriate candidates for use as the IRMOS workflow language.

YAWL - Yet Another Workflow Language [49]. YAWL was created based on a rigorous analysis of existing workflow management systems and languages. It has come out of work performed at the Queensland University of Technology and the Eindhoven University of Technology. The concept is based on a formal semantics based on Petri-nets.

BPMN – Business Process Modeling Notation [50] created by the Business Process Management Intiative (BPMI) is currently a standard under the governance of the Object Management Group. The goal of the standard is to provide a standard that is usable for the whole range of users from the business analysts to the technical developers.

BPEL - Web Services Business Process Execution Language [51] is a language to model Web service based business processes. The core concept is the representation of peer-to-peer interactions between a process and its partners using Web services and an XML-based grammar. It is built on top of WSDL (both the processes and partners are modelled as WSDL services). This standard is owned by OASIS and is currently in version 2.0.

A.1.4. Evaluation summary

For control-flow YAWL claims to support most of the identified patterns. BPMN supports more of the patterns than BPEL does.

Using the provided tooling, YAWL does not visualise task allocation to resources in the workflow net. This allocation is done in conjunction with the YAWL server, where different roles and resources are defined. In BPMN indication of allocation of tasks is possible using the swimlanes and pools notations.

BPEL does not, alone, support the definition of human tasks. In essence this means that BPEL has not been evaluated toward the Resource perspective of the Workflow Patterns. The scenarios defined in IRMOS do include human tasks, so using only BPEL will not suffice. The addition BPEL4PEOPLE that is currently under standardisation will provide this possibility.

For exception handling BPEL has support for Deadline expiry while BPMN in addition has support for Constraint Violation. According to the evaluation none of them support resource unavailability.

From a technical perspective it is important for IRMOS that the workflow specifications can be executed using some available engine. YAWL comes with an engine that is tailored to run YAWL based workflows with integrations of Web Services. The YAWL system is open source and available under the LGPL licence. However YAWL is not a standard, and to the best of our knowledge there is only one implementations available.

For BPMN the usual approach taken is to translate the BPMN to BPEL for execution. The BPMN standard document provides general mappings for this, but also states that there are known issues to the defined mappings and that these will be remedied in the next version of the standard.

There exists multiple engines for execution of BPEL, such as AciveBPEL and Apache ODE, many of these are open source.

A.1.5. Conclusion

Through investigations of the different technologies and the needs for these in IRMOS it seems necessary to allow for the use of more than one workflow technology depending on the purpose of the description. In fact workflow descriptions may exist on different levels in IRMOS. At the lower level we need something that can be executed in a controlled manner, orchestrating the different services. For this purpose BPEL seems a sensible choice both due to the multitude of implementations and the fact that there is experience in using this technology within partners of the project.

For high level descriptions, for instance describing tasks where Application Components are used by different roles to perform a business process another language can be used. At this level one of the key purposes of the description is human comprehension as one could imagine that non technical people should be able to define high level workflows possibly with timing constraints in planning of their business processes. For this purpose both YAWL and BPMN are suitable.

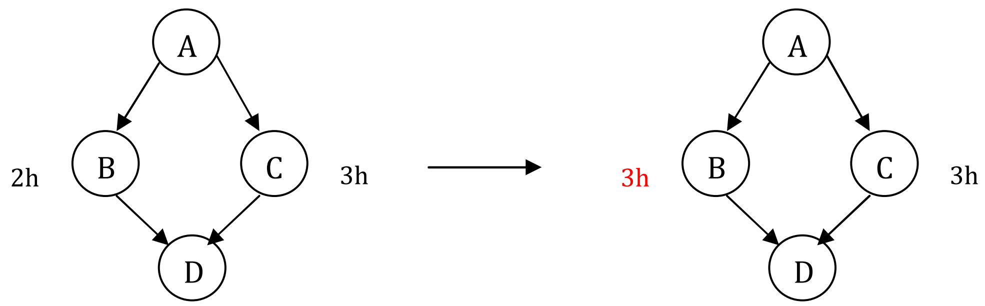
By applying Model Driven technologies to this area IRMOS can allow for the definition of workflows using a set of defined languages, and through transformations create appropriate descriptions for execution or other use at different levels. One example would be by providing transformations between BPMN and BPEL (albeit as mentioned not always completely straightforward). These transformations would pick up the parts needed from the higher level descriptions and the generated lower level descriptions and could then be augmented if needed. Using the model driven approach a greater degree of freedom exists in the choice of what to use to provide the descriptions, as long as a target platform is defined, and in this case it is BPEL for the workflow execution.

A.2. Workflow Lifecycle

The rest of this section provides an overview of the IRMOS workflow lifecycle. This provides the background context for modelling as a whole within IRMOS. The descriptions are brief since it is not the purpose of this document to provide a complete detailed view of the whole IRMOS development process.

From a high level point of view, the lifecycle an IRMOS workflow can be categorized into the following different phases:

1. *Offline phase:* The key aim of this phase is the creation of abstract workflow descriptions and Application SLA (A-SLA) template(s) that could be used during the online negotiation phase and includes the following steps:
    a. *Benchmarking:* At the first step of this offline phase, the application provider is responsible for benchmarking his/her ASCs in an Infrastructure Provider environment.
    b. *Mapping:* The benchmarking results will be used to produce a behavioural model for each ASC (For more insight into the mapping process see Section B.1).
    c. *Modelling:* This results of the benchmarking and mapping process will be furthered refined by the MAP tools, and returned to the Application Provider. In more detail, the MAP tools will analyze the abstract workflow description and produce an optimized one by identifying and analysing interdependencies among the service components that comprise the workflow. For example, if two tasks B and C both need to be finished in order for a third one D to start execution (synchronization), then the timing requirements for both should be set to the slowest one in order to cut on costs.



    At the end of this process, the bounds on resource performance e.g. upper and lower limits on QoS needed and an optimised version of the abstract workflow description will be produced.
    d. *A-SLA template creation:* Using these results, the Application Provider will be able to produce and store Application SLA template(s) for his/her abstract workflow(s) that will be used by the consumer during the online phases.

At this point, it should be stressed that within the IRMOS framework benchmarking is considered to be a special case of execution of an application and that the two online phases that follow up are involved in this offline process. To this end, a special Application SLA template is used for negotiating the resources needed for benchmarking.

2. *Online phase:*
    a. *Negotiation and Reservation:* During this phase the Framework Services are in charge of negotiating an application SLA and performing advance reservation on the resources that are needed for the execution of the application requested by the consumer.
        i. *A-SLA offer:* The negotiation phase is triggered with the customer making a request for SLA negotiation and providing an abstract workflow description. The consumer will fill one of the A-SLA templates produced in the previous phase according to his/her needs using high level performance parameters.
        ii. *Mapping*: These high level performance parameters will be mapped to the low level ones using the results of the offline mapping and modelling phase for each ASC in the abstract workflow description,
        iii. *Discovery*: At next step appropriate Infrastructure Providers, based on their advertisements and the low-level QoS constraints as obtained at the end of previous step, are discovered. In doing so, the technical SLA templates of each Infrastructure Provider are being filled according to the abstract workflow description and the low-level constraints and sent to the corresponding Infrastructure Provider.
        iv. *Selection:* Each of these Infrastructure Providers will answer with an offer, and these offers will be prioritized according to the customer's preferences. At next step, the best offer is selected and sent back to the Application provider for approval.
        v. *Reservation:* Upon approval of the technical SLA, a request for advance reservation of the required resources is sent to the winning Infrastructure Provider. At the end of a successful reservation, the Infrastructure Provider resources are booked and are ready to use within the reserved time interval. If the reservation of the Infrastructure Provider resources fails, then the next candidate Infrastructure Provider from the prioritized list is selected and process is repeated.
        vi. *Creation of concrete workflow description:* The information included in the technical SLA is used to transform the abstract workflow description into a concrete one.

    It should be stressed that during the entire negotiation phase no real-time guarantees are needed, but steps should be taken to enable this during the execution phase.

    b. *Execution and Monitoring:* In this phase, the execution of the ASCs is being orchestrated according to the workflow description. The consumer will be able to receive monitoring data about the execution of every ASC in the workflow. At any time during the execution, the application user should be able to pause and restart the workflow execution as well as stop and re-negotiate the SLA if he/she wishes to do so. Furthermore, the consumer will be notified in case of exceptions and/or violations, and will be able to trigger a re-negotiation process. The architecture, interfaces and behaviour of the monitoring process are covered in depth in Section A.5.

An overview of the workflow lifecycle is depicted in the following figure.

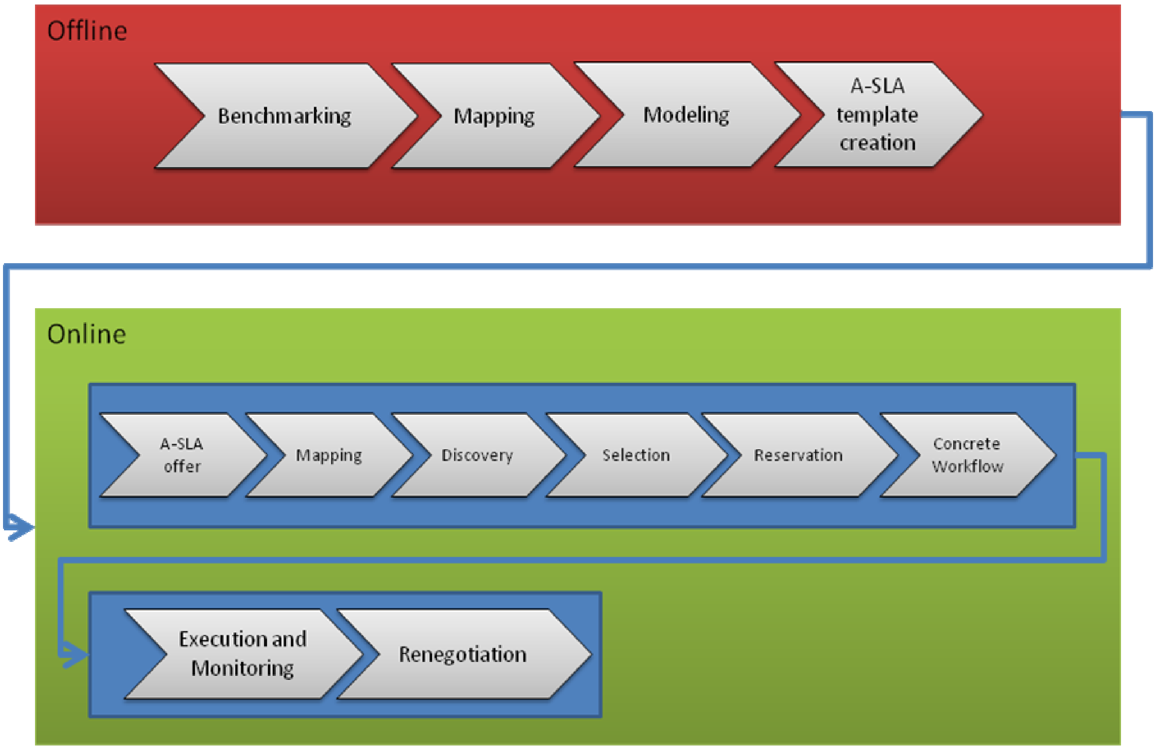


Figure 44: From abstract to concrete workflow

A.3. Models used for Application Adaptation

Adaptation of applications in the sense of IRMOS describes the transformation of a more or less monolithic, non-distributed application software (or software package) into a distributed chain of application components (ACs). Parts of the application become a service, i.e. one or more application service components (ASCs) while the user's interface turns into an application client component (ACC). The application can be represented as a graph consisting of ACs as nodes and edges as interconnections. Details of how application component behavioural models can be constructed for the purposes of deriving performance estimates in combination with workflow behavioural models can be found in Annex C.

In addition to known SOA approaches real-time aspects have to be considered. They are putting extra requirements on the communication between the different application components. So beside functional parameters needed for interconnecting the components (like network addresses) technical parameters have to be defined representing the required performance values (e.g. bandwidth).

Moreover IRMOS services will be used for interactive applications, i.e. an user may influence the required performance dynamically. In IRMOS, ISONI as our Infrastructure Provider does not allow dynamic changes in the requested (virtual) infrastructure so we can neglect this issue in modelling and focus on worst cases, i.e. considering values for the highest possible performance required in an application. Moreover typical interactive applications like video editing will swap between zero and 100% resource usage immediately, e.g. when the operator toggles between "pause" and "play". Therefore we cannot rely on statistical probabilities regarding user interaction in such scenarios. Hence the application model may remain stationary in this respect.

Just as the application itself an AC can be regarded as static (apart from changes in different versions) but with application specific parameters for each current workflow affecting the performance parameters (e.g. frame rate, image size, number of users). The service part of an application along with its concrete parameters will finally be translated into a virtual service network (VSN) for execution on the virtualized infrastructure[18]. As only fixed amounts of resource can be booked (and must be paid) scheduling for optimizing resource usage is done on this underlying level so that it gets out of scope of the application level discussed here.

So the application can be modelled as a graph with ASC as vertices and edges representing (network or storage) interconnections. The key parameters turn out to be throughput (alias bandwidth), latency (alias response time or delay) and jitter – characterising each vertex as well as each edge. While these parameters can directly be mapped to the interconnections (edges) they have to be mapped to processing

[18] The primary targeted infrastructure for IRMOS is the ISONI platform.

performance values for the vertices. This affects the ASC discussed in the following section.

A.3.1. Functional Requirements for SC Models

The main outcome of the MAP tools is an IRMOS workflow description. An IRMOS workflow (i.e. an IRMOS application[19]) basically consists of the ASCs and its interconnections. The ASC model has to describe each component in a way sufficient for creating and analysing the workflow by remaining agnostic to the specific application.

Generically an ASC is doing processing on input data and/or generates output data. Each ASC is characterized by this input/output data described in high level (application specific) parameters like image size, frames per second, number of users etc. When it comes to meet QoS constraints these will need to be mapped to low level (i.e. technical) parameters like throughput or latency. These parameters affect the edges, i.e. network and storage and also affect processing parameters like CPU and memory.

Throughput (i.e. bandwidth) can be regarded as the common characteristic covering processing, network and storage. For optimum resource usage this parameter has to be aligned for two consecutive members of the processing chain. Of course its value may change during the chain, e.g. when an ASC is performing a format conversion.

The processing throughput is influenced by the algorithm itself, its parameters, input/output data formats and the execution environment, i.e. type of CPU, number of CPUs, size and speed of CPU cache, cache memory, main memory as well as the bandwidths of involved busses. Figure 45 shows a simplified model of an ASC.

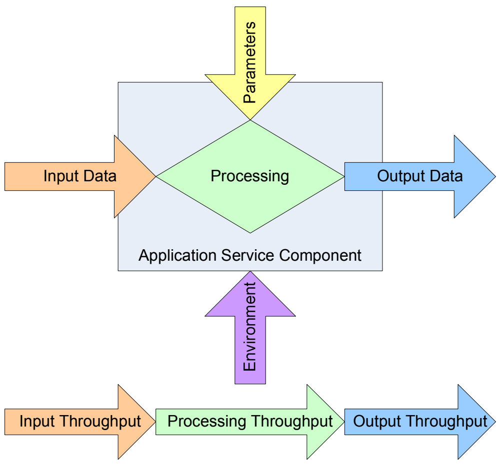


Figure 45: Simplified Application Service Component Model

Hence processing throughput is a function of the following entities:

- processing (algorithm)[20]
- processing (algorithm) parameters
- input data (formats etc., may be null, one or multiple)

[19] The terms (IRMOS) workflow and (IRMOS) application are regarded as synonyms in this context.
[20] "Algorithm" maybe does not fully fit to the description of the variety of different kinds of processing in an ASC. Therefore "processing" might be a better term.

- output data (formats etc., may be null, one or multiple)
- execution environment, i.e. number of processors, the system performance

In general the algorithm parameters, input data and output data are known; the values for the system performance have to be found for the given algorithm. Usually a comprehensive description of the processing algorithm is unavailable but its behaviour is considered constant for a given set of the rest of the parameters. Real world ASCs will have a couple of different functions ("algorithms") with different parameters and behaviour, e.g. an ASC doing image conversions with different formats, filters and sizes; formats and sizes are part of the I/O data, the applied filter counts as processing/algorithm parameter.

Regarding the interconnections of ASCs to shape an application, input and output specifications (i.e. the interfaces) must be aligned to each other. In order to avoid wasting resources the environmental parameters of each ASC have to be adjusted to maintain the same height of throughput. The network parameters for this connection can be derived from these I/O parameters – or its mapping respectively (Figure 46).

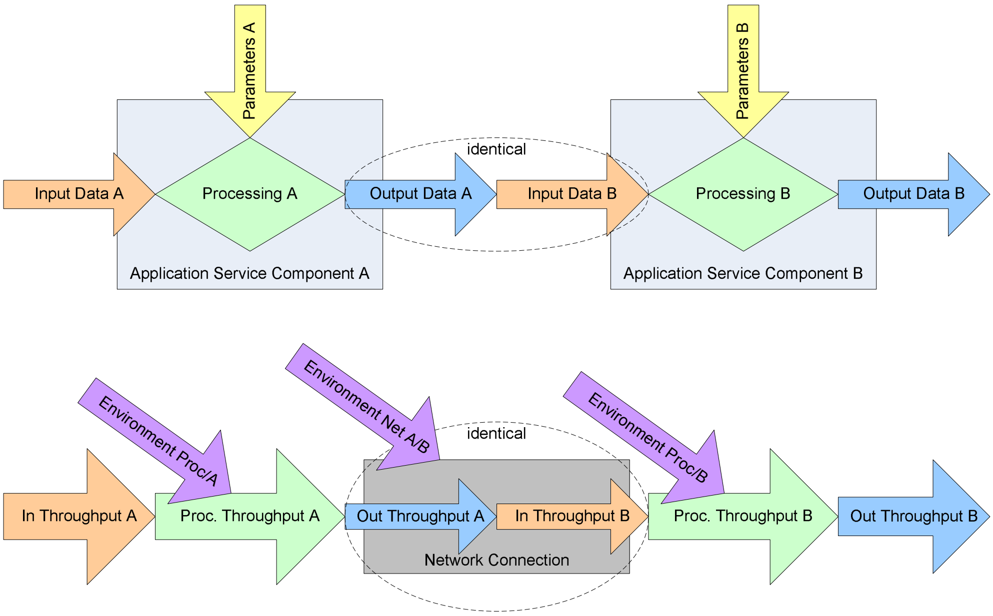


Figure 46: Throughput of concatenated ASCs

The latency of an application can be derived as sum of the latencies of each ASC plus each network connection in the chain. Regarding jitter things might get more complicated as the jitter of one component may affect the jitter of the following in different ways.

A.4. Models used in Requirements Mapping

The idea of mapping is to define a tightly coupled association between high level application parameters and low level parameters that is appropriate for a particular application instance. In depth technical details of mapping techniques are given in A.4.2 and B.1.

In an ideal world a mapping would take the form of an elementary equation. An example best illustrates this idea. Below we define a simple hypothetical formula that computes the amount of MFLOPs (mega-floating point operations) that will be required to execute colour correction in an efficient manner. This is the same formula used in the case study from Section 4.

Define processing MFLOPs = benchmarking constant * number of frames * resolution of each frame, where

- Resolution is in Mpixels per frame
- Number of frames = duration * framerate
- Processing is in MFLOPs
- Benchmarking constant is a factor that is derived during a benchmarking process on a reference platform

For example, to perform colour correction of 30 minutes of 24fps 2k film and with a benchmarking constant of 1.5 will require 1.5 * 24 * 30 * 60 * 2 = 130 GFLOPs (giga floating point operations) of processing resource. The total processing time then required for an uninterrupted activity is defined to be

processing MFLOPs/platform MFLOPS

where platform MFLOPS (which is MFLOPs per second) is the amount of processing power provided by the platform on which the software is run. Note, the above formula is not the amount of time it will take for a human operator to complete colour correction for that amount of film. It is the amount of CPU time that will be taken in executing floating point operations that are required by the software. Nor does the above formula capture all the resources that might be required from a CPU during execution. The formula specifies nothing about integer operations that may be required, for example. In general for an application that provides features that are primarily a set of numerical algorithms, as in the case of colour correction, then MFLOPs is likely to be the dominant measure of overall processing time. For a feature in a different domain, for example merging database tables, then MFLOPs will not be relevant and integer operations are much more relevant.

The above example shows an ideal that is rarely achievable in practice. There is usually no such exact function available except for simple algorithms that provide a single feature within an application component and that do not interact with other components.

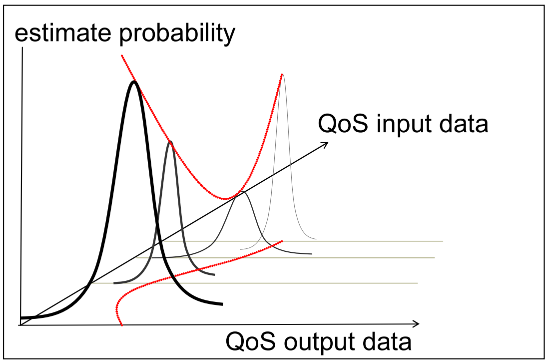


Figure 47: Stochastic Mapping

For complex algorithms embedded in a software application it may not be possible to isolate the factors that determine with absolute certainty how parameters affect the overall performance. It is very unlikely that there will be one simple benchmarking constant that acts as a multiplying factor as in the above formula. Nor will it be certain exactly how low level parameters such as MIPS or MFLOPS are defined as combinations of high level parameters. It is much more likely that through some kind of experimental process such as benchmarking a statistical correlation, like that shown in Figure 47, will be defined. The diagram illustrates that when measuring some output QoS parameter (such as time taken to complete some feature) based on some input QoS parameter (such as size of input) the answer may vary each time it is measured. Such variation is shown in the diagram as a probability distribution that details the error estimation in the given measurement.

Such statistical approximations are usually very specific to well understood algorithms. They also work well when the data they are based on can be fitted against continuous differentiable functions on real numbers. For example, using standard curve fitting techniques to fit a quadratic surface to experimental data. When the experimental data being used does not closely resemble such a continuous differentiable function any curve fitted to the data will not be a true representation of the underlying complexity. In such a case predictions based on such a curve will be unreliable if used to extrapolate to any extent beyond the available data.

When source code is available it may be possible to analytically determine the computational complexity of core algorithms [29]. At least crude upper bounds on the complexity may be possible. Determining how that complexity, which will be measured in terms of some atomic algorithmic operations, is then translated into MFLOPS or MIPS is something that is extremely difficult to establish analytically. In practice, again, this is most likely to be achieved through experimentation. In a commercial context it is unlikely core algorithms will be available in the form of source code except where the software is implementing a standard algorithm that already exists in the public domain.

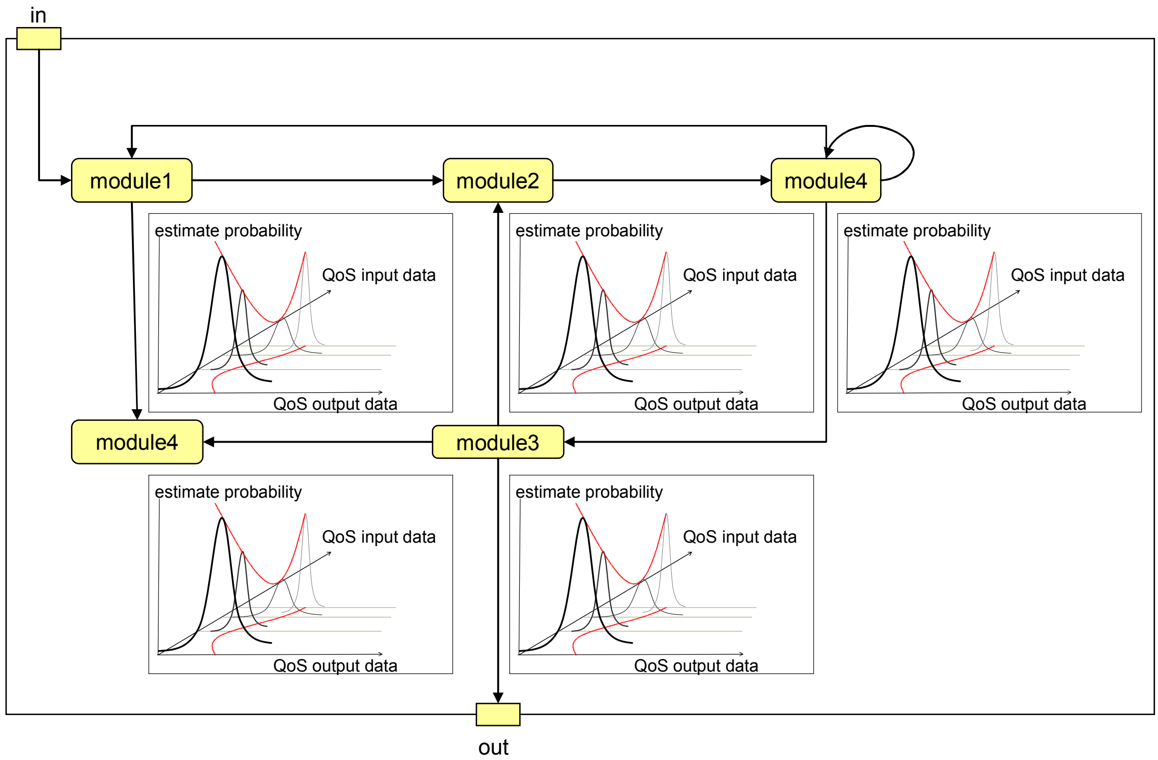


Figure 48: Decomposition of internal structure of an application component

In some cases the overall structure of an application component may be known in terms of a decomposition into modules that represent core algorithms. It may also be that these modules are well understood in terms of complexity. In that case the overall behaviour of the component may be too complex to predict accurately as a continuous differentiable curve, but each module performance can be described in such a form.

If the interactions of these modules are also defined in terms of interface protocols, then the same modelling techniques that are used to estimate application performance can be used to estimate application component performance, i.e. estimate component performance based on complexity characteristics of the individual modules. These characteristics will probably be stochastically defined through benchmarking. The probability distributions annotating Figure 48 are attempting to illustrate such complexity characteristics associated with each module. These probability distributions would define the stochastic mapping functions of the sort shown in Figure 47 that do relate lower level parameters to high level parameters. From such descriptions of the application component it is then possible to construct models that can describe the sort of discontinuous behaviour that results from interactions between the modules.

A.4.1. Worst Case Execution

Worst-case execution time (WCET) analysis is an approach for discovering resource performance requirements (throughput/bandwidth, latency, availability, jitter) and resource time allocation requirements (start, end and duration). The techniques can be useful where the hardware and software platforms are very well understood at a high degree of granularity. WCET analysis is a well established field concerning hard real-time systems (HRTS) [6]. For IRMOS the underlying hardware is virtualised, distributed and dynamically variable. Therefore these techniques will not be directly applicable. They are nevertheless informative in the difficulties they encounter in making accurate predications, and the level of detail required in making such predictions.

Given that any application will be executed as an executable binary, a machine executing the code will be interleaving the machine instructions for the application binary with many other tasks. Even if the application is the only one running there will be many other operating system programs that will be executing concurrently with the application. For an accurate WCET it is necessary to know what all such programs might be, how long they require, what the scheduling policy is, the estimated CPU usage for the application and how long it takes to switch between programs.

Modern CPUs have large amounts of level 2 cache. To make accurate predictions about the WCET of an application it is necessary to know when the application code is resident in the cache and when it is not. It is necessary to know how long it takes to refresh the cache. CPUs also contain pipelines that hold various predicted branches for code execution. These pipelines are filled based on heuristics that determine which branch is likely to be taken. It is necessary to know how often such heuristics fail and how long it takes to refresh a branch if backtracking is required. Thus a very good understanding of the likelihood of cache misses is necessary.

As well as knowledge of the CPU architecture nature of the RAM and hard storage used by the implementation platform are also required. On occasion paging will occur. To accurately calculate WCET it is necessary to know if and how often paging will occur and how long that takes. Such issues are sometimes referred to as scheduling anomalies.

The hard real-time system (HRTS) paradigm is used where the system needs to operate under strict constraints, i.e. strict deadlines from event occurrence to system response, and where the impact of constraint violation is of critical nature. Mission critical applications are regarded as HRTS. A HRTS constraint must be met regardless of the system load (presuming system load will be restricted in some way to be in some predefined boundaries). The product of an operation completed after a deadline is considered useless and the system is ultimately considered to be at fault. WCET analysis considers the execution time of each system task in isolation and the overall system performance is analysed on the bases of WCET results for each task. The analysis can be performed at different levels, i.e. tasks can be defined at different levels of abstraction, but in order to have reliable results WCET should be performed at the lowest possible level, the closer to the hardware level the better. WCET analysis yields an upper bound of the execution time of a task on a specific resource and the required resource performance and scheduling should be such that this upper bound is within the imposed timing constraints.

In the context of IRMOS, when considering WCET analysis the first thing to note is that we are dealing with a system where the constraints of the individual tasks are not as extremely strict and that the impact of constraint violation is not usually critical, although it can be costly. This kind of system fits the paradigm of soft real-time systems (SRTS) more closely than of HRTS.

SRTS operates under relaxed (soft) constraints, i.e. they can tolerate deadline violation (lateness), or the application might respond with decreased quality. The SRTS paradigm is suitable for real-time interactive systems, where failure to meet a constraint introduces a higher cost or decreased quality but it is not critical and of extreme consequences (e.g. physical damage or loss of human lives). In SRTS the output of an operation completed after a deadline is still of use but with higher price and/or lower quality, and the system is still considered to be operational. In this context WCET analysis will probably be a very pessimistic tool; it is very likely that the resulting system design will be massively over-provisioned (very costly), and will operate under strict load constraints. For IRMOS this is excessive and not desirable.

Furthermore, the use of WCET in the IRMOS would require exact guarantees and reliable QoS from the virtualisation layer that an Infrastructure Provider exposes to the users. Some of the QoS parameters of an Infrastructure Provider will include at least some level of uncertainty in the performance characteristics and will not provide 100% guarantees.

Finally, the issue of variability of the system load will need to be considered. In the context of IRMOS the nature of the operations performed by tasks and the size of the managed data could vary within quite large margins depending on the particular application. In this respect, the best we should hope for at the system analysis stage is to have some well behaved stochastic characteristics of the task complexity and load size rather than the exact characterisation of resource requirements needed for WCET.

In summary, any performance estimation technique for IRMOS will need to deal with multiple layers of non-trivial and possibly stochastic uncertainty. This coupled with the fact that WCET caters only for the worst case scenario implies that WCET will suggest a massively over-provisioned, under-utilised and non-cost-effective system. Other techniques should be employed for performance analysis for the purposes of IRMOS.

A.4.2. Benchmarking

Benchmarking is a methodology for measuring the performance of an application component in the context of a particular platform of known characteristics with known input data. Benchmarking is a form of experimental analysis in that it works best when providing experimental verification of a given hypothesis. Usually benchmarking output data is not exact, but will give slightly different values from one run to the next. Initial experiments in WP6 have shown that the resources needed by an application can differ greatly in the case of sharing usage of a CPU and depending on the type of the application that is sharing the processor.

Benchmarking is helpful in determining constants for a mapping function whose general form is already known or assumed. Benchmarking is not a panacea that can determine the form of a completely arbitrary mapping function. For example, trying to use benchmarking data to discover if an algorithm has polynomial complexity where the degree is unknown is difficult and requires a very large amount of experimentation.

Indeed, the general problem of determining the computational complexity of an arbitrary algorithm is undecidable [7]. Whereas, given that an algorithm's complexity is polynomial of degree two say, then it is relatively straightforward to produce an effective set of experimental input benchmarking data that can be used to determine what the relevant constants are in the quadratic polynomial. However, through a series of tests in which we assume the form of the behavioural function of the algorithm we can approximate the real life problem, especially in cases where there will be no one to pinpoint the general complexity of the software.

Benchmarking data can be used to estimate the mapping between parameters through statistical curve fitting provided that the data represents some kind of continuous differentiable surface whose structure is known apart from some constant factors. Unless the benchmarking platform is generic in the context of the application domain the results do not provide reliable data for performance estimation purposes. For example, Figure 49 summarises some benchmarking results from a popular hardware review web site [53] for transcoding MPEG-2 video material to MPEG-4.

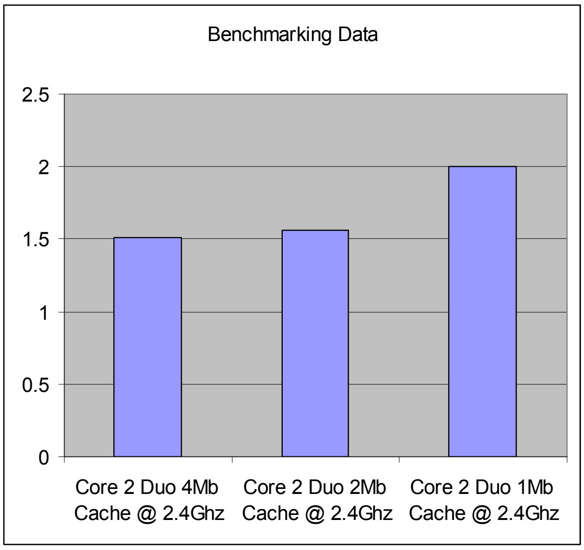


Figure 49: Benchmarking Example, (time measured in hours and minutes)

Figure 49 is an example of benchmarking a specific software application on the same platform and the same chip, except that the three different examples have different amounts of level 2 cache. In this example increasing the cache size from 1MB to 4MB reduced the transcoding time by nine minutes, which is a decrease of 7.5%. This represents a noticeable difference created by an item that will not be visible if benchmarking is done on a virtualised platform that does not provide information on cache size to the application provider. If there are only three examples of this kind of factor that are not visible within an application SLA then the overall uncertainty in a benchmarking results could easily be of the order of 25%. That assumes all other resource data is certain and all performance is exactly known, which clearly will not be the case in practice.

If application components are benchmarked on a time-shared host then the resulting mapping estimates may be valid when applied to a similar time-shared host with similar resource loading. However, even if two applications are time-shared in a fixed way (i.e., round-robin 50% - 50%), it may happen that the performance of one application is affected by the data size that the other application is dealing with, due to different patterns of cache access interferences. Having a similar level of sharing between the benchmarking phase and the run-time phase would not eliminate this problem. With luck, if the component is deployed on a dedicated host, the mapping will generate an over-estimation of required resources. For example a memory access that in dedicated machines could be found in the cache memory, on a shared host may be found in RAM, thus resulting in more CPU cycles in order to execute the same command. Mapping results based on benchmarking on a dedicated host will similarly be valid for deployment on dedicated hosts.

In fact an Infrastructure Provider may well never make any guarantee of allocating a dedicated platform. Since IRMOS is specifically intended to provide a virtualised layer, abstracting service infrastructure from platform hardware, it would be difficult for the consumer to determine if a host is shared or dedicated. It is also not clear how a consumer can know what level of sharing between VMUs on the same hardware will be present. That can also affect the validity of using benchmarking results to obtain parameter mappings. Due to the fact that IRMOS uses a virtualized Execution Environment approach, there can be an increased guarantee that these dangers will be avoided. This is investigated in the context of WP6.

In a streaming context such as IRMOS provides, benchmarking will result from monitoring component behaviour. The monitoring data will be provided from two sources. The first source is the metering service of the Infrastructure Provider infrastructure. This will cover basic low level parameters such as CPU cycles, storage (or memory) requirements and network bandwidth. The second source will be the application components themselves, which will monitor high level requirements posed by them and which can not be measured in a different way since they are application specific and Infrastructure Providers are typically application unaware. These may include for example frames per second for multimedia applications. The results from both these cases will be gathered and sent to the mapping service, which will handle the modelling of each individual SC. The results of the modelling process could then be stored in a repository; from which all other involved services such as the MAP tools will be able to retrieve them. For the initial stages of benchmarking, since this procedure will not be different from normal executions as far as an Infrastructure Provider is concerned and we may have no knowledge of the resources needed for this process, a strategy could be to significantly overprovision, such that utilization should not exceed 100%, and then note the percentage actually used.

The benchmarking process needs to be conducted on the smallest parts of the application that can be individually measured. In practice this is likely to be at the application component level. As explained above, statistical curve fitting techniques are only valid for continuous differentiable mapping functions. Benchmarking should therefore be executed on processes that have this characteristic. Another reason, probably more important in the framework of SOIs, is that during the design of the application the developer may use the elementary SCs in whatever way he sees more fit in order to create the workflow. The potential for combining these SCs is exponential and has the potential to be dynamically configured. A developer may create an arbitrary workflow to orchestrate an application. If we benchmark on an application/workflow level, then every time a new application is implemented we will need to follow this process. If we benchmark on a SC level, then by having the models of the building blocks we will be able to provide a general model for all possible combinations. However, this approach inserts another parameter as far as execution during benchmarking is concerned. It is not certain that SCs will be able to run as standalone. As they are meant to be used inside a workflow, thus accepting input from another SC for example, then in order to be executed this input must be provided by different means. In this case, the SC developer must provide the input necessary for the component to run in a somewhat simulated manner.

We will also investigate the possibility of having application specific SLAs, so that for each application we have the parameters needed. These high level parameters will be defined in the ASCD and for these cases we will have benchmarking runs. So the ASC developer should provide a range of direct ASC input for which he wants to test the ASC and monitoring parts that will meter the performance of the requested high level indirect parameter. Examples of input variables in this case could be the resolution of the film to be processed and monitored indirect output (of the benchmark process) could be the fps produced by the ASC (which can later on be used as an "SLA template input"). The mapping service will then use this info (a number of test runs containing different resolutions, used resources and the according fps measured) and will produce a model function. The mapping service may be able to map numerical input (high level parameters) to numerical output (low level CPU, storage, network) as long as there are appropriate experiments that support this.

This will not influence the automation of the process, since the methods do not actually care about what kind of data they are processing. As long as the values of the input and the according low level monitoring data are provided, the association between them will be performed regardless of the nature of the parameters.

Pre-processing of the data set will be considered in order to ensure the benchmarking are as accurate as possible. The input given for the modelling process must be indicative of the SC's behaviour with the variation of the input parameters. This means that we must carefully choose the values that we use for the modelling in order to extract information from them in the fittest way. One methodology that ensures a good quality data set is the use of the Principal Component Analysis (PCA). Through PCA we can limit the dimensions of a multivariate data space to a lower dimension, but through keeping the information included in the set intact. The main point is to keep the data of the set that contribute the most to its variance. The use of this method will be investigated.

A.4.3. Data Sampling

The SC developer needs to provide a number of input values as input data points on which benchmarking must be performed. These values should be decided based on his expertise and his general knowledge of the SC and they must indicate the normative behaviour of the SC. Moreover he must indicate the range of values that these input data may take. Due to the fact that extrapolation for the estimation is highly unreliable, the prediction will be limited in the input range identified by the user, and for which he will provide a number of values for test runs. In order to avoid the aforementioned extrapolation, benchmarking should be performed also for the edge points of the value intervals specified by the SC developer. This data will later be the input for one of the statistical curve fitting techniques for parameter mapping.

Statistical curve fitting techniques work best when sample data is `orthogonal'. That is the input values are independent from a statistical perspective. Sampling inside an interval can be performed in numerous ways, the most simple is to divide it by the number of data points we need (depending on how many test runs we desire) and take the subsequent evenly distributed points. Another approach is to take more samples for an area that we expect is going to be met more frequently in real life executions. The possibility of having a validation set will also be investigated, in order to have a clear view of the model's behaviour and the error inserted during the estimation. State of the art statistical packages, such as JMP for example, will automatically generate orthogonal sampling points. These guarantee that benchmarking results will be the best possible for statistical curve fitting.

Within an IRMOS context it is also relevant to consider if there is a need for two benchmarking datasets. One with different input data in order to measure the component's behaviour with respect to the variation in this data and another one with input data suitable to be executed on a number of different CPUs, in order to check the dependency of the behaviour on the different hardware configurations. The effort will be towards using metrics that are as independent as possible from different architectures.

Another case is the smoothing of data based on criteria such as the mean value or the deviation from this for a number of neighbouring data points. Through this process we choose to disregard a number of extreme points in the data set so as to achieve better approximation (and eventually estimation) for all other values. The advantage with this process is that it eliminates part of the noise that may be inside the initial data set.

There are a number of available, both commercial and open source, tools that can be used in this stage, many of them containing a number of possibly useful methods, whose use and effect on IRMOS will be tested. Matlab [43] is commercial software that comes with a number of toolboxes. The most interesting for this case are the curve fitting and the statistics toolbox. Octave [44] is an open source version which resembles Matlab greatly and contains the same in most cases functions and structure.

Scilab [45] is another tool that resembles Matlab and contains statistical analysis tools. It was developed by INRIA and ENPC and is used by a number of European and national research projects. Scilab, unlike Octave, also has a graphical package that mimics the well-known Simulink toolbox by Mathworks. JMP [42] is a commercial product which is intended to be used for data statistics analysis and design of experiments. It has a number of features that can be used both for data pre-processing and model fitting.

A.4.4. Dwarfs

The Dwarfs methodology for estimating resource usage in high performance computing originated in Berkeley [28]. The basic approach is that many software programs have a behaviour that can be modelled according to a number of basic computational paradigms, known as *dwarfs*. Initially there was a list of 7 of these paradigms that was later on extended to 13. In more detail:
"A dwarf is an algorithmic method that captures a pattern of computation and communication. The Seven Dwarfs, constitute equivalence classes where membership in a class is defined by similarity in computation and data movement. The dwarfs are specified at a high level of abstraction to allow reasoning about their behavior across a broad range of applications. Programs that are members of a particular class can be implemented differently and the underlying numerical methods may change over time, but the claim is that the underlying patterns have persisted through generations of changes and will remain important into the future." [21]

This approach had initially the goal to measure state of the art multi-core architectures in terms of efficiency for a range of common problems. Its usefulness will be considered within the framework of IRMOS scope.

The term Dwarfs denotes a methodology developed by the parallel computing (PC) and high performance computing (HPC) community. In the initial PC/HPC context dwarfs are algorithmic methods that capture patterns of computation and communication that are bottlenecks in the application. For an application, the choice of a particular PC/HPC solution will depend on the presence of a particular dwarf in the application kernel.

Currently the term "Dwarf" seems to be used not in the sense of bottleneck but in the sense of a class of algorithms that tend to have a common pattern of communication and computation, and are well defined targets from algorithmic, software, and architecture standpoints [28]. Yet, we could not find a clear definition to serve us as guidance when identifying dwarfs in a given target domain, so in order to build some intuition next we list the currently identified dwarfs in a number of computational areas as given in [28]. In total across the different computational areas 13 dwarfs have been identified, which are as follows:

1. Finite State Machines
2. Combinational
3. Graph Traversal
4. Structured Grids
5. Dense Matrices
6. Sparse Matrices

[21] [http://view.eecs.berkeley.edu/wiki/Dwarfs]

7. Spectral (FFT)
8. Dynamic Programming
9. Particles
10. MapReduce model
11. Backtrack/ B&B
12. Graphical Models
13. Unstructured Grids

A subset of these 13 dwarfs will apply to any one individual area. In the context of IRMOS it is worth noting the areas of High-End Simulations in Physical Sciences which has parallels with the simulations done in COVISE (see Section 3.1), of Video-Games which has parallels with the virtual worlds scenario in eLearning, and the Desktop/Server area which has parallels with the post-production scenario:

- **High-end simulation in the physical sciences (PC/HPC):**
    1. Structured Grids (including locally structured grids, e.g. Adaptive Mesh Refinement)
    2. Unstructured Grids
    3. Fast Fourier Transform (FFT)
    4. Dense Linear Algebra
    5. Sparse Linear Algebra
    6. Particles
    7. Monte Carlo

- **Video Games:**
    1. Finite State Mach.
    2. Graph Traversal
    3. Structured Grid
    4. Dense Matrix
    5. Sparse Matrix
    6. Spectral (FFT)
    7. Particles
    8. Unstructured Grid

- **Desktop/Server:**
    1. Finite State Machines
    2. Graph Traversal
    3. Structured Grids
    4. Dense Matrices
    5. Sparse Matrices
    6. Particles
    7. MapReduce model

The dwarfs are building blocks of the current application and together with understanding of the algorithmic trends will aid the analysis and design of the future computational systems. The current uses of the dwarfs methodology revolves around identifying performance bottlenecks for improved design and benchmarking of computational architectures. How the benchmarks are to be used as higher levels applications performance indicators will not be a trivial problem to solve but a link is obviously present.

The dwarfs methodology seems to be a good candidate for getting indication of the complexity of an IRMOS application. First we will need to identify the exemplary algorithms, the dwarfs, for the problem domains of IRMOS, e.g. multimedia computing. Next, the complexity of each dwarf will need to be analytically or experimentally assessed. The presence of a dwarf in the kernel of an application will give an indication of the overall complexity of the application and together with the size of the input data will characterise the workload of the system. However, the practicality of this approach in the use cases identified for IRMOS is still being discussed.

A.5. Monitoring

The process of coordinating and observing resources and real-time services in a service-oriented environment in order to track their status for purposes such as fixing problems and tracking usage is a significant undertaking. This is due to the dynamic, distributed nature of the SOI environments and the high demands that real-time interactive applications impose. The Monitoring service and the Evaluator service provide links between the Infrastructure Provider infrastructure and the IRMOS platform and assist in the task of ensuring the Quality of Service for the execution of real-time interactive applications.

In the section that follows, we describe the Monitoring service architecture and behaviour that allows for efficiently monitoring resource usage and application service component states, receive updates on current status, and visualize monitoring results within the IRMOS framework.

The functionality of the monitoring process can be separated into two phases: (1) the Execution phase, and (2) the Benchmarking phase. In the following two sections we provide insights into the functionality that can be achieved via the use of the Monitoring service in the aforementioned phases.

A.5.1. Monitoring during Execution Phase

During the execution phase, the Monitoring service gathers information both about low-level performance parameters coming from the Infrastructure Provider infrastructure through its Metering service as well as information about high-level performance parameters coming from the Application Service Components that are being executed. All captured data will be filtered and forwarded to the Evaluator component which in turn is responsible for performing the following list of activities:

- The Evaluator contacts the SLA Manager to detect possible violations of the customer's SLA.
- Aggregates both high and low level information and sends it to the Mapping service for further processing. Resulting data will be stored into the History component (through the Mapping Service) which gathers historical data during the application execution so that more appropriate resources could be located in the future (e.g. when applying models, analysing the application behaviour and performance) as well as for statistical reasons.
- Having identified an SLA violation through the SLA Manager and/or a high risk for getting a performance anomaly, the Evaluator will contact the MAP tools to identify possible adjustments that could be made to the running application. For example, if a set of services fails to deliver what an application requires, then the MAP service could be used to analyse extra resources and necessary changes to the SLA.

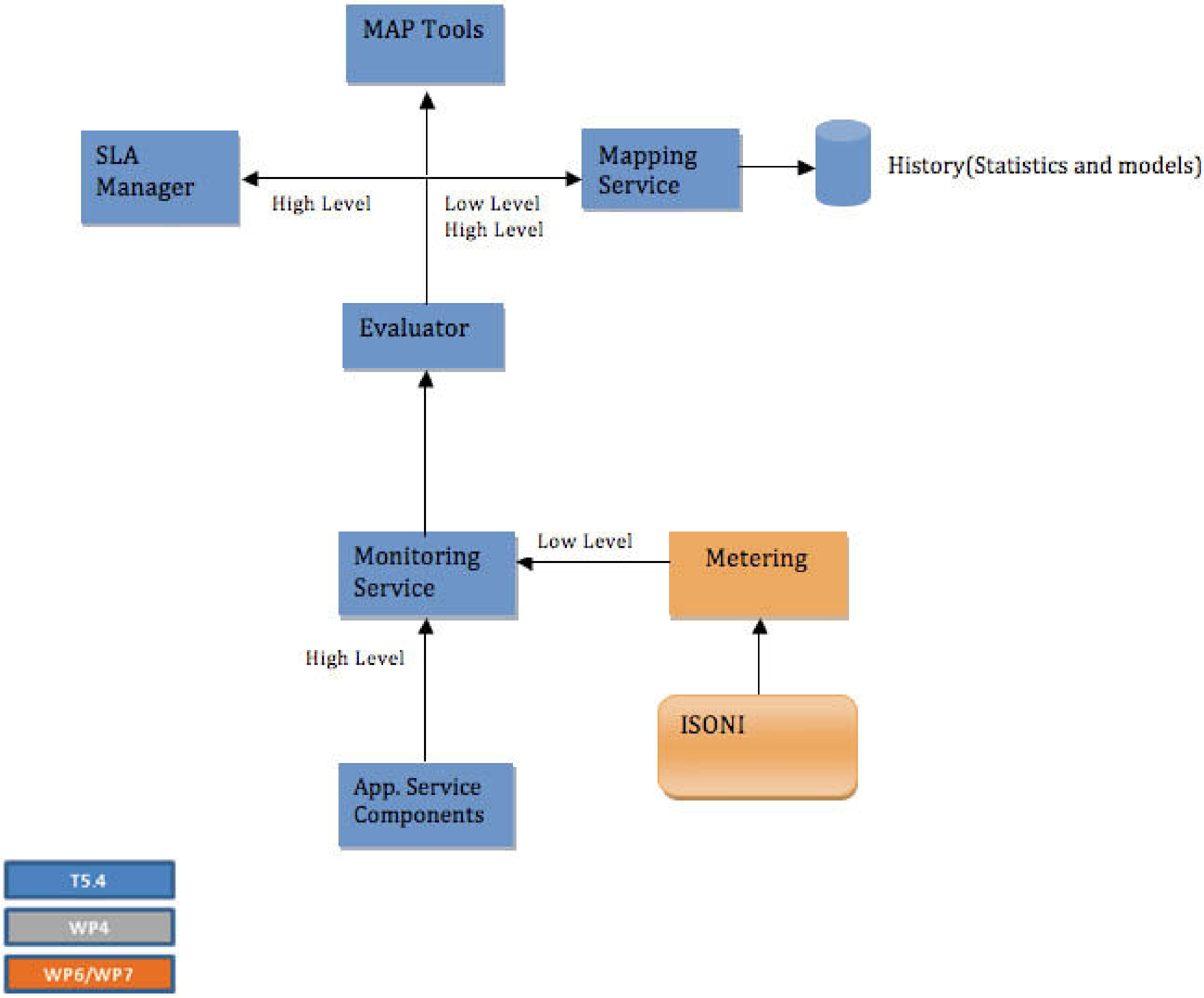


Figure 50: Monitoring Execution Phase

It should be noted that the SLA violations will be detected initially by the Infrastructure Provider (ISONI in the IRMOS case) regarding the low level parameters while the SLA Manager is responsible for analysing the high level parameters and detect SLA violations.

A.5.2. Monitoring during Benchmarking Phase

As the benchmarking is considered to be a special case of actual execution, the monitoring that takes place during the Benchmarking phase is similar with the Execution phase. As we can see at the following diagram, the monitoring data retrieved from the Metering and Application Service Components are forwarded to the Mapping Service and stored into the History and then to the MAP tools for further analysis. Since there are no performance constraints in this phase, the SLA Manager is no longer getting any information from the Evaluator, while all data are propagated to the Mapping Service. The benchmarking data will be afterwards used for calibrating the models (mapping application parameters to resource needs).

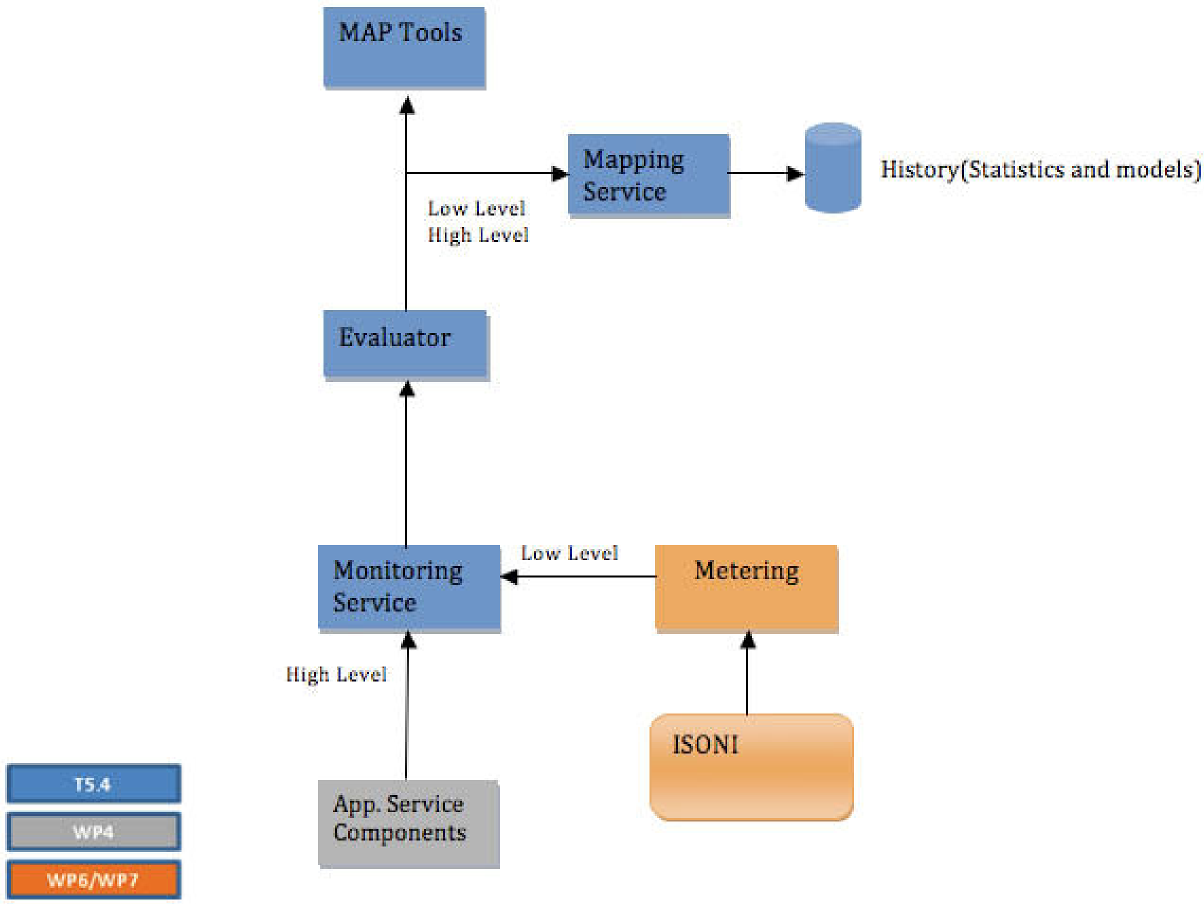


Figure 51: Monitoring Benchmarking Phase

A.5.3. Design details

The Monitoring Service will build heavily on capabilities provided by the WSRF and WS-N specifications which define the mechanisms used to describe information sources, access information via both queries and subscriptions, and manage information lifetimes. It will provide interfaces to different information sources, namely ISONI Metering Service and the various Application Service Components, translating their diverse schemas into appropriate XML schema for transmission over WSRF/WS-N protocols. Above the Monitoring Service, various tools and services can take advantage of its functionality to get insight into the execution of the application components inside the ISONI environment. Furthermore, due to the fact that performance is a critical issue for the any real-time execution process, a dedicated instance of the Monitoring Service will be deployed along with the VSN that supports the execution of the application over the ISONI.

A.6. Specification Languages

In the context of the IRMOS offering there are a set of things that need to be specified or modelled. This includes things such as the interfaces of services, orchestrations and QoS specifications at different points in the architecture. Specifications are usually performed using some kind of language.

In IRMOS WP5 will pursue the topic of specification languages from a Model Driven Engineering approach. This section provides a short introduction to languages in general in this context, and an overview of different existing specification languages that are relevant in the context of IRMOS. The section is ended by listing some of the identified needs for specification languages in IRMOS, and a description of how these relate.

A.6.1. Language basics

The context for the term "language" here is that of modelling languages, languages used to specify some abstraction (or model) of something. In the Model Driven Engineering field the development of languages is termed Language Engineering. Through language engineering one can develop languages that are specific to a certain business domain or technology area, such languages are called Domain Specific Languages.

The anatomy (or architecture) of such languages typically consists of two key elements: the *abstract* and the *concrete* syntax. The abstract syntax is usually defined using a model, since this model is actually a model of other models it is called a metamodel. The metamodel describes the different concepts of the language and the relationships between them.

The concrete syntax describes what syntactic elements constitute the language, or how the elements of the abstract syntax are to be presented or specified. One example is that in the Unified Modelling Language (UML) a box represents a Class. A language may have several concrete syntaxes for the same abstract syntax, for instance both a graphical (diagrams) and a textual notation.

The field of language engineering is not new, but in the later years more tool support has become available in order to make the creation of both abstract and concrete syntaxes and the connections between them simpler. Examples of such are the Eclipse Graphical Modelling Framework (GMF) and MetaEdit from MetaCase.

The Object Management Group, the standardisation organisation behind the UML, typically now standardise such specific languages asking for a metamodel and a UML profile representing the concrete syntax.

A.6.2. OMG Models of Real-time Systems

The Object Management Group (OMG) has several standards for languages relevant to the real-time aspects of IRMOS. These are *UML Profile for Schedulability, Performance and Time*, now to be replaced by *UML Profile for Modeling and Analysis of Real-time and Embedded Systems (MARTE)*, in addition *UML For Modeling Quality Of Service And Fault Tolerance Characteristics And Mechanisms*.

A.6.2.1. MARTE

As the name suggests the UML Profile for Modeling and Analysis of Real-time and Embedded Systems (MARTE), provides concepts and mechanisms for modelling and analysis of real-time and embedded systems. This is a significant area to cover, and the resulting standard is large. The challenge is to pick those parts of MARTE that are relevant to IRMOS, as much of MARTE is aimed at "hard" real-time aspects. Basically MARTE has tree main parts; Foundations, Design Model and Analysis Model.

MARTE defines the concept Non Functional Properties, while Functional Properties define the purpose of a system or application, the Non Functional Properties (NFP) are concerned with the systems fitness to the purpose. NFPs can include information about for example throughputs, delays, deadlines and memory usage. MARTE allows for the definition of such NFPs and the usage of them to annotate UML models. NFPs are separated into two main categories: qualitative or quantitative. The quantitative ones are measurable properties that may be specified using sample realisations and measures. The sample realisations are relevant for the benchmarking aspects discussed related to IRMOS mapping and analysis. A measure is a statistical function, hence it is related to the mathematical modelling presented in this document. Qualitative NFPs refer to characteristics that may not be measured directly. This is related to the different levels or classes that will be provided by the Infrastructure Provider. The NFP package also provides a set of probability distribution operations, that also are related to the more mathematically founded modelling.

NFPs are typically specified using the Value Specification Language (VSL). VSL expressions can be used to specify non-functional values, parameters, operations, and dependency between different values in a UML model. UML modellers can use VSL to specify non-functional constraints in their models.

MARTE includes a package for dealing with modelling of time, making the distinction between casual/temporal, clocked/synchronous and physical/real-time. The concept of time is naturally relevant to IRMOS.

MARTE offers support to model platforms for executing real-time applications. Different types of resources are defined, among them StorageResource and ComputingResource are the ones immediately relevant to IRMOS. Support is also provided to define Resource Managers and Resource brokers. These are relevant to IRMOS albeit possibly only in the internals of an Infrastructure Provider.

In the Design area MARTE provides the concept of a General component. It extends the UML Structured Classifier (the typical component representation in UML) with concepts relevant to real-time such as Flow Ports that allow for flow-oriented communication between components. Related to IRMOS the General Component concept matches that of the IRMOS Service and Application components.

MARTE also provides detailed support for modelling software and hardware resources. In IRMOS the hardware is virtualised, but elements of MARTE in this area may be relevant in the definition of SLA specification languages.

The third part of MARTE is about analysis. MARTE analysis is intended to support accurate and trustworthy evaluations using formal quantitative analyses based on sound mathematical models. It provides a Generic Quantitative Analysis model and specifies two sub-profiles; one for schedulability analysis and one for performance analysis. In general the quantitative analysis techniques determine the values of output NFPs based on data provided as input NFPs. Key terms are Workload Behaviour that contains a set of end-to-end system level operations with defined behaviour and Resources Platform that is a container for the resources used by the behaviour mentioned. A Workload behaviour contains scenarios each constituted by a set of steps. Performance Analysis Modelling describes the analysis of temporal properties of best effort and soft-real-time systems including multimedia and networked services. The analysis part of MARTE is relevant to IRMOS for performing analysis based on models using the mathematical models presented in this document.

In order to provide the needed expressiveness in models MARTE has defined a set of normative model libraries that extend the basic support in areas such as time.

A.6.2.2. UML For Modeling Quality Of Service And Fault Tolerance Characteristics And Mechanisms

This standard provides a metmodel and UML profile for the definition of Quality of Service and Fault tolerance. It is more limited in scope than MARTE, and relates to the NFP part of MARTE.

The key concept of QoSFTC are Quality Characteristics and Quality Levels. A quality characteristic is a set of quality attributes that are the dimensions along which to describe quality satisfaction. A quality level describes the quantifiable level of satisfaction to a NFP. The concept QoS Constraint is used to define restrictions on characteristics. Figure 52 shows a simplified version of the metamodel for QoSFTC.

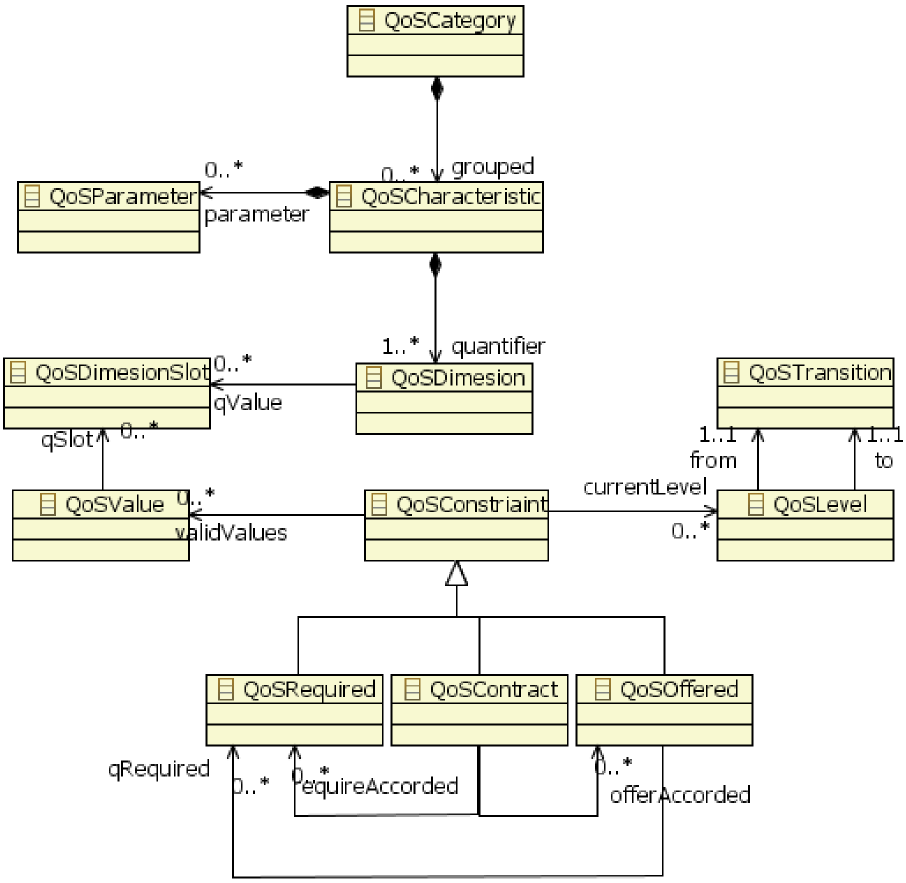


Figure 52: Simplified metamodel

QoSFTC includes a set of predefined QoS Characteristics and Categories that are independent of problem domains, named the QoS Catalogue. This catalogue covers the areas: Performance, Dependability, Security, Integrity, Coherence, Throughput, Latency, Efficiency, Demand, Reliability and Availability. All providing details of what aspects of the area are relevant to model.

The QoSFTC provides a limited set of essential concepts related to the definition of QoS. Most of the basic concepts are relevant to IRMOS specification languages.

A.6.2.3. OMG Models of Service Oriented Architectures

The OMG are currently working on a standard that relates to the other key aspect of IRMOS; Service Oriented Architectures. The standard is called *UML Profile and Metamodel for Services* (UPMS), and is currently nearing finalisation. As the final details of this standard is not available yet the summary in this section only concentrates on the main concepts.

A ServicesArchitecture is a network of participant roles providing and consuming services to fulfil a purpose. A Role defines the basic function that an entity may perform in the given context. A Services Architecture is denoted using a UML Collaboration Use.

A key concept of a Service is the Service Contract. A Service Contract defines the terms, conditions, interfaces and choreography that interacting participant must agree to. Each role in the Service Contract is defined by a Service Interface. A Service Contract choreography defines how the involved party are to interact, and is specified using a UML behaviour such as a sequence diagram.

QoS issues are not dealt with specifically by this standard, it will therefore be relevant for further work in IRMOS to look at how mechanisms from MARTE and QoSFTC can be applied to models using the UPMS profile. As all are profiles of UML this seems feasible.

A.6.3. Business Process Execution Language for Web Services (WS-BPEL)

The Web Services Business Process Execution Language (WS-BPEL) [51] is a language to model Web service based business processes. The core concept is the representation of peer-to-peer interactions between a process and its partners using Web services and an XML-based grammar. It is built on top of WSDL (both the processes and partners are modelled as WSDL services).

WS-BPEL – BPEL for short – is a language based on XML that allows for controlling the process flow (states, coordination and exceptions handling) of a set of collaborating Web services. For that, it defines interactions that exist within and between organisation processes. The language uses either a graph based or algebraic representation, and offers the ability to manage both *abstract* and *executable* processes. It provides constructs to handle long running transactions (LRTs), compensation and exception using related standards WS-AtomicTransaction, WS-BusinessActivity and WS-Coordination.

BPEL offers an interesting feature that allows having an independent representation of the interactions between the partners. The interaction protocols are called *abstract* processes, and they are specified in *business protocols*. This concept separates the external behaviour of the partners (public and visible message exchange behaviour) from their private internal behaviour and implementation. *Executable* processes are represented using the BPEL meta-model to model the actual behaviour using the three classical flows: the control flow, the data flow and the transactional flow. It also includes support for the message flow.

As in traditional flow models, the control flow defines the execution flow as a directed acyclic graph. The language is designed to combine the block oriented notation and the graph oriented notation. It contains powerful constructors for modeling *structured activities*: aggregation, branching, concurrency, loops, exceptions, compensations, and time constraints. *Links* are used to define control dependencies between two block definitions: a source activity and a target activity. Activities can be grouped within a *scope*, and associated with a scope are three types of handlers: *fault handlers*, *compensation handlers*, and *event handlers*. When an error occurs, the normal processing is terminated and the control is transferred to the corresponding fault handler. Then, a process is terminated when it completes normally, when a terminate activity is called (abnormal termination), when a fault reaches the process scope or when a compensation handler is called.

BPEL *basic activities* are handled by three types of messages: <invoke> to invoke an operation on a partner, <receive> to receive an invocation from a partner and <reply> to send reply message in partner invocation. For each message, one must associate a *partner*, which prohibits the message exchange between two internal components for instance. Furthermore, there is no ability to associate a timeout to the <invoke> activity. This can block the system if no response is returned.

Data flow management is ensured using scoped variables. Input and output of activities are kept in variables, and data is transferred between two (or more) activities thanks to shared data spaces that are persistent across Web services and global to one scope. The <assign> activity is used to copy data from one variable to another.

BPEL also proposes a compensation protocol to handle the transaction flow, and particularly long running transactions. One can define either a fault handler or a compensation handler. Handlers are associated with a scope, and a fault handler defines alternate execution paths within the scope, while the compensation handler is used to reverse the work performed by an already completed scope.

On collaboration aspects, BPEL is able to model several types of inter-actions from simple stateless interactions to stateful, long running, and asynchronous interactions. *Partner Link Types* are used to model partner relationships and *correlation sets* represent the conversations, maintaining the state of the interaction. The choreography of the collaborative business processes is defined as an *abstract* process.

A.6.4. Service composition

Web service composition refers to the development of new Web services by interconnecting existing ones according to some business logic, expressed (for example) as a business process model. For example, a composite Web service for travel arrangement could bring together a number of Web services for flight booking, accommodation booking, attractions search, car rental, events booking, etc. in order to provide a “one-stop shop” for its users. Web service composition is a key element of the Web services paradigm, as it provides a means to integrate heterogeneous enterprise applications and to realize business-to-business collaborations.

Orchestration deals with implementation management (what happens behind interfaces, i.e. process execution). This means that orchestration is a private process, controlled by one party, and it defines steps of an executable workflow. Propositions such BPEL and BPML are clearly at this level. Choreography is more about what happens between interfaces. It can involve static or dynamically negotiated protocols. In this sense, choreography is a public abstract process, where conversations are made up of equals, and they define sequences of observable messages [Peltz 2003].

Visual Service Composition Studio

The Visual Service Composition Studio (VSC) has been developed in the SODIUM project (IST-FP6-004559) by SINTEF. It is a toolkit for creating, editing, storing, and loading service compositions. It uses a variant of UML 2 activity diagrams, which is called the Visual Service Composition Language (VSCL), to describe compositions. The main difference is that VSCL introduces a special transformation node, which is not found in UML2. VSCL includes the following model elements:

- Task nodes which are similar to action nodes in UML and represent a service operation.
- Initial nodes, which define the start of the execution path.
- Final nodes, which define the end of the execution path.
- Control flow, which represents a step in the execution path.
- Object flow, which represents a step in data path, i.e. the path that a data object may take.
- Transformation nodes, which transform data objects from one type to another.
- Input and output pins, which represent the input and output parameters of a task.
- Decision nodes, which represent a choice in the execution path. A decision node may have several outgoing control flows, and the execution path is decided by guard conditions on these control flows.
- Merge nodes, which join the alternative execution paths.
- Fork nodes, which split a single execution paths in to several parallel paths.
- Join nodes, which combine and synchronize parallel execution paths.

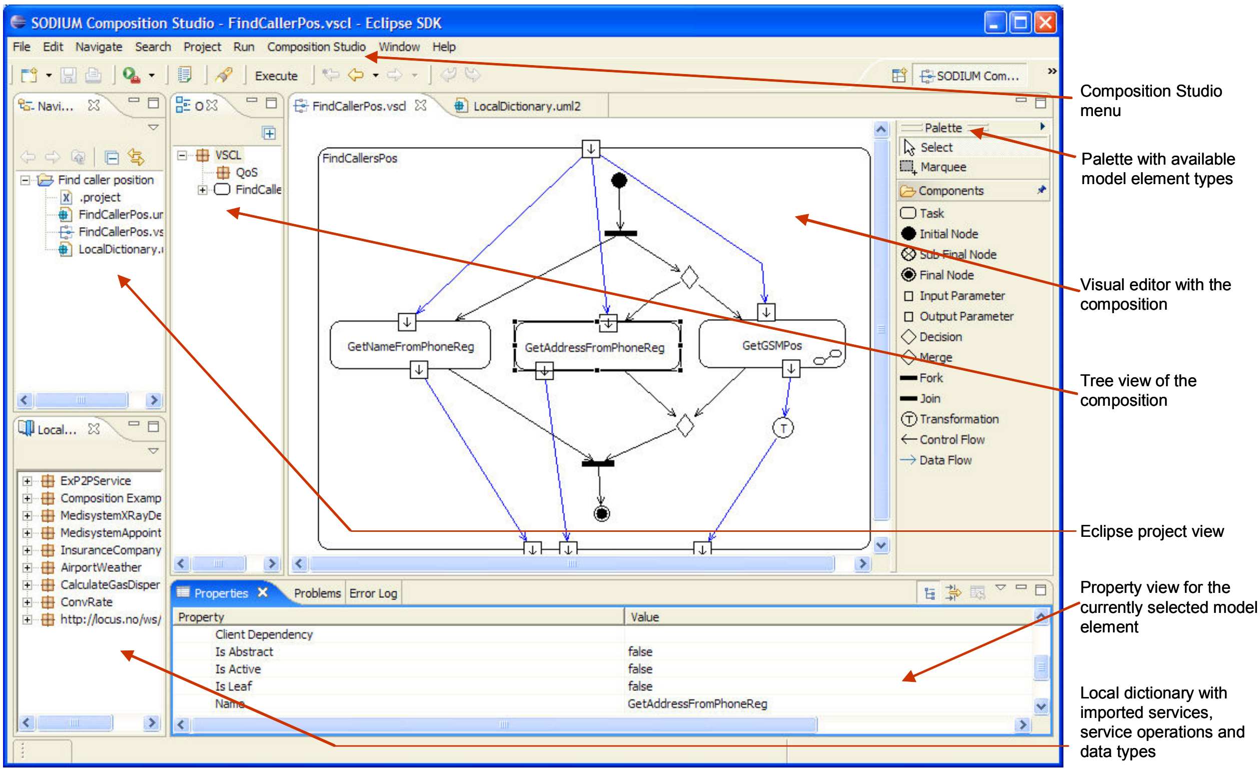


Figure 53: The Visual Service Composition Studio

The figure depicts the graphical user interface of the Visual Service Composition Studio. The main components are:

- The **project view,** which lists the projects and compositions that a user is working on. Eclipse is projected oriented and a composition is created within one project. Hence, the user must create a project before creating a composition. The project view offers also standard functionality for deleting, renaming, and moving a project.

- The **palette**, which contains the model constructs from the Service Composition Language.

- The **visual editor** where the service composition is defined. The user creates a composition by selecting model constructs from palette and putting these inside the composition.

- The **local dictionary,** which is a repository for storing service descriptions. The user may import service descriptions in WSDL, and OWL format. A service operation in the dictionary can be added to composition by selecting the operation and drop it onto the visual editor. A task will then be created with the selected service operation.

- The property view, which allows editing of properties of the currently selected model element.

Future work in WP will look into whether the Visual Composition studio can be further developed to be used as part of the IRMOS tool offering. At the current point in time discussions have taken place and it seems feasible that the tool can be the basis for defining compositions of Service Components into either new service components or possibly application components. The tool will then be extended to make use of IRMOS Service Component Descriptors as input, and already is capable of creating BPEL as output.

A.6.5. The different languages of IRMOS

In the IRMOS solution there will be a number of different things that are to be specified using some mechanism or language. The following presents a non exhaustive list of the specifications identified

- Application Component Specifications
- Service Component Specifications
- Workflow Specifications at different levels
- SLA Templates
- SLA Specifications
- VSN Descriptors

It is important to understand the relationships between the different languages, especially from a MDE point of view as one possibly will need to provide transformations between selections of the languages. This topic will be pursued further in subsequent work and deliverables from WP5, but the following section provides an outline of an initial model to understand how some of the descriptions in IRMOS are related.

A.6.5.1. Concept model outline

One of the goals of IRMOS is that services should be able to provide an agreed quality of service. This agreement is defined by a Service Level Agreement (SLA). In modelling of QoS using UML one has typically annotated the model it self with QoS values (including offers and agreements), in IRMOS many of these parameters will vary depending on what virtual hardware the service is deployed to and what virtual network is set up for the service. This in turn will be defined by SLAs and will therefore not be defined in a general model of the services. It is however necessary for the different services to define what QoS parameters they can handle, and how these can be monitored by the outside world.

Figure 29 shows an outline of a model of the key concepts related to the IRMOS components at design time, where Client Components and Service Components form an Application component that provides useful functionality to a user. Service Components have a SLA template that indicates what parameters can be part of an agreement of service. On the technical side a Service Component has a description of its interface(s) and a defined list of monitorable parameters. Such parameters are a representation of possibly application specific indicators of the performance of a Service Component, for instance frames per second for a transcoding service. During runtime these parameters can be used to monitor performance toward the defined SLA for the current execution. It is probable that most of the monitorable parameters will be part of the SLA template.

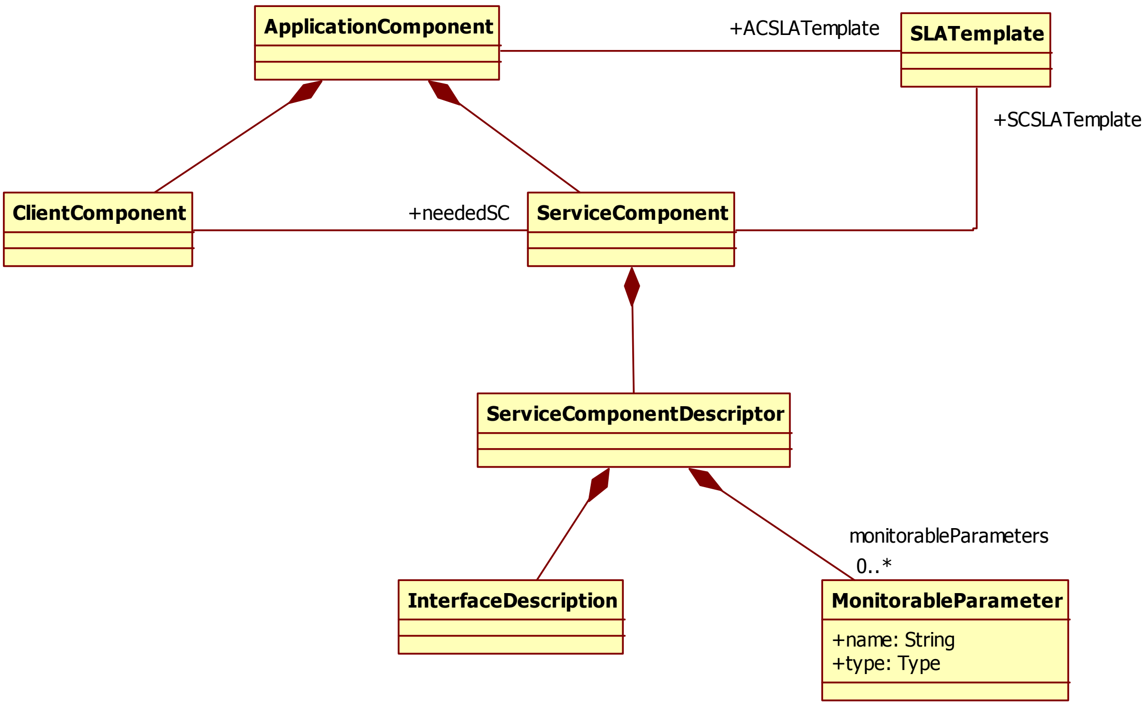


Figure 54: Intial outline of concept model

The reason for mentioning this here is that IRMOS will have to provide mechanisms (in form of a language) to enable service component developers to provide the Service Component Descriptors. This initial metamodel can be seen as a first step in the direction of an abstract syntax for this language. The metamodel will be refined and extended in subsequent WP5 deliverables.

Annex B.

B.1. Curve Fitting from Benchmarking Data

In the context of the SOA paradigm, a number of resources is offered as a service. These resources are offered by the ISONI provider and are available to all the requesting users. But in order to provide strong guarantees of the QoS levels throughout the infrastructure, the IRMOS platform performs advance reservation on the required resources. Therefore, in order for an application to be executed successfully, at least in terms of finishing in time and not violating the amount or duration of the requested infrastructures, an estimation of the resources needed by this application must be provided beforehand in order to be used during the SLA negotiations for performing advance reservation.

A number of methodologies is often used when the software code is available. These include the analysis of the code segments and hardware infrastructure for the determination of the algorithmic complexity [29]. However in the case of SOIs, the source code will probably not be available for a number of reasons, mainly licensing. Also the infrastructure on which this code will be executed is not known a priori and is based on the provider's scheduling processes and resource utilization methods. Even if a number of hardware parameters is set, such as CPU type, there are many parameters such as for example graphics cards, cpu utilization etc. that cannot be foreseen or guaranteed.

In some cases it may only be possible to estimate mapping through benchmarking. That is there is no understanding of an application component as a set of modules, or there is no information on the behavioural function or complexity for each of the modules. Instead all that is available is one or more software binaries that can be run with different parameter values. In this case the only possible technique for mapping high level application to low level parameters is through benchmarking of the binaries and then attempting to discover appropriate mapping functions through curve fitting.

So the only way to have information on the resource requirements of a component is either from test executions (benchmarking) or real life historical data and the use of metrics that are generic, such as CPU cycles. Due to the fact that for historical data a time period needs to exist where the performance will not be desirable or there will be a need for significant over-provisioning, we have decided to include both approaches. Benchmarking is used in order to obtain data that are absolutely needed for the modelling of the Service Component while historical data are later on used for the improvement of the model and for a better determination of the probability of the estimation to be inaccurate.

B.2. Curve fitting techniques through data point approximation functions

In curve fitting techniques the basic approach is that we have a number of experimental data (points) and we need to find a function that satisfies them, meaning that the curve presented by the function will pass from these points in the graph (exactly or very near). Then by having this function we can estimate the points for which the output is not known, for inputs that are not part of the experimental process. A number of curve fitting techniques exist [40], and a number of them are introduced here. There are two main categories, the ones that satisfy the experimental data points, meaning that the resulting curve includes them and the ones where the resulting curve passes very near from them. We will focus more on the second category, since the first works well for points near the experimental data but quite poorly overall.

In this methodology the function is assumed at the beginning (being linear, exponential, logarithmic etc.) but the main difference from the above is that it does not need to satisfy the given benchmark data. The main criterion is the minimization of the error inserted during the function assumption. This error is usually the LSE, meaning the minimization of the sum of least squares:
LSE: $S=[y_1-f(x_1)]^2+\ldots+[y_n-f(x_n)]^2$, where *f* is the assumed function and *i=1…n* the experimental data points.

In the linear approximation, a function of the form *y=ax+b* is used and calculated through the least squares method in order to reduce the distance from the data points.

In this process, if we have *n* data points we use as a criterion the minimization of the sum of squares:

$$S=[y_1-(a+bx_1)]^2+\ldots+[y_n-(a+bx_n)]^2$$

Where x_i, y_i are the known data points and *a, b* the parameters of the line that we want to determine.

In order to minimize the latter we have the equations:

$\frac{\partial S}{\partial a}=0$ and $\frac{\partial S}{\partial b}=0$ from which we derive the following:

$$na+(\sum_1^n x_i)b=\sum_1^n y_i$$

$$(\sum_1^n x_i)a+(\sum_1^n x_i^2)b=\sum_1^n x_i y_i$$

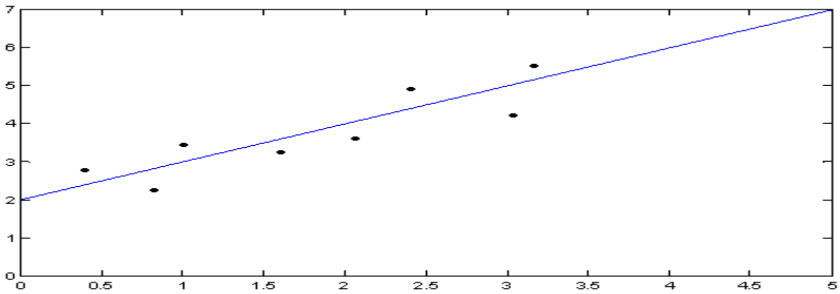


Figure 55: Example of linear approximation

This method is of course good only for linear cases, in other circumstances the error is severe. But the same approach can be used with other kinds of functions such as polynomial, logarithmic, exponential etc which may not be linear but may be transformed into such.

For non linear cases (such as multiple degree polynomials) the method is the same but with the difference that the final equations are more complex. We will also experiment on the use of multivariable regression techniques, since we expect that there will be cases where multiple parameters will influence the behaviour of the SC. For some of the above methods the increase in dimensions comes with increase in the complexity of the equations that need to be solved.

One other method that can scale better in terms of input parameters is the factor analysis. Through this process we determine a combination of the input parameters and their according factors so as to form the behavioural function of the ASC that can later on be used for estimation.

There is the possibility of implementing a number of these methodologies and then through evaluation of the results to decide which one is the fittest and use this for the according service component. This means that for every SC we run the mapping service based on the data set, calculate the error of the estimation and then choose the best curve fitting technique. This is based on the fact that each technique has a number of functions or behaviours to which it fits more accurately than others. So with this dynamic approach we can switch methodologies according to each SC's nature. Also each of the above methods comes with a number of improvements, which will also be investigated. The models derived from the mapping service through this process can afterwards be used by the MAP Tools for the optimization of the workflow.

An initial experiment that we have conducted is based on an application that will be most probably met in IRMOS. It is the encoding of an uncompressed video (with no audio) to mp4 format. For this test we have created a number of video samples, taking as parameters the duration of the video, the frames per second used for the capture, the resolution of the captured image and an index regarding the motion inside the sample (still image, semi moving and extreme moving). These are all independent parameters with no correlation between them. As outputs we extracted the execution time taken for transforming the original video to mp4 format with the use of the ffmpeg algorithm and the storage size of the original video. The former was intended to be used as a crude metric for computational resources and the latter for storage resources that would be needed to be reserved if this application was a part of IRMOS. All the tests were executed on the same computer with similar percentage of CPU usage. However, the unexpected result came from observing the size of the original video and the regarding execution time needed which could be directly correlated. What we discovered actually was that the four input parameters influence could be summarized in the storage parameter, thus converting a multivariable problem into a single variable one. For this case we applied a number of basic curve fitting methods which were mentioned above and appear in the following figure.

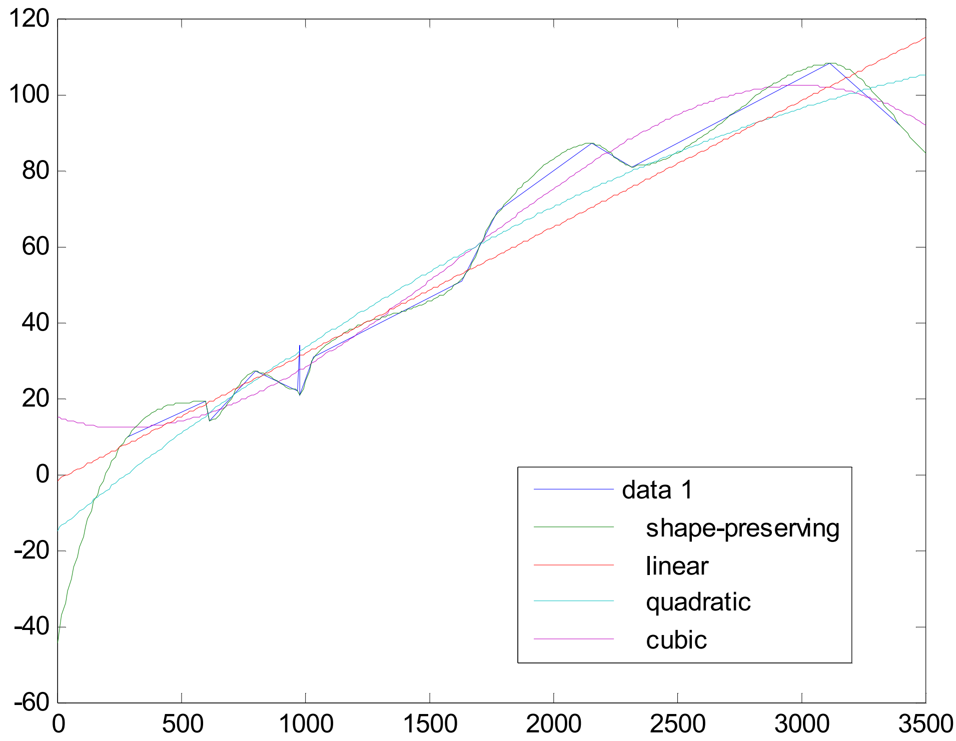


Figure 56: Approximating curves for the FFmpeg example

For the measurement of the performance of each method we used a separate data set with dispersed values in the variable space, but in all cases inside the investigated interval (interpolation). For these cases we derived the following results for a variety of curve fitting and approximation methods:

Method	Mean error of estimation (%)	σ	Max error of estimation (%)
Linear	20.7	0.161	41
Quadratic	14	0.0827	24.7
Cubic	10.6	0.0711	21.5

4th degree polynomial	13.2	0.1058	30
5th degree polynomial	11.9	0.0824	24.6
6th degree polynomial	15.1	0.1727	40
7th degree polynomial	21.2	0.085	31.7
8th degree polynomial	19.9	0.06	24.6
9th degree polynomial	17.2	0.104	26.9
Shape preserving interpolant	15.8	0.1745	46

Table 15. Comparative results of methods for the approximation of ffmpeg algorithm

From this table we can conclude that the best case was the cubic approximation which gave an average estimation error of 10.6%. Of course this does not need to be the same for all applications. That is why a number of these methods may be used in order to investigate which is best for each application in order to describe its behaviour.

With the use of PDFs we can remove a part of the uncertainty of the estimation and with the use of confidence intervals to make sure that the application will not run out of resources. The higher the probability needed for this will lead to greater amount of resources reserved. There is of course a trade-off in this approach. If we reserve a large amount of resources that will probably not be of use will increase the probability of the task to complete in time but with additional cost for the customer. But this is also a decision for the business model. One provider may choose to offer lower cost at the expense of quality and another the opposite. Of course this is also affected by the quality of the estimation. The higher this is, the lower will the need for over-provisioning be.

One problem of these methods is the noise inserted in the samples. In the original curve some spikes can be observed that do not seem to follow the tendency of the measurements. For this reason we can apply methods for data pre-processing that in general are considered as a way to remove the noise from the input or at least minimize its influence. In this approach we remove the points from the data set that can be regarded as outliers, meaning that they are more than three standard deviations away from the mean. After the implementation of this method we observed an improvement in the linear estimation, which had a mean error of 15.2 % (in comparison to the original 20.7%) and a standard deviation of 0.13 (in comparison to the original 0.16).

Based on the LSE method, we can test whatever form of function (linear or non linear) to the experimental data set and investigate the precision of the according estimates. This means that the originally assumed function can include logarithmic, exponential or any kind of terms. This was investigated with the help of [41], where a large number of methods is implemented and is used in order to find the best approximation in the data set provided. There is also the implementation of a general form of function with different kinds of terms and is searched each time for the best fit. The results of the tests made in this case were almost identical to the cubic approximation mentioned above (10% mean estimation error and 0.07 standard deviation).

B.3. Multivariable methods

We have also explored the potential of multivariable methods for estimation. The most basic is multi-linear regression. In this case we assume that the response of y is a linear combination of the values of the predictors x_i,

$$y = a_1 x_1 + a_2 x_2 + \ldots + a_n x_n + c$$

In our case the predictors were the variables of the experiment (duration, resolution, fps and index of movement) and y the estimated time of completion. Through the observations made during the experiment the coefficients of this model were calculated. This approach did not give satisfactory results due to its simplistic nature. Indicatively, the mean error was about 30% but with large maximum error (nearly 50%). The encouraging feature was that in all cases there was an overestimation of resources, a fact that leads to poor utilization but no execution failures due to under-provisioning.

One more complicated approach that is interesting to investigate for the case of multivariable estimation, is a model estimation of the form:

$$y = a_1 f_1(x_1) + a_2 f_2(x_2) + \ldots + a_n f_n(x_n)$$

Where $x_{1,} x_2, \ldots x_n$ are the independent variables from which y depends on. Then by altering one variable and keeping the others constant, we can create a data set that will aid us in choosing each of the individual functions $f_{1,} f_2, .. f_n$ from a wide variety of known forms (linear, exponential, logarithmic etc.) . With this method we can find the dependence form of y from each input variable separately. By trying to fit the data on the entire function we then find the values of the coefficients $a_1, a_2, \ldots a_n$ for the final estimation. The performance of this method will be explored in the following project months.

Annex C.

C.1. Reactive Behavioural Models

Modelling real-time applications requires models that are capable of describing how an event triggered in one part of the application causes changes in behaviour in a different part. This kind of model is necessary when modification of a resource constraint in some part of the system changes the system behaviour. This is the case where an application is composed of separate activities, each of which represents a logical part of the application that can be thought of as independent. A model that describes how events can be triggered by the activities and how behaviour changes as a result of these events is known as a reactive behavioural model. This type of model is appropriate for reasoning about the case study in Section 4. For example, in the scenario the telecine device may reset to an earlier point if bandwidth becomes too low. During that time no events are triggered by the telecine device. Therefore the telecine device is no longer transmitting to the data link and that may mean the bandwidth available improves during that time. Consequently there is some chance that other activities that were causing the available bandwidth to drop may complete by the time the telecine device tries to transmit again. Therefore the resetting action of the telecine may inadvertently free up bandwidth by the time the telecine tries retransmission and permit the remaining transmission to complete successfully.

The following sections will define the formal semantics for reactive behavioural models that are appropriate for reasoning about IRMOS applications. The models are suitable for estimating performance characteristics and verifying properties based on stochastic knowledge of input data, potential interaction events, and benchmarking data.

The semantics and models that are derived have the following significant characteristics

- the application model is composed of concurrent component models
- stochastic variation in event occurrences are encapsulated solely in the activity models
- provided that the temporal profile for the component models are valid then the application models are guaranteed deadlock free
- analytic verification and performance estimates can be automatically derived through stochastic model checking

C.2. State of the art for analytic reactive models

The general trends in the state of the art for time constrained reactive models and system specification have been surveyed in deliverable D2.3.1 [5]. This section picks up on those topics that are particularly relevant to the behavioural reactive models necessary for the IRMOS project, specifically automata theoretic approaches. This section will cover in more depth the foundations of these topics since they bear a particular relevance to IRMOS. Much of the current research in the area is focussed on protocol modelling and verification [8], [9]. It is not a straightforward exercise to use such approaches for IRMOS as they do not directly lend themselves to modelling reactive systems that are characterised by performance benchmarks as opposed to the event response nature typical of a protocol specification. Benchmarks measure time and space complexity characteristics of batch jobs, and tend to assume uninterrupted continuous runs. Whereas, a reactive model describes changes in behaviour caused by external events.

Other approaches that focus on performance evaluation and worst case execution time estimation tend to focus on complete system specifications that require very detailed knowledge of all aspects of an application, including hardware specific details of any platforms where the application will be deployed and of the whole OS stack that will be used on such a platform. Since IRMOS services are virtualised such knowledge is rarely possible and may vary from one deployment to the next. The only invariants for these services are defined by QoS values in SLAs. This subject is discussed further in Section A.4.1

The primary method for estimating time and resource performance with a reactive model is to experiment with questions of the form: 'if I am constrained to use only these resources for this activity when will the activity finish'. A successful reactive model can automatically answer such questions directly. A constraint on a resource can include hard and soft time constraints, such as this activity can not start before time t0, as well as constraints referring to more obvious constraints such as network bandwidth or processing power. When a model can automatically answer such questions a user can then experiment with different levels of resource to discover what choices they have within the time frame available or if the time frame is unrealistic and has to be adapted. Thus to discover what bandwidth in one activity will permit an application to complete in a specific time frame a user would usually run the model with a variety of levels of bandwidth until a suitable value is found.

When eliciting requirements for applications that are to be adapted for an IRMOS platform the exact nature of the infrastructure communication protocols may not be fixed since the applications will be virtualised. At the initial stages of requirements elicitation much of the application may only be specified at a coarse level of granularity. For example in the scenario we defined in Section 4.1 there is a telecine device that will ingest film content and stream the output to a data link. In the scenario, if the bandwidth is not high enough for some percentage of the time then the telecine crashes and has to be reset. When modelling the scenario it is not initially important to model how the data link works, what matters is the consequence of a lack of bandwidth on the telecine. Hence, for investigating what temporal performance is possible for the telecine it is important to first model how the completion time will vary with respect to the percentage of time bandwidth is sufficient. That is for each unit of time the bandwidth is either acceptable or not as far as the telecine is concerned, and it is not necessary to know the exact amount of bandwidth. Later, once bandwidth temporal constraints are captured, it will become possible to determine what performance characteristics of the data link will ensure bandwidth will satisfy those temporal constraints.

The rest of this section considers discrete time probabilistic finite state automata. The section concludes with a recommendation for using the PRISM tool as a backend analytical verification and performance estimation tool.

C.2.1. Discrete time stochastic finite state automata

This section discusses discrete time stochastic finite state automata. This formalism is one that can be analytically investigated using the PRISM model checking tool [18], [9].

For discrete time it is assumed clocks have distinct time periods that are indexed by the natural numbers ***N***. Each of these periods for any given clock has exactly the same length so that all measurements can be regarded as counting clock ticks. Often it is assumed that ticks for all clocks are multiples of some tick given by a single global clock. For each clock there is a variable that holds the current tick value.

For a set of clocks K, a function v : K → ***N*** is a clock valuation. 0 is the clock valuation that resets all clocks to time 0. A clock constraint is a conjunct of primitive constraints of the form (x >= c), (x <=c) and (x=c) where x represents the clock tick for a clock and c is a constant natural number. Let KC be the set of all possible constraints on K. A probabilistic timed automaton *A* consists of the following

- a finite set L of locations
- an initial location *l* in L
- a finite set of clocks K
- a finite set of events E, and a subset U of E that are defined as urgent
- an invariant function *inv* : L → KC
- a finite set of stochastic edge relations *prob* that is a subset of L x KC x E x D, and D is the set of probability distributions on pairs (X, y) where X is a set of clocks and y is a location in L.

A state in *A* is a pair (y, v) where y is a location and v is a clock valuation that satisfies the invariants *inv*. A probabilistic edge e in *A* is a tuple (y, kc, V, p), where y is a location, kc are clock constraints, V a set of events and p assigns a probability to each pair (X,y) consisting of a set of clocks and a location. From (u,v) the automata can choose (nondeterministically) to take one of two choices, either to allow time to pass or to move on to a new state. Moving to a new state will result in resetting clocks to 0. Allowing time to pass to some new set of clock ticks is only allowed if after the time period has finished *inv* is still satisfied by the new clock valuation. The edge (y, kc, V, p) is enabled when kc satisfy the current clock valuation v. A state (z, 0) is a possible destination state from (y,v) with probability p(X,z) where X is the set of clocks that have none zero valuation according to v. If there is an edge from state (y,v) that contains urgent events then the option of letting time pass is prohibited.

In the definition for automata we did not use the events along a probabilistic edge to define possible next states except where urgent events are present. The complete set of events on an edge are used to control synchronised parallel composition of automata in an analogous manner to synchronised parallel composition of process algebra terms. For two automata $A_{i} = (L_i, \underline{l_i}, K_i, E_i, inv_i, prob_i)$, where i is 1 or 2, the parallel composition of A_1 and A_2 is denote $A_1 \| A_2$ consisting of a tuple (L_1 x L_1, $\underline{l_i}$ x $\underline{2_i}$, K_1 x K_2, E_1 x E_2, *inv*, *prob*). The invariant *inv* is the conjunct of the respective invariants at each state and the probability distribution *prob* is defined so that synchronous transitions must occur when the respective edges from A_1 and A_2 contain common events. In combining automata in this product there is nondeterminism between the choice of transition distributions for each automata. This is resolved by either choosing the minimum or maximum probability for each transition. Hence, verification of properties over a system are with respect to one of these two distributions. The exact details of these distributions are given in [9].

Given a system comprised of the composition of automata $S = A_1 \| A_2 \| \ldots \| A_n$ it is possible to automate model checking for temporal logic properties of the system providing that the system is only finite branching. Probabilistic reachability for S has been studied in [25], [26].

C.2.1.1. Pros and cons for IRMOS

For IRMOS the pros and cons of using discrete time probabilistic finite state automata are as follows.

- Pros
    - Time constraints can be defined with respect to discrete clocks
    - Some parallel composition is possible between automata
    - Model checking can be automated when the model is guaranteed finite branching
    - The PRISM process algebra has a semantics in terms of probabilistic finite state automata, which means temporal logics can be used to reason about them with the PRISM tool.
- Cons
    - Timing constraints must not compare the values of clocks with one another or contain strict comparisons with constants. This is a significant disadvantage for IRMOS where strict comparisons will be necessary.
    - Parallel composition can only be defined at the top level of system specification.

C.2.2. Evaluation of PRISM Modelling Technology

The PRISM tool provides a superset of functionality available in the PEPA toolset. In this section we only consider PRISM. The PRISM tool is useful in investigating hard timeframes when models are defined as Markov decision processes (MDPs). PRISM does allow discrete time and continuous time models but these will not be discussed here. The purpose of this section is to give an overview of the tool and an insight into the type of MDP models that are feasible.

```
global tm : [0..bd] init 0; // variable used to count clock ticks
module Clock
  st : [0..2] init 0;
 [tick] st = 0 -> st' =1; // clock ticks
 [] (st=1)& (tm<= bd-1) -> (st'=0) &(tm'=tm+1);
 //stop the clock when hit the upper bound for the model
 [] (st=1)& (tm = bd) -> (st' = 2);
endmodule
```

Figure 57: Simple module example from PRISM

Figure 57 gives a trivial example of a PRISM system definition consisting of one module that defines a discrete clock and that does not contain any probabilities, that is all transitions occur with probability 1. This document is not intended as a PRISM manual and the reader is referred to the website [18] for full details. However we do need to give enough understanding of the notation in order for the models explored in this document to make sense. Each command of the PRISM model is of the form

[synchronous event label] (guard) → probabilistic update of state variables

A synchronous event label is just a string and when null this represents that the transition occurs independently of other processes, or nondeterministically. An event label only effects concurrent composition when it is used to synchronise commands with the same event label. The guard is an expression that can evaluate values of variables within a general logical expression. The transition once triggered results in modifying variables according to the expressions on the right hand side of the →. The notation in PRISM uses v' to denote the updated value of a variable v. Figure 58 gives a UML state machine semantics for the PRISM code in Figure 57. Certain care must be taken in reading the state machine since in UML 2 square brackets are used for guard conditions on transitions, whereas they are used to prefix commands in PRISM.

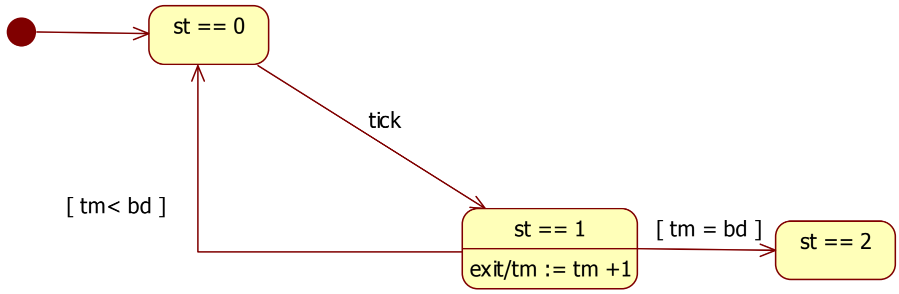


Figure 58: UML State machine of PRISM clock

The Clock module can increment a counter tm that represents the current clock tick and then returns to a state where st == 0 or to the state where st == 2 depending on whether tm has reached a bound given by a constant bd. PRISM will not allow a global variable to be modified within a command that has a non null synchronisation event label. From the perspective of any other state machine that is run concurrently with that of Figure 58 the only relevant information is whether a tick event occurs. Thus a semantically equivalent state machine within a concurrent system for the Clock module is given in Figure 59.

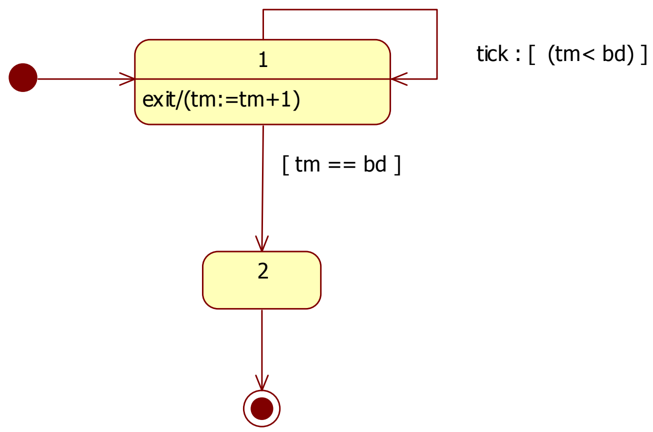


Figure 59: Optimised clock process

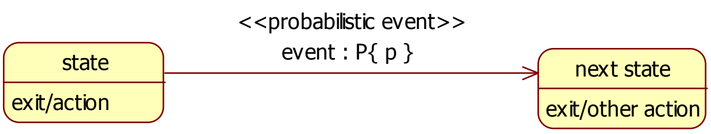
Figure 60 defines another PRISM module that describes a simple generic kind of activity. Lines starting with keyword `formula` are effectively macros that will be inlined prior to verification. An expression of the form (boolean expression) ? a : b will evaluate to a if the boolean expression is true and evaluates to b otherwise. The macro `rho_iterate0` linearly changes from value `rho0` to `rho1` as `count0` increases from 0 to `interval0`. The macro is used to define the probability distribution for a transition to occur when the Activity module is in state (x=go). Thus the distribution is dynamically modified during execution. Initially the probability of a transition from state go to terminate is almost zero. With each subsequent iteration this probability is increased until it reaches certainty. That results in guaranteed termination at that point if it has not already occurred probabilistically prior to that.


```
const int interval0 = 8;
global count0 : [0 .. interval0] init 0;

formula rho_itr0 = rho0*(1- count0/interval0) + rho1*count0/interval0;
formula rho_iterate0 =
   ((count0 >= 0)& (count0 <= interval0)) ? rho_itr0 : rho1;
formula incrmnt_count0 = (count0 < interval0) ? count0+1 : count0;

const int idle =0;
const int go =1;
const int add_one_to_count =6;
const int exec=2;
const int terminate=3;
const int return=4;
const int stop=5;
const int num_act_states = 6;

module Activity
     x : [0.. num_act_states] init idle;
     [tick] (x=idle) -> (x'= idle); // waiting for trigger
     [startA] (x=idle) -> (x' = go);
     // Probabilistic choice of run or end with incremental probability.
     [] (x=go) -> rho_iterate0:(x'=add_one_to_count)
                 + (1 - rho_iterate0):(x'=terminate);
     // Increment the iteration counter by one.
     [] (x=add_one_to_count) -> (x'=exec)&(count0'=incrmnt_count0);
     [run] (x=exec) -> (x'=return);
     [tick] (x=return) -> (x'=go);
     [end] (x=terminate) -> (x'=stop);
     [tick] (x=stop) -> (x'=stop); // terminating loop
endmodule
```

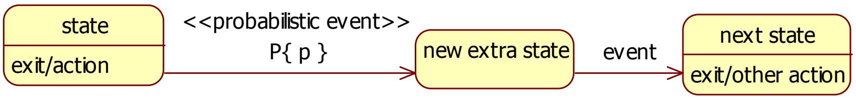
Figure 60: Elementary PRISM activity

The automaton in Figure 60 is described abstractly by the state machine in Figure 61 The state machine contains all the transitions from the automaton but does not include the code defining how the transition probability is recalculated at each iteration. UML state charts do not allow probabilistic transitions so we have encoded them as trigger events on the transition, which informally represent the probability of the transition occurring. Such annotations are labelled with the stereotype <<`probabilistic event`>>.

We define a semantics for this stereotype by defining a translation from a UML state machine into a MDP that is described in the PRISM scripting language. The translation proceeds as follows. First replace each transition of the form



by two new transitions of the form

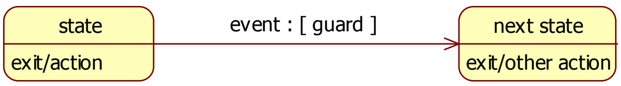


Where "new extra state" is not already present in the automaton and performs no actions. The need for this construct is because PRISM does not allow stochastic transitions to also be labelled with synchronous events. The two parts must be separated to first make a stochastic choice and then a synchronisation. The separation must be in this order to ensure that all the stochastic transitions leaving the state are still defined by a probability distribution. Once this is complete define a one to one function st from state labels to natural numbers. This will be used to define state values in PRISM, which must be natural numbers.

The resultant automata can then be translated into a single PRISM MDP automaton. We will use a new variable x ranging over the values st(state) to define the states of the PRISM module representing the automaton. For each stochastic distribution of state transitions state → p_i $state_i$, where i ranges between 1 and n, and the p_i sum to 1, define a PRISM transition as follows. We assume each $state_i$ contains at most a single exit action a_i that can be interpreted as a PRISM expression. The additional PRISM command is

[ ] (x = st(state)) → p_1 : (x' = st($state_1$)) & a_1 + ... + p_n : (x' = st($state_n$)) & a_n

For each non-stochastic transition in the automaton of the form



add a PRISM command to the module of the form

[event] (x = st(state)) & guard → (x' = st(next state)) & action

Once all transitions are processed in this way we have constructed all the PRISM commands necessary to define a module that is equivalent to the original probabilistic automaton.

Notice that because Activity and Clock contain transitions labelled with tick each one can only perform a tick transition when the other does so synchronously. This effectively forces the transitions in Activity to occur as time passes according to Clock. However, we do not need to force every transition to cause time to move forward. Only transitions that represent noticeable amounts of time need to cause a subsequent tick transition to occur.

From a cursory glance at the state chart we see that in the idle state and the stop state tick event transitions can occur any arbitrary number of times without a state change. These tick loops are essential if this automaton is run synchronously with the Clock module. There is also a tick transition from the return state to the go state. That can only occur when a tick transition synchronously fires in the Clock automata. Initially Clock and Activity are in their respective start states. At this point Clock can only progress by executing a tick transition. If there were no other tick transition in Activity apart from the one between return and go there would then be a deadlock since Activity would not be able to perform a tick transition in the idle state. Next consider if Activity has the first tick loop but not the one on the stop state. In that case once Activity reaches the stop state Clock would again become deadlocked. This illustrates a drawback with such automata. Intuitively it would be appealing to allow synchronisation only to be necessary during the execution of the automata and not for this to be required before the automaton is initialised or after it has completed.

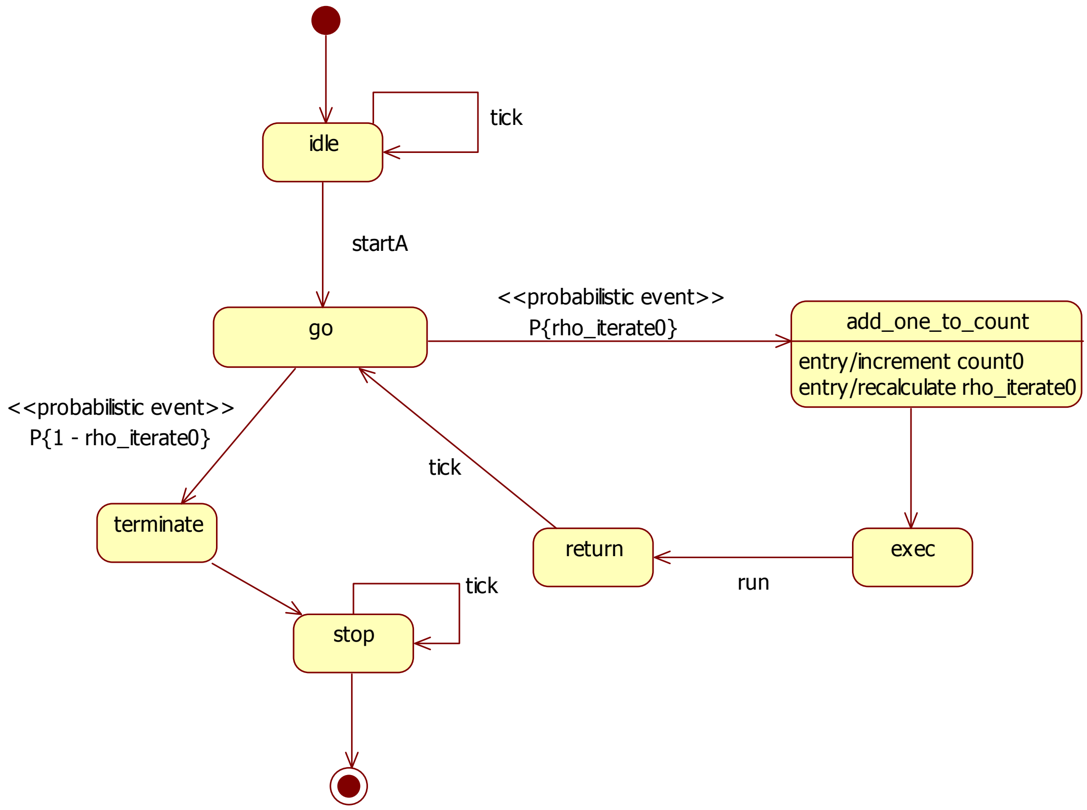


Figure 61: State machine for Activity module

Another point worth noting is when it is required for an automaton to synchronise with some other automaton, say for example via the run event label, and also to synchronise on a tick event to mark time passing. That can only be achieved with two transitions, one for each synchronisation, as is the case for Activity.

```
label "endsBeforeT" = (tm<T);
label "ActivityFinished" = (x=terminate);
Pmin=? [ F "endsBeforeT" & "ActivityFinished" ]
```

Figure 62: Probabilistic temporal logic properties

Figure 62is the content of a properties file used by PRISM to verify the automata. Abbreviations are allowed via labels. The variable T is used to represent the amount of time that has passed. The temporal formula is the minimum probability that eventually the terminate state is reached before tm becomes T. When this has probability 0 that means Activity did not terminate by time T. A subtle point here is that if we only consider Activity and Clock running synchronously then the probabilistic temporal property in Figure 62 will always have value 0. This is because in PRISM transitions do not have to be enabled fairly, which formally means if a transition is enabled infinitely often then it must eventually be executed. Therefore the startA transition does not have to occur, leading to a minimum probability of 0.

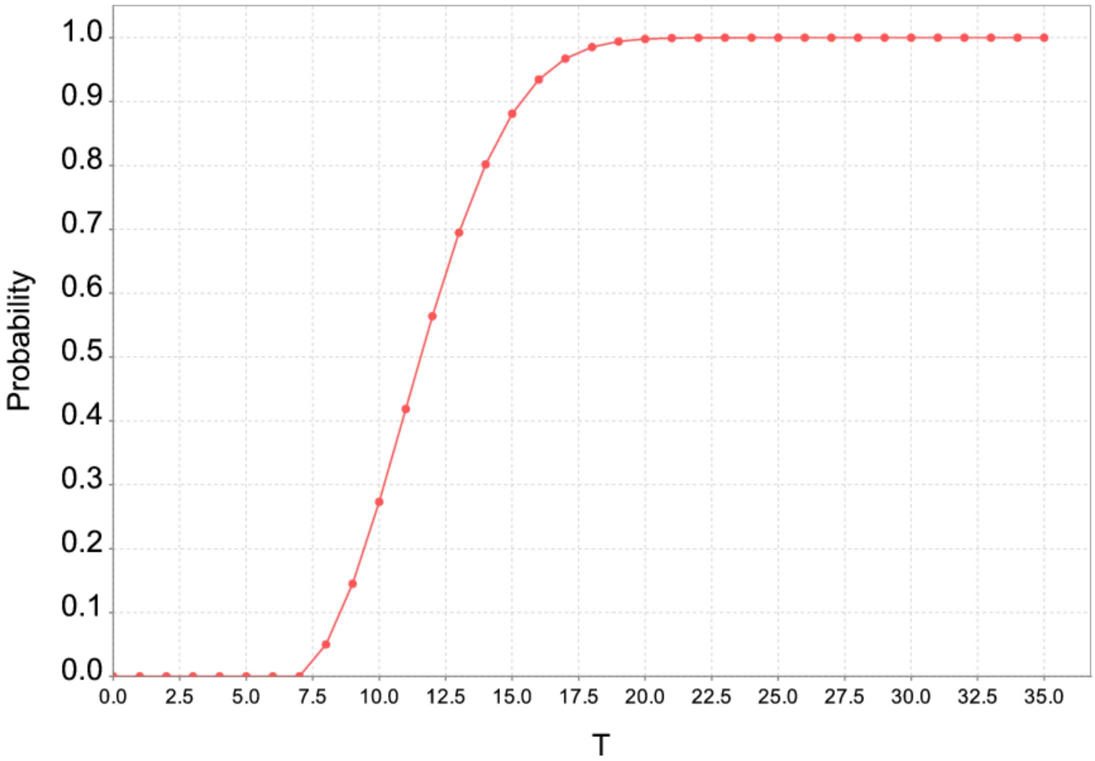


Figure 63: Estimated termination time T for Activity after reaching state go

If we introduce a third automaton that forces startA to occur at time 0 through additional synchronisation then the time to complete Activity is shown in Figure 63 The graph shown in Figure 63 is output by the PRISM tool itself.

C.2.3. Role of PRISM for real-time application reactive modelling

The pros and cons of the PRISM tool for IRMOS are as follows (where MDP is short for Markov decision process):

- Pros
  - probabilistic temporal logic properties allows arbitrary properties to be investigated in a model, which means analytical automated verification against specifications is possible
  - the PRISM language is an excellent candidate for a backend language that frontend languages can be compiled too

- MDP automata allow specification of hard time constraints and exact calculation of expected time, which can then be automatically estimated
- MDP automaton allow interactive decisions to be modelled directly
- PRISM tool can describe light weight high level conceptual models as well as detailed component descriptions
- PRISM tool is open source distributed under GPL

- Cons
  - PRISM tool is suitable for use only by experts
  - PRISM tool has subtle semantics that can easily introduce unintended deadlocks
  - PRISM does not provide a powerful IDE for authoring models

In summary at the technical level PRISM is an excellent vehicle for describing the kind of models that are relevant to IRMOS. The difficulty is ensuring that these models are correct with respect to real-time application semantics and do not contain spurious deadlocks caused by poor temporal synchronisation or states that can be bypassed due to unintentional nondeterminism. From this discussion it seems that a fruitful approach is to describe real-time application reactive models in a formal language constructed explicitly for that purpose and then define a translation into PRISM for automated analytic verification and performance estimation. The translation will guarantee that the MDP reactive model is then correct. A promising candidate for such a formal reactive modelling language is a stochastic process algebra that can be given MDP semantics. This option will be investigated during the next phase of the workpackage for planning virtual service networks.

.

C.3. Stochastic Behavioural Models

This section describes the model semantics that have been implemented so far with stochastic timed automata and discusses what support can be given for automated analysis of the models.

A fundamental characteristic of a reactive temporal model is to determine at each instant of time if an application can continue or will terminate. What happens in the next moment is dependant on interaction events that are triggered between activities and internal processing of data within each activity. When time is modelled by discrete clocks then at each clock tick the reactive model must determine whether the application will continue to execute until the next clock tick. In practice the characteristics of input data and benchmarking results will be stochastic so that the reactive model will at best be able to determine a probability of whether the activity will continue from one clock tick to the next.

We discuss how such a reactive model can be constructed from an abstract application component together with benchmarking statistics, estimates of data inputs and the interactions between the activities. The purpose of the following sections is to explain how reactive real-time application models can be constructed in the form of probabilistic timed automata. For this document it is assumed that the defining characteristics for activities within an application will be in terms of benchmarking over some data domain. As mentioned earlier benchmarking data is not enough to define application performance since benchmarks do not characterise how interactions between activates and users affects performance. Interdependencies and interactions between activities over time have to be modelled separately using the benchmark data as one of the inputs.

C.3.1.1. Reactive probabilistic automaton of an activity

A temporal benchmark consists of a series of experiments to derive performance variation over some set of critical parameters in order to discover how long some activity will take to complete. When modelling an abstract real-time application components there will be uncertainty in the precise nature of the input data. This is true since the output of one activity will be an input to some other activity. Activity outputs will most likely not have an exact known value prior to enacting the application and will only be estimated from other types of benchmarks. It may also be the case that the initial data characteristics may be uncertain. Therefore an application reactive model starts with the following inputs

- temporal benchmarking figures dependant on a set of parameter values for each activity in the application
- estimates for the sizes of input data for each activity

We assume that the estimates for the input data size are given in the form of a probability distribution. That means we assume the benchmark data parameters are defined over a Borel measurable space. If the data is a bounded subset of the real numbers this will always result in a measurable space. For example, data parameters might include the number of pixels in a frame. The estimates for the data are assumed to define a probability distribution p over BV, where BV is the vector space consisting of all possible benchmarking parameter values.

A temporal reactive model of an activity defines at each time instant the probability that the activity will terminate or continue. Since we are modelling over discrete time the reactive model must define at each clock tick the probability the activity will terminate or continue until the next clock tick.

Benchmarking characterised as a single function

In the first instance we will assume benchmarking for an activity A has resulted in the discovery of a function f, where f(v) is the time it takes for A to complete when the input parameters have values given by v, which is a vector in the measurable space BV. Define

$$T(f,t) = \{ v \in BV \mid f(v) > t \}$$

Consider what happens if we execute A. Suppose that A has been running up to time t without terminating. If the activity does not terminate before t then it follows that the size of data being processed by A must be of a size given by some v in the set T(f,t). Let rho(t) be the probability that A will continue to execute after time t until at least the next clock tick at time t+d. That is the conditional probability that if A has not terminated by time t it will continue to execute to time t+d. This is the conditional probability that given the data is in T(f,t) what is the probability the data is in T(f,t+d). We use rho for the notation since phonetically it pronounces the Greek letter ϱ, which is often used as notation for a probability distribution. We refrain from using the Greek letter for legibility and consistency with ASCII code in PRISM files.

Hence, the probability of A continuing beyond time t until at least time t+d is given by the equation

$$\text{rho}(t+d) = \frac{\int_{v \in T(f,t+d)} p(v)dv}{\int_{v \in T(f,t)} p(v)dv}$$

We write the expression in integral form since in many cases BV will be a continuous region over the real numbers where the integral will then become a surface integral. If the space is discrete then integral becomes a summation over all the points in the space. Since BV is a measurable space the probability distribution p is defined on all measurable subsets of BV, and hence p(T(f,t+d))/p(T(f,t)) will be defined and gives the same value as the integral above.

Distributions of benchmarking functions

In reality it will be rare that benchmarking results are neatly characterised by a single function. It is more likely that benchmarking data can be reasonably well fitted to a range of functions that are themselves characterised by a probability distribution. We therefore assume a benchmark defines a set of functions Fn from BV to the real numbers. In addition there is a probability distribution specifying the probability that the completion time for activity A with data values v is given by f(v) for a given f in Fn.

Let comp(v) denote the time it takes for A to terminate on input v, then the benchmark results ensure that the probability of (comp(v) = f(v)) is defined for each f in Fn. We write Pr(comp(v) = f(v)) to denote this value. Note that for any time t since the benchmark defines a probability distribution over Fn, the probability that comp(v)=f(v) given that f(v) = t is also defined and can be calculated using Bayes' theorem. Let q(f,t)(v) denote this probability, so that

q(f,t)(v) = Pr(comp(v) = f(v) | f(v) = t)

The function q(t) itself defines a probability distribution over BV. Consider again what happens if we execute activity A. Suppose as before that A is still executing up to time t, and has not terminated before that time. If we knew that the completion time of A was given by a particular f in Fv, then we could use the same integral as above to calculate the probability that A continues for another clock tick. However we do know that the probability f will be the correct function is given by q(f,t)(v). Hence, the probability rho(t) that A continues for at least one more clock tick is given by the equation

$$\mathrm{rho}(t+d) = \left(\frac{\sum_{f \in Fn} \int_{v \in T(f,t+d)} q(f,t)(v)p(v)dv}{\sum_{f \in Fn} \int_{v \in T(f,t)} q(f,t)(v)p(v)dv} \right)$$

We write the equation in integral form even though it may be that T(f,t) is discrete. It should be understood that in this case the integral becomes a summation. More properly the integral should be written as measures applied to a space in a Borel algebra. For functions g and h define $(g \otimes h)(u) = g(u)h(u)$. Since both p and q(f,t) are probability distributions the function $p \otimes q(f,t)$ is a measure (though not a probability distribution in general). The above equation can then be written in the form

$$\mathrm{rho}(t+d) = \frac{\sum_{f \in Fn} p \otimes q(f,t+d)(T(f,t+d))}{\sum_{f \in Fn} p \otimes q(f,t)(T(f,t))}$$

When we are using a discrete clock with ticks counted as natural numbers each clock tick will have a duration d. Then the probability A will continue to clock tick n+1 if it is still executing at clock tick n is rho(d.n). By convention we will assume d has value 1 in practice so that it is understood to be the unit of time in which all results are measured.

Example of benchmarking on single parameter characterised by a single function

As a simple illustration suppose that the benchmarking for an activity A is completely characterised by a continuous function f on a single non zero real value v. In that case f(v) gives the exact amount of time it takes for A to terminate on input v. In this case the above integrals collapse to

$$\frac{\int_{u \geq f^{-1}(t+1)} p(u)du}{\int_{u \geq f^{-1}(t)} p(u)du}.$$

Consider if the completion time is linear in the size of the data. Suppose the data distribution for BV is itself generated by a stochastic automaton over some interval, say [v0,v1] where the probability of termination is linearly increased. In this case rho(n) will be a linearly decreasing value, starting at 1 and decreasing to 0.

Elementary automaton based on rho

An elementary reactive probabilistic automaton for an activity characterised by data distribution and benchmarking can now be defined. The elementary form of the automata only describes whether the activity will continue to execute or not at each instance and does not permit interaction with other activities. Interaction will be incorporated once the types of interactions and consequences of such interactions are known. The automaton is shown in Figure 64 in parallel with a discrete time clock automaton. The two automata are synchronised on the clock tick event. This elementary automaton provides the basis from which the scenario models are derived. Those models are built by extending the elementary automaton to include other states that describe how an activity can be interrupted and resume execution and how the activity depends on resources that are available.

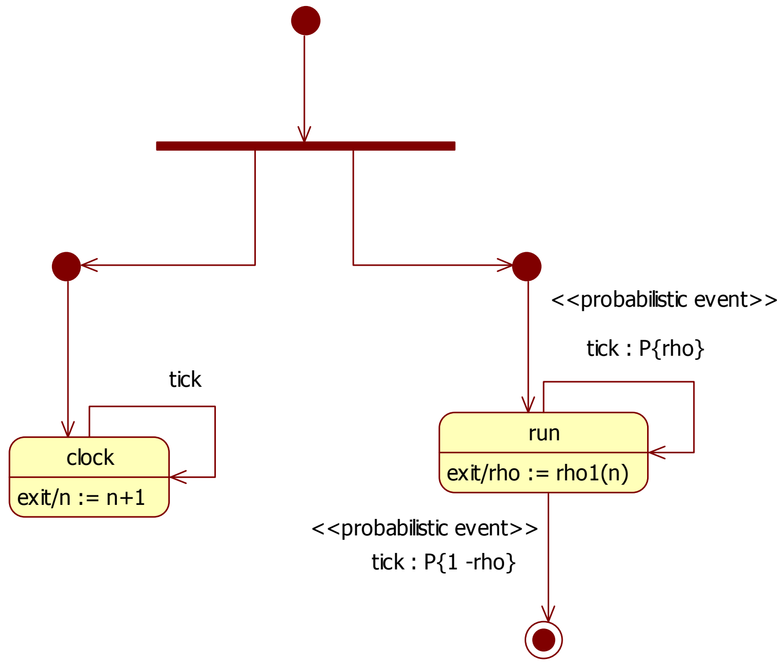


Figure 64: Reactive Probabilistic Automata for Elementary Activity

UML does not include a probabilistic event stereotype. The semantics for this stereotype are defined in Section C.2.2

C.4. Colour Correction

From a performance estimation modelling perspective this application is not substantively different from the review process. For this reason the model for this activity has been included in the Annex rather than the main body of the document.

To highlight how to model composite applications in this case we will break up the input data into different ranges. Each range will be modelled by a separate automaton, and we then compose these together to form the complete model for the application.

In this example one of the post houses will perform colour correction on material allocated at the dailies review. The amount of material to colour correct is not exactly known, but is instead defined as a probability distribution. Figure 65 shows the stochastic distribution of material for colour correction that was used in the example. In the distribution the probability for the material to colour correct being between 30 and 40 minutes is taken as close to normal with suitable adjustment to allow material outside this range to occur with 20% probability.

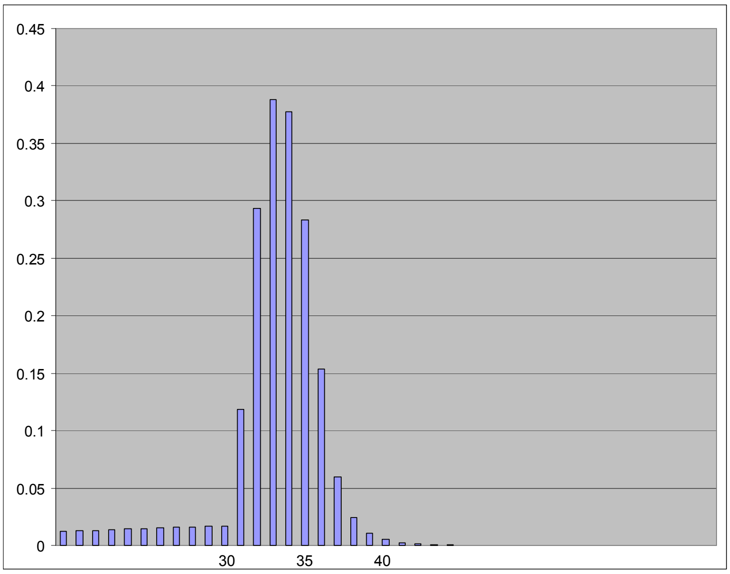


Figure 65: Probability distribution of material for colour correction

In this example the distribution on Figure 65 is crudely approximated by four scaled uniform distributions in the intervals between 0, 30, 40 and the maximum value. The scaling of the uniform distributions is according to the integrated probability mass/density of the original probability distribution in each of the intervals, and the integrated mass/density of all four scaled uniform distributions is 1. The approximation can be done at a higher accuracy by increasing the number of intervals but the model will become too large for our current aim is only to demonstrate that it is possible to use and approximate an arbitrary probability function.

In the model we allow for a more general case where there are values `cc_median_low`, `cc_median`, `cc_median_high`, which will correspond to 30, 35 and 40 respectively.

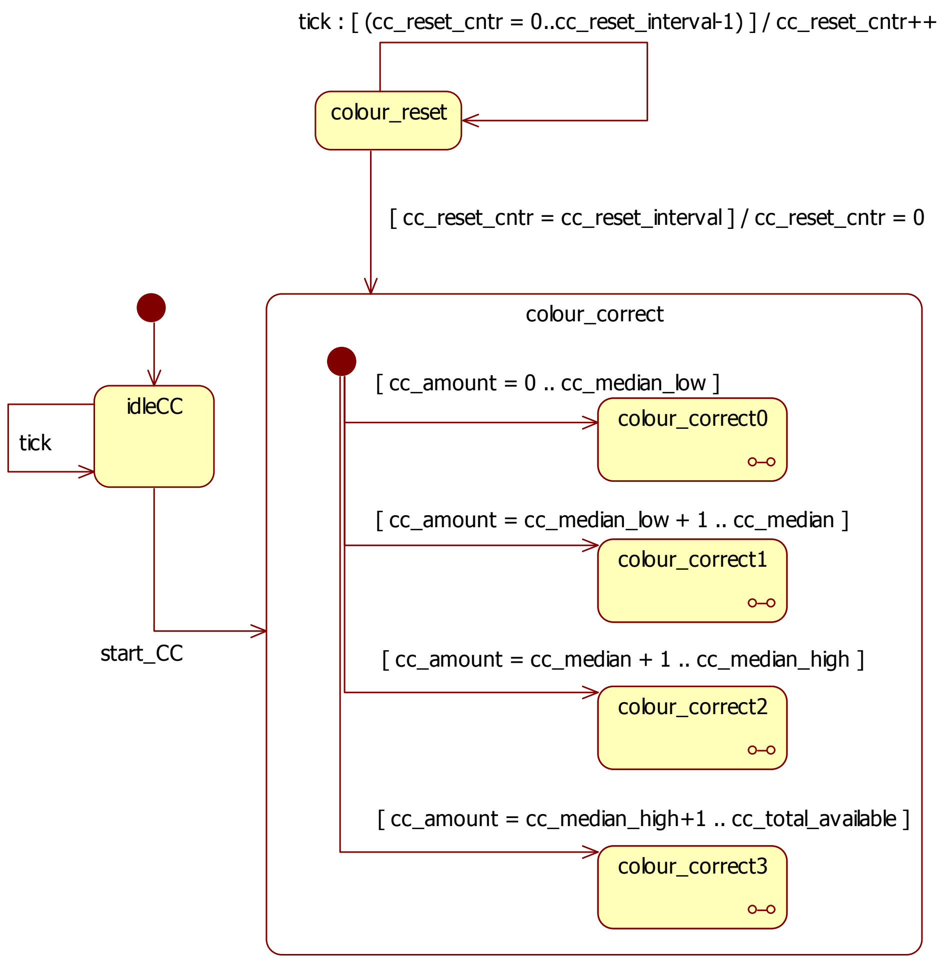


Figure 66: Top level composite automaton for colour correction

For this example we will model how bandwidth affects completion time. Other aspects such as storage could also be included. Since these are covered in great detail in the audio correction application there is no need to discuss these here. The automaton models colour correction as a process that executes at a constant rate when there are no problems. The post house accesses material via a streaming data link. If bandwidth drops too low then a reset occurs as in the other examples we have considered.

Figure 66 shows the top level composite automaton for colour correction in the form of a UML state machine. The variable `cc_amount` is used to store the amount of material that has been corrected at time T. As this amount moves from one range to the next different sub-automaton are used to model colour correction.

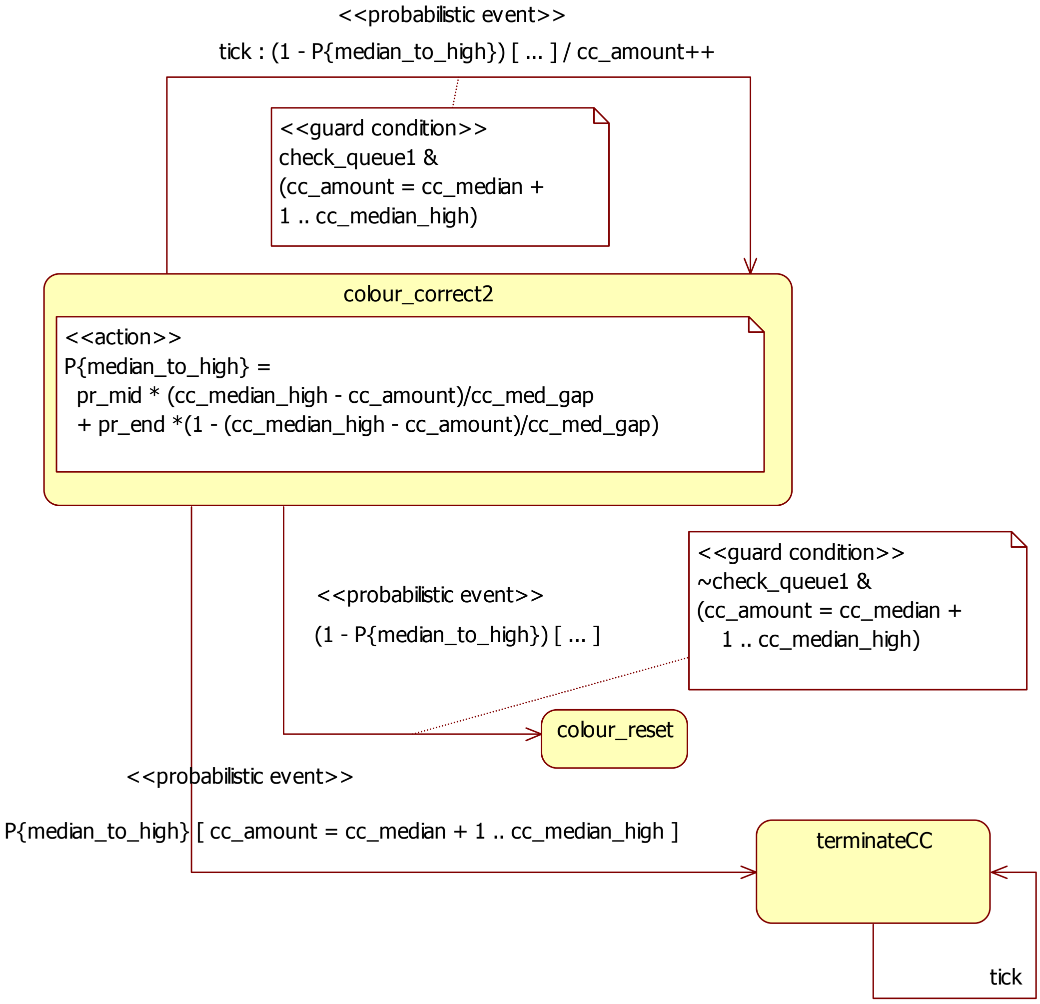


Figure 67: Sub automaton colour_correct2

Figure 65 gives the sub-automaton for the colour_correct2 sub-state of the main automaton. To clarify the various different aspects of this automaton stereotypes have been introduced to highlight when an item is an action or a guard condition. As before, we use the stereotype <<probabilistic event>> to annotate a transition that occurs stochastically. As described in Section C.3.1.1 the model varies the probability that termination will occur with each clock tick. By doing so, we effectively implement a particular probability distribution for termination occurring during a particular interval. This variation occurs in the <<action>> within the colour_correct2 state. For simplicity in this example we change the termination probability at each clock tick in a linear fashion. This is sufficient to give a good approximation to the distribution in Figure 65 and permits a compact representation for the automaton. In practice it will probably be the case that models are first approximated in this way to obtain rough and ready figures before detailed and more exact models are defined. This may well be the case, for example, when more exact figures will only be available from historical data that does not yet exist in the initial stages of service development.

The other composite automata within the colour correction model are analogous to colour_correct2 so we include them without further discussion.

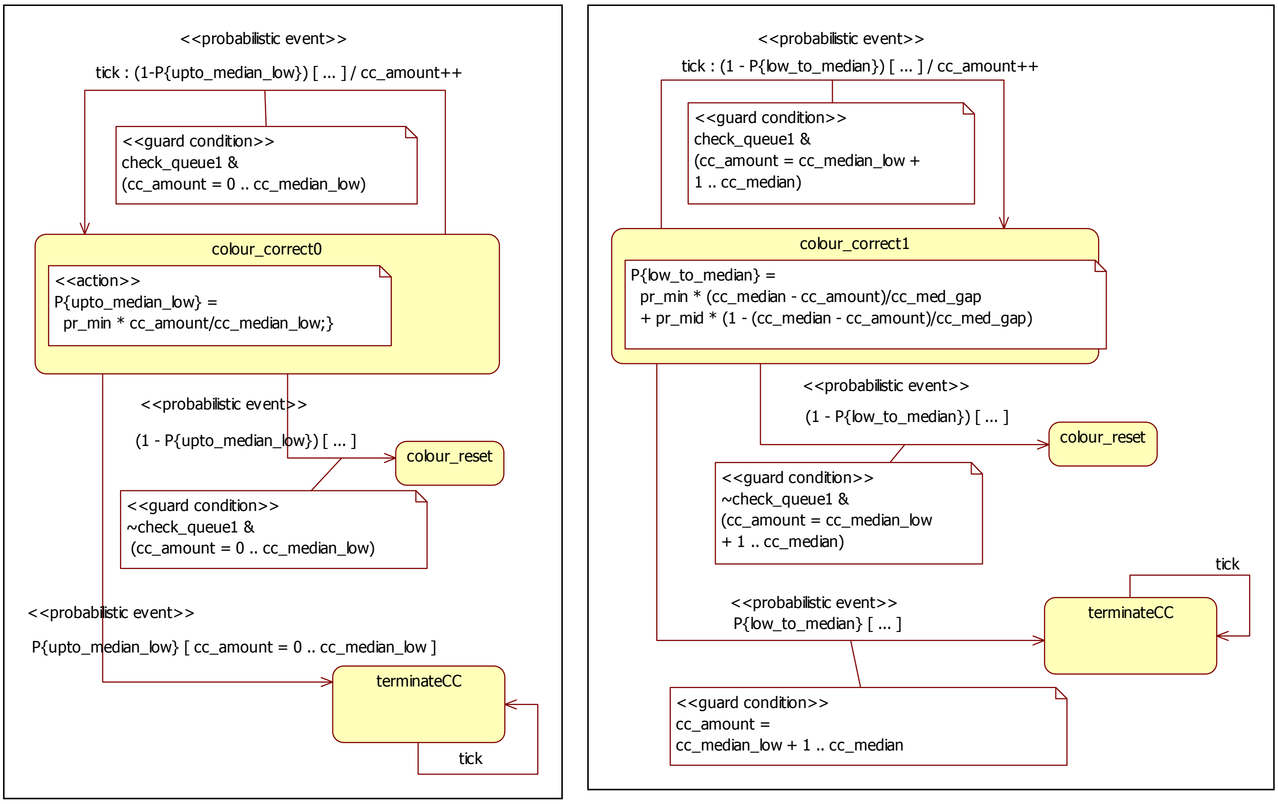


Figure 68. Sub automaton colour_correct0 and colour_correct1

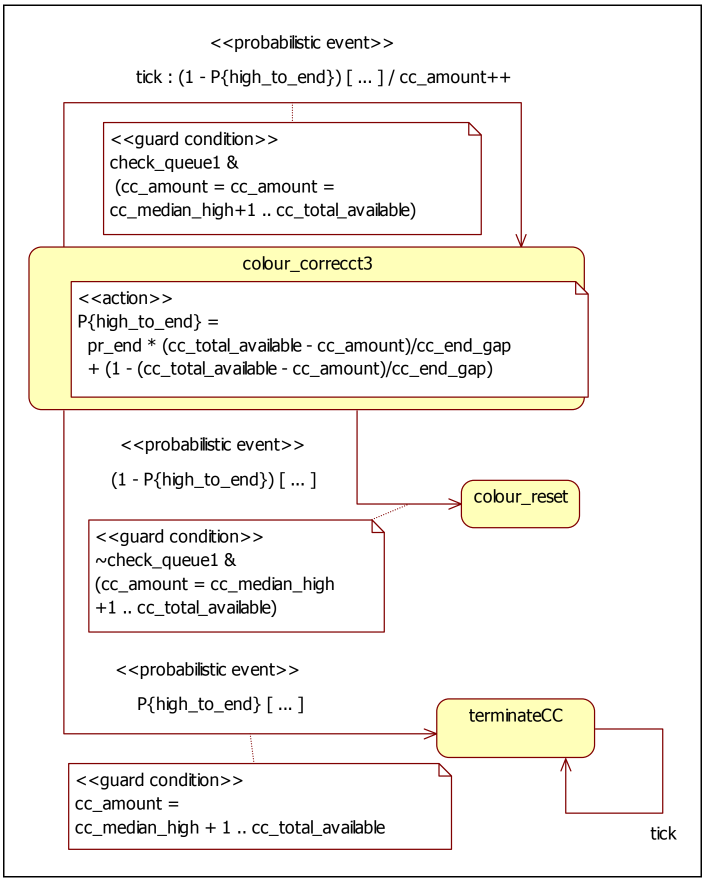


Figure 69 Sub automaton colour_correct3

The bandwidth stochastic behaviour is modelled in the same way as for the review example. The entire colour correction model consists of the four sub-automata illustrated above together with the above composite automaton, an automaton for bandwidth variation and finally a clock automaton. In the bandwidth automaton we use a probability `rho_b` to denote the probability that the level of bandwidth is sufficient for the amount of streaming video during one time unit. The model takes one time unit as a minute.

Figure 70 shows the estimated completion time for the colour correction process for just two values of `rho_b`. The blue curve shows the probability that colour correction will complete at time T when bandwidth is absolutely certain to be sufficient at all times. In this case we should see a distribution for completion time that is exactly the same as the input distribution shown Figure 65, which is indeed the case. The red curve shows the estimated time at which colour correction will complete at when rho_b is 0.98. That is, the probability the bandwidth level will be sufficient for data streaming over one time unit is 98%. This graph shows how completion time is quite sensitive to even this small amount of variation.

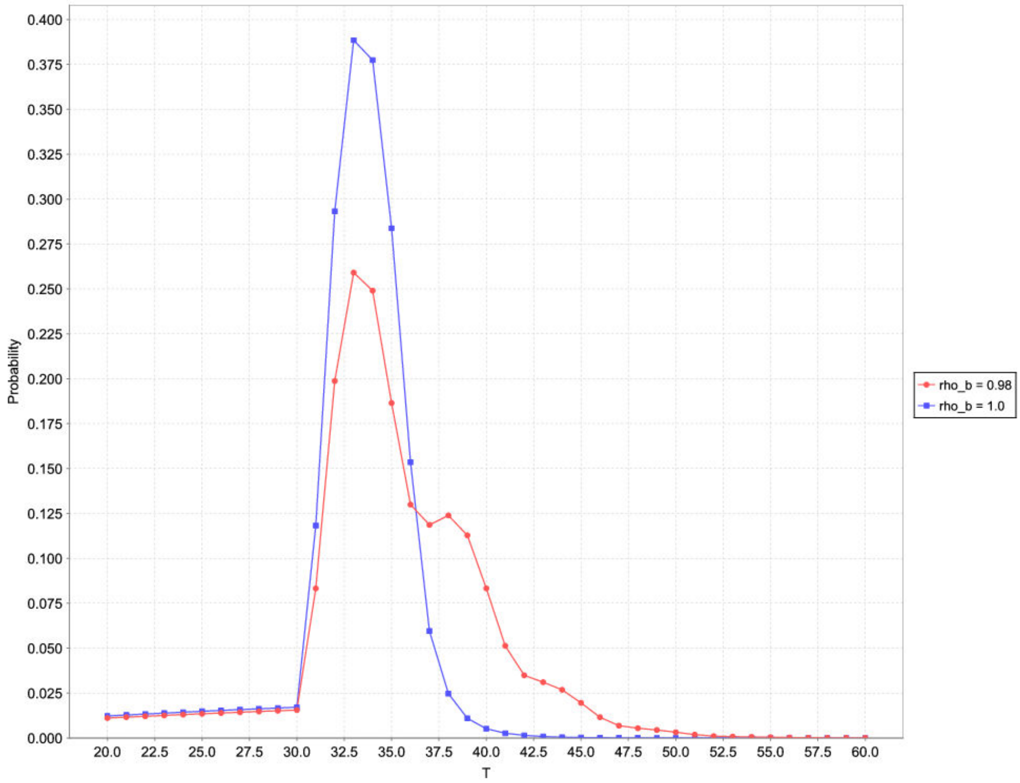


Figure 70: Estimate of completion at time T

The next graph in Figure 71 shows the probability that colour correction will complete by time T (as opposed to completion at time T). The graph gives estimates for `rho_b` varying from 80% to 98%. The interesting point to note from these figures is how the termination time becomes very much larger once `rho_b` drops below 94%. Once it drops to 90% then completion time will be significantly beyond one hour.

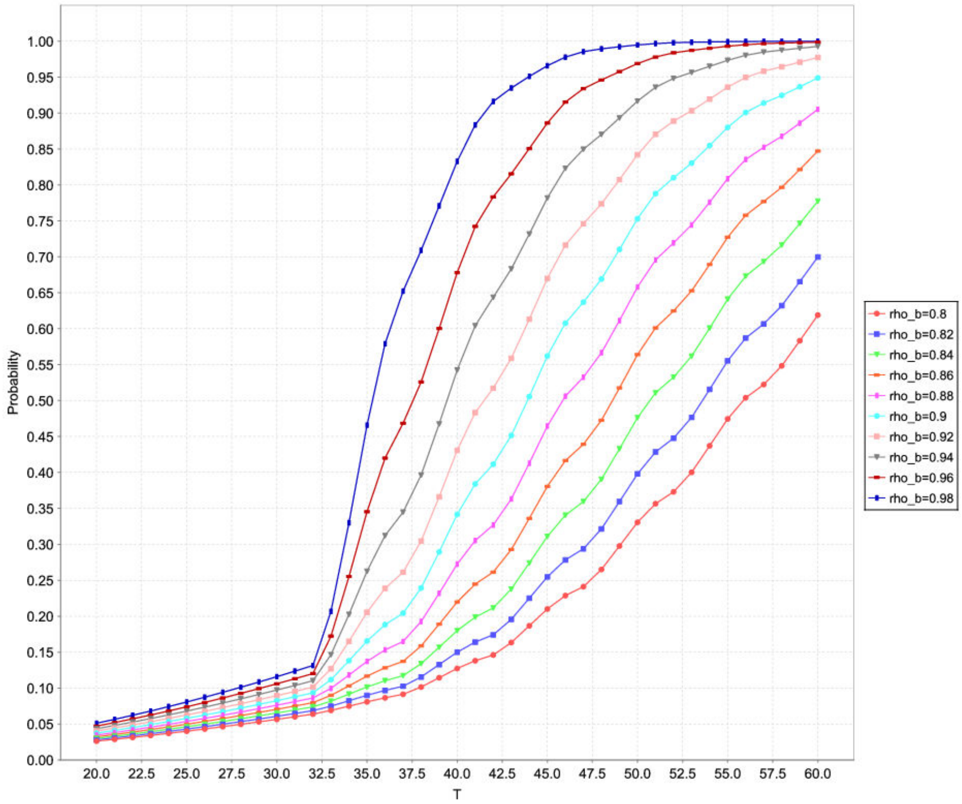


Figure 71: Estimate for colour correction to complete by time T

C.5. Original Footage Transcoding

In this section we describe the model for transcoding original footage, which is then transmitted to the director for comparison with the results from colour correction and audio correction. This is the final example we consider from the scenario. The interest for this example is in considering different levels of CPU and bandwidth levels that achieve similar results. This is of significance when trying to optimise costs whilst preserving performance. The model itself does not provide any significant advancement on the earlier stochastic timed automaton describing the audio correction activity, which is why the model is part of this Annex.

Working Scenario

As detailed in the case study description in Section 4.1 the original footage is transcoded into a compressed form to be sent to the film director, along with the version with corrections and effects applied, for before/after comparison. Compression can be done at different ratios but higher compression ratios require more processing work. Compression can be done at the ratios of 2, 4, and 8.

Scenario analysis and further assumptions

The working scenario describes only a footage transcoding step but it is implicitly preceded by a step for obtaining the footage, i.e. transferring the digitised footage over a network. Therefore, the original footage transcoding activity consists of two steps:

footage transfer over a network and footage transcoding on a processing resource. However, these two steps in essence are the same as the audio downloading step and the audio processing step respectively of the audio correction activity (the first and the forth step). We will not go into these steps in any great detail as they do not present any new challenges. We will only state the basic formulas and the assumed parameter values that are necessary.

The film footage volume ($V_{Footage}$) is the sum of the video volume (V_{Video}) and the audio volume (V_{Audio}), but by the audio correction scenario the audio is 10% of the video volume, so we calculate the footage volume as: $V_{Footage} = V_{Video}*(1.0/0.9)$.

The video volume, V_{Video}, is calculated as per Expression **(5.3.1-2)** for the audio correction model and the transfer time (T_d) is calculated as: $T_d = V_{Footage}/S_{link}$.

For the footage transfer activity we assume the same parameter values as per the audio correction activity: link bandwidth 50MB/s, video frame rate of 24frames/sec, colour depth of 16 bit, frame resolution of 2 MPixels and video time duration the same as the duration of the audio and is uniformly distributed between 20 minutes and 40 minutes.

To model the transcoding processing time (T_p) we need to know the amount of processing work (W_p) that needs to be performed and the speed of the processing resource (S_r). To calculate these quantities we use the same expressions, (2.4) and (2.5), as for the audio correction model. The speed of the processing resource is assumed to be 100MFLOPS/sec as per the audio correction activity, but here we have different possibilities for the algorithm factor: for transcoding with compression rates of 2, 4, and 8 the respective algorithm factor is 0.5, 1.0 and 3.

Activity model

The original footage transcoding activity model consists of two state machines, footage transfer and footage processing for transcoding. These state machines are identical to the first and the forth state machines of the audio correction model but use different transition probabilities because of the differences in some of the activity parameters values. Consequently, the UML model is similar to the one for the audio correction activity in Figure 40, but composed only of the top and the bottom state machine diagrams, respectively the transferring and processing state machines.

Numerical experiments

The first experiment we have performed was to obtain the completion time PDF of the two individual steps, footage transfer and footage processing for transcoding. The numerical values used for the first experiment are as stated in Section … (Scenario analysis and further assumptions), where for the compression ratio we choose to use the maximum of 8:

- footage time length: uniformly distributed between 20 and 40 minutes.
- video frame rate: 24 frames/sec, colour depth: 16 bit, frame resolution: 2 MPixels.
- average link bandwidth: 50 Mb/sec.

- transcoding processing algorithm performance factor: 3.0 (for compression factor 8).
- speed of the processing resource: 100 MFLOPS/sec.

Figure 72 shows the obtained completion time PDFs. We can perform experiments to trade off compression rate for processing power but it is obvious that with the current transfer times much larger than the processing times and without any cost model these kind of experiments will have no much value.

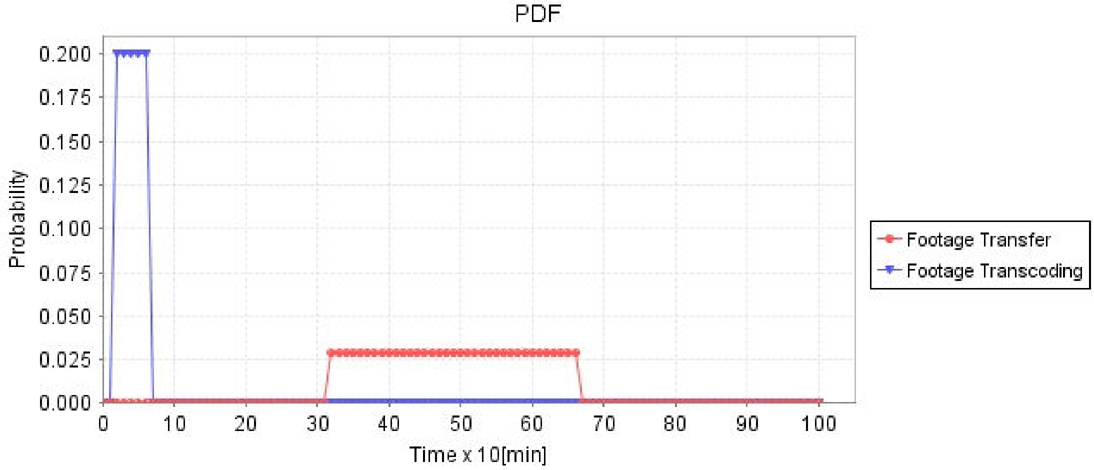


Figure 72: Footage transfer step and trascoding step completion time PDFs.

With the transcoding completion time much lower than the footage download time we might be tempted to trade some processing power for link bandwidth, though. After performing a number of experiments for different processing resource speed values and link bandwidth values and all other parameters unchanged from the first experiment we found that, trading 10 times less processing speed for 5 times more link bandwidth will give us approximately the same overall activity completion times. Figure 73 shows the activity completion time PDFs for these two cases. So, if a potential cost model indicates that it is cheaper to use 10 times less computational power than to use 5 times more bandwidth we can potentially save money.

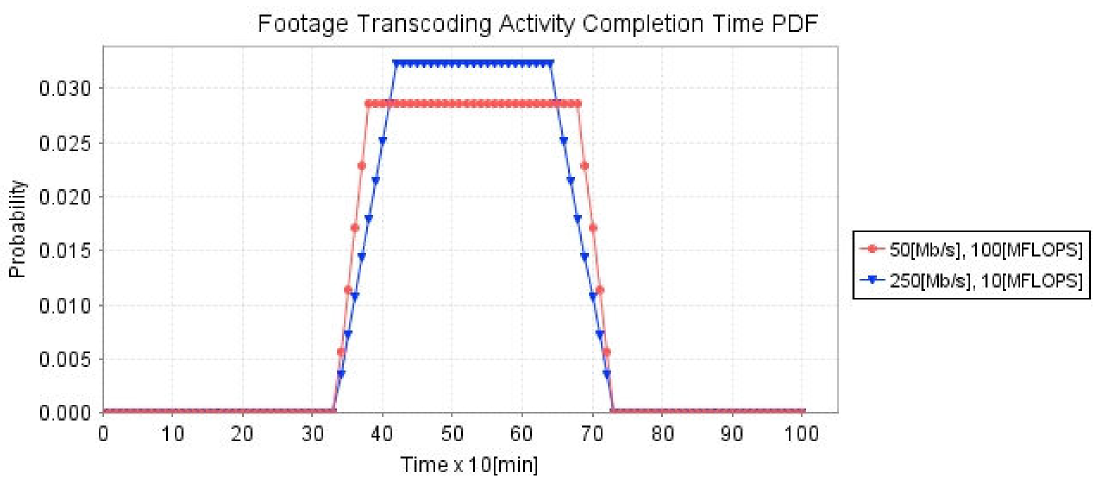


Figure 73: Overall footage transcoding activity completion time.

Annex D.

D.1. Quality Assurance and Metrics

An internal Quality Procedure for the quality assurance of IRMOS deliverables has been implemented in the project and is documented in the IRMOS Quality Plan. One of the actions included in the Quality Plan is the definition and of metrics for each deliverable that help internal reviewers of the deliverable to assess suitability for release to the Commission. The metrics are organised per work package objective as written in the project Description of Work document.

This document has been produced as part of IRMOS WP5. The WP5 objectives are:

(O1) Produce models and descriptions of the requirements of real-time applications when deployed and executed on SOIs.

(O2) Design and develop a tool and a supporting framework service for value chain participants to use when planning the layout of application specific networks of services.

(O3) Develop a formal programming language for use by application developers and service providers for the technical integration of real time applications into the IRMOS SOI.

(O4) Develop framework services (e.g. discovery, orchestration, messaging, monitoring) that enable the adaptation of applications to work on the IRMOS SOI.

This document addresses WP5 objective O1. The metrics that apply to this objective are:

Metric	Target Value	How metric is addressed in D5.1.1
Models describe all three application scenarios	YES: VERY GOOD NO: UNACCEPTABLE	The three scenarios are characterised and the modelling approaches described are shown to cover these characteristics. The post-production test case used to demonstrate and evaluate the modelling techniques is carefully designed to cover a wide range of real time characteristics so is representative of the modelling needs of the other application scenarios as well.

Models can describe all the actors in the envisaged IRMOS value-chains/networks	YES: VERY GOOD NO: UNACCEPTABLE	This metric would be better expressed as "Models are useful to the actors in the envisaged IRMOS value-chains/networks". ID2.1.2 already contains a model *of* the value-chain. The purpose of this document is to provide models that can be used *by* the actors in the value-chain. The document pays particular attention to use of modelling at different stages of the application lifecycle (design, development, deployment, execution etc.) and by different parties in the value chain (e.g. IRMOS providers, application service providers, customers).
Models describe the requirements/constraints in sufficient detail for use in SLAs and QoS specifications	YES: VERY GOOD NO: UNACCEPTABLE	As demonstrated by the case study, the techniques used in this report are capable of describing both the functional and non-functional requirements, including the QoS needed and temporal constraints on when resources need to be available.
Models capture the mappings in both directions between application-level and resource-level parameters	YES: VERY GOOD NO: UNACCEPTABLE	D5.1.1 concentrates mainly on the forward mapping problem, i.e. from application-level to resource-level where the question is 'what resources are needed to run this application'. This aligns with the resource reservation and SLA negotiation process supported by IRMOS. Reverse mapping is possible to some extent, e.g. by analysing how application performance is dependent on variations in resource performance. However, the models do not directly answer questions such as 'what range of application parameters are possible given this amount of resource?'. Further work on mapping from resource level to application level will be done in T5.4 in particular monitoring and metering to support SLA management where resource level usage information from Infrastructure Provider needs to be aggregated and mapped to higher level application oriented parameters.

[1] http://aws.amazon.com/
[2] http://aws.amazon.com/solutions/case-studies/