Experiences using the UML profile for MARTE to stochastically model post-production interactive applications

Stuart E. Middleton[1], Arturo Servin[1], Zlatko Zlatev[1], Bassem Nasser[1], Juri Papay[1], Michael Boniface[1]
[1]*IT Innovation Centre, University of Southampton, 2 Venture Road Southampton, SO16 7NP, United Kingdom*
Tel: +44 23 8076 0834, Email: {sem,als,zdz,bmn,jp,mjb}@it-innovation.soton.ac.uk

Abstract: We describe a practical approach applying the UML 2.0 standard MARTE profile to model stochastic interactive application workflows, using the PapyrusUML editor. We use the PaStep, PaCommStep, PaLogicalResource and GaCommHost MARTE stereotypes and find them sufficient for stochastic modelling with the exception of being unable to define non-standard probability distributions. We have investigated both Markovian stochastic models and discrete event simulation models, serializing UML deployment and state machine diagrams to automate model creation. The choice between using a stochastic model (e.g. PRISM Markov models) or discrete event simulation model (e.g. Monte Carlo simulations) depends on the complexity of the model, accuracy required and compute time needed. We find that PRISM models are fast to execute if the complexity is small and produce numerically accurate results. Discrete event simulation models are slower to execute but scale much better and are probably the default solution to a model of unknown complexity.

1. Introduction

The IRMOS project [1] is developing tools and techniques for modelling, simulating, analysing, and planning interactive real-time applications on cloud infrastructures. These tools and techniques support the processes involved in designing, developing, deploying and executing applications where guaranteed quality of service is needed. Our use-case applications for this paper are a post-production application based on the Digital Film Technologies Bones Digital Dailies production system [2] and an e-Learning mobile device content notification and download server [8].

The Unified Modeling Language [3] is a standard promoted by the Object Management Group (OMG). The latest release, UML 2.0, with the new UML profile for the Modeling and Analysis of Real-Time and Embedded systems (MARTE) [7] is emerging as a possible solution for providing benefits to real-time and embedded systems developers. The MARTE specification consists of three main packages. The first package defines the foundational concepts used in the real-time and embedded domain, including basic model constructs for non-functional properties, time and time-related concepts, allocation mechanisms and generic resources including concurrent resources. The second package addresses model-based design, including the description of detailed software and hardware execution platforms. The third package addresses model-based analysis, including schedulability and performance analysis, and is the focus of this paper.

In IRMOS we want to stochastically model interactive application quality of service performance metrics (e.g. completion time, response time, dropped frames) for different resource configurations so we can provision the optimal cloud resources (CPU, RAM,

storage) needed to achieve a desired quality of service. The goal is to incorporate such models into a service engineering environment that can translate high-level application descriptions into resource provisioning policies with minimal manual intervention. Our models are defined using stochastic process algebras, with finite state automata, and discrete event simulations.

There are many open source UML editors available today, but many do not support UML 2.0 and very few have a MARTE profile. To date MARTE profiles exist for Rational Software Architect 7.0, MagicDraw 15.5, PapyrusUML and the Cheddar analysis tool.

We investigate an open source UML editor, PapyrusUML [4], to create our UML state machine models and deployment diagrams which we serialize to XML. For the post-production use case we look at using a XML transformation technique, MOFScript [5], to transform serialized UML into a PRISM [6] process algebra which we execute using the PRISM toolkit. For the e-Learning use case we have developed a Matlab discrete event simulation that we hope to configure in the future using a serialized UML model. Our models generate estimated performance metric probability distributions for different resourcing solutions, allowing us to optimize the resource provisioning for our cloud deployment environment.

2. Objectives

Our objective is to evaluate, in the context of our use cases, how well the MARTE profile for UML 2.0 standard can be used to model interactive application behaviour for the purpose of discrete event simulation and Markovian stochastic performance modelling and subsequent cloud resource provisioning optimizations. We want to identify the key MARTE stereotypes we can use, assess their strengths and weaknesses for representing stochastic models and report our practical experiences doing this. We also want to give concrete examples of how we have used MARTE to help other practitioners in the field.

3. Use Cases

Our first use-case is a post-production application based on the Digital Film Technologies Bones Digital Dailies production system [2]. In this scenario, collaborative and distributed colour correction is performed as part of film post-production. A post-production house is contracted to perform colour correction to some film shots that will be selected by a film director during a review of the digital dailies of a film currently under production. The number of shots needing colour correction cannot be determined in advance as this depends on decisions made by the director. The director estimates that colour correction will be applied to approximately 30 +/- 10 minutes of footage. The colour correction and review activities occur concurrently and involve: (1) the colourist effects specialist streams the digitised video from the Bones Service provider, (2) the colourist applies initial correction to the video and (3) the colourist and director interactively review the corrections that are applied through real-time stream processing of the video using applications installed at the service provider.

The second use case is an e-Learning application based on GIUNTI's mobile content notification and download system [8]. In this use case users representing a variety of different groups of people (tourists, students and business travellers) subscribe via mobile phone for content notification. As users from different groups physically move location they are notified of media content relevant to their location and can request downloads of this content from a web server. Each user will have their own quality of service requirements, and these will be translated into provisioning resource expectations, based on their hardware, current mode of transport (walking, car, train), number of concurrent users

and subscription fee paid (gold or standard service). Interactions are asynchronous and cloud resources need to be provisioned to support different user group profiles.

4. Methodology - UML 2.0 and MARTE profiles to model applications

Modelling with UML 2.0 state machines and StarUML

We chose to experiment with both UML state machine and deployment diagrams to assess suitability of MARTE profile stereotypes at different levels of granularity. We used the UML 2.0 editor PapyrusUML [4] to create state machine diagrams for the post-production colour correction workflow, and deployment diagrams for the e-Learning use case. Deployment diagrams offer high level coarse grained modelling, whilst state machine diagrams offer a finer level of granularity. Each use case requires a different level of analysis.

For our post-production use case we created two UML 2.0 models. Our first model was a set of three state machines using three custom stereotypes (no MARTE profile), created using the StarUML [9] editor. We serialized our UML diagrams to XML and used an open source tool, MOFScript [5], to transform the saved UML to a PRISM continuous-time Markov-chain. Once loaded into PRISM we ran our stochastic models and generated probability distributions for application performance metrics that our cloud provisioning system could use to allocate 'best fit' resources (CPU, RAM, disk storage). Our second model used the PapyrusUML editor, with MARTE profile stereotypes, to replace these custom stereotypes and validate the suitability of the MARTE profiles for our stochastic modelling approach.

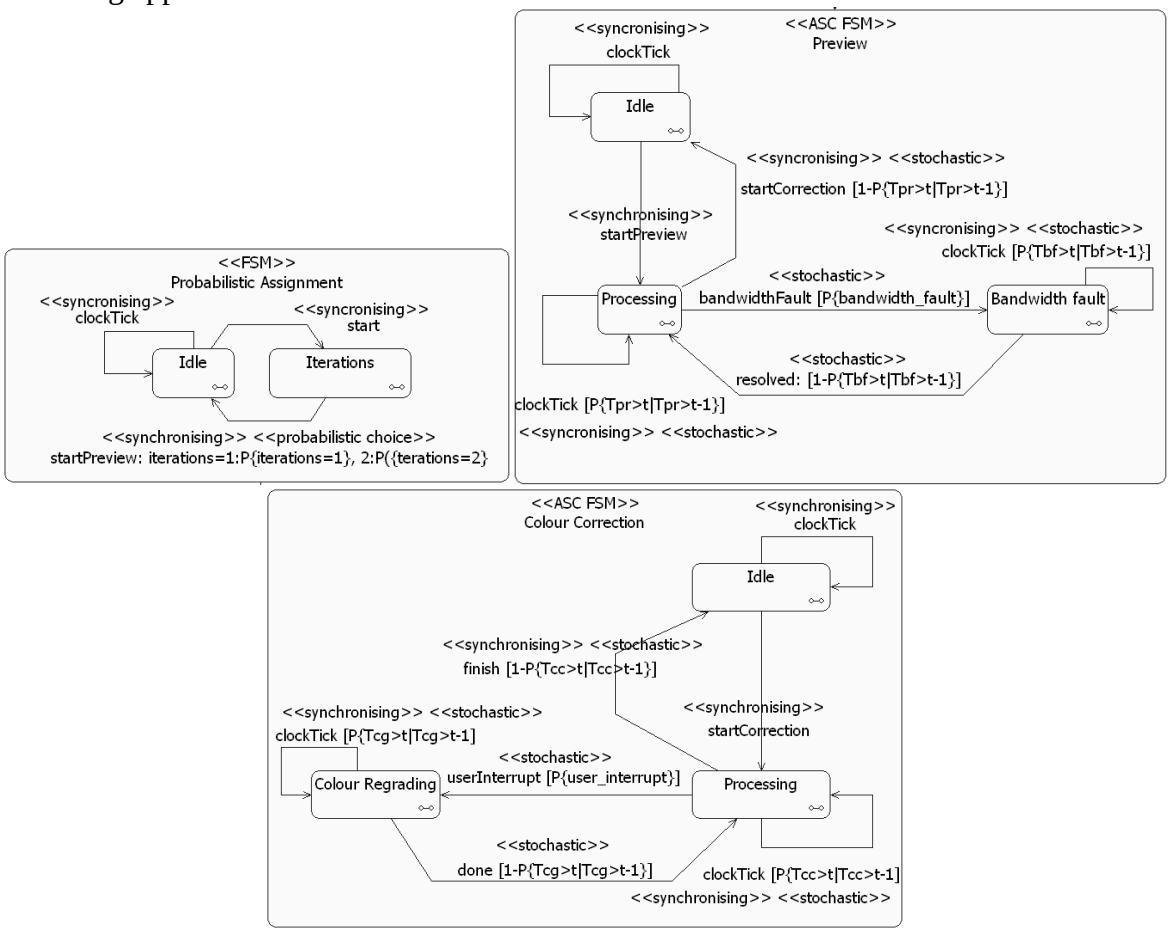
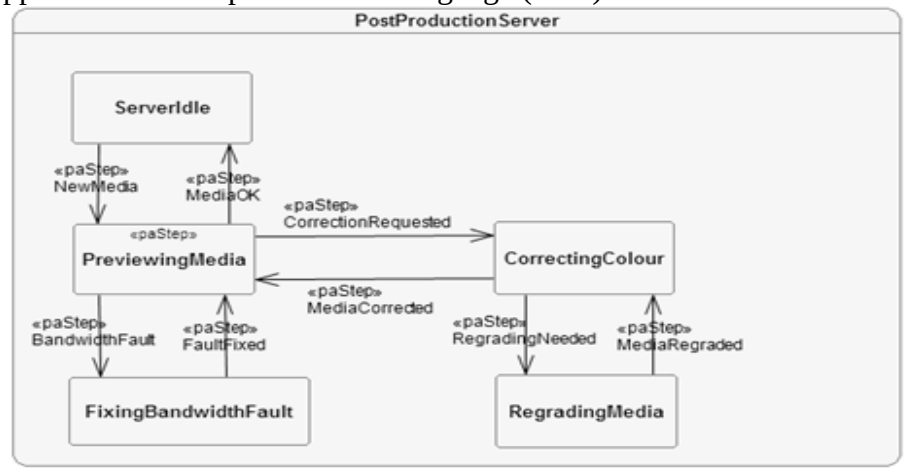


Figure 1: UML 2.0 model with custom stereotypes for post-production use case

The state machines shown in figure 1 show the UML model used for our simplified post-production model, focussing on the colour correction process. We have used three custom stereotypes, "synchronizing", "stochastic" and "probabilistic choice". The "synchronizing" stereotype allows us to synchronize the firing of events in different state machines. The "stochastic" stereotype indicates a state transition that is triggered according to a probabilistic guard function. The "probabilistic choice" stereotype indicates a stochastic user choice that will change a global variable defined in the guard function. These custom stereotypes are directly mapped to PRISM syntax by the MOFScript translation step and are a requirement of our stochastic modelling algorithm on the UML model.

Modelling with UML 2.0 state machines, MARTE profile and PapyrusUML

Our second UML model makes use of the MARTE Performance Analysis Modelling (PAM) package via the PapyrusUML editor. Each state uses the 'PaStep' stereotype, derived from 'GaStep', and defines a number of non-functional properties (NFP's). We use properties 'blockT', 'rep' and 'prob', all of which are of type NFP_Duration or NFP_Real, and thus support the Value Specification Language (VSL) for the attribute 'value'.




```
Some MATRE property values (state & state transition elements)

PreviewingMedia::paStep
+ blockT = (value=3.0, unit=min, precision=1.0)
+ rep = (value=2.0, statQ=mean, expr=[1.0..5.0])

BandwidthFault::paStep
+ prob = (value=normal(0.05,0.02), statQ=distrib)

MediaCorrected::paStep
+ blockT = (value=10.0, unit=min, precision=3.0)

MediaRegraded::paStep
+ blockT = (value=normal(4.0,5.0), unit=min, expr=[3.0..20.0])
```

Figure 2: UML 2.0 model with MARTE profile for post-production use case

The colour correction state machine in figure 2 represents a number of stochastic state transitions which have a probability distribution associated with them. Figure 2 shows some of the property values of these 'PaStep' UML state and state transition elements. To represent a probabilistic distribution of processing time we use the probability distribution operations defined by the NFP_CommonType data type (e.g. 'normal()' or 'uniform()'). The value of the 'blockT' property is the duration of a transition and is defined as either a literal value or a call to a probability function, which returns a probable distribution of values according to the parameters provided (guassian mean/stdev for 'normal()'). The modeller needs to decide on a representative value distribution for each state transition. The same technique is used for 'prob', the probability of a state transition branching, and 'rep', the

number of repetitions each branch should invoke. Where needed the 'expr' attribute can be used to specify a more complex auxillary expression to go alongside the value, such as an interval (e.g. '(value=2.0, statQ=mean, expr=[1.0..5.0]'). We do not specify the clock, assuming the default PAM ideal clock and letting the subsequent MOFScript processing add this as required by PRISM.

Modelling with UML 2.0 deployment diagrams, MARTE profile and PapyrusUML

Some MATRE property values (nodes & communication links)

```
MobilePhone::paLogicalResource
+ utilization = (value=50.0, statQ=mean)
+ throughput = (value=5.0, unit=Hz)
+ poolSize = 1000.0

MobileServerLinkIN::paCommStep
+ blockT = (value=100.0, unit=ms, precision=10.0)
+ msgSize = (value=1024.0, statQ=mean, expr=[128.0..8192.0])

WebServer::gaCommHost
+ throughput = (value=1000.0, unit=Hz)
+ blockT = (value=5.0, unit=s, precision=2.0)
+ packetT = (value=10.0, unit=ms, precision=1.0)
+ capacity = (value=1000.0, unit=Mb/s)
```

Figure 3: UML 2.0 model with MARTE profile for e-Learning use case

For our e-Learning application use case we define a UML 2.0 deployment diagram again using the MARTE stereotypes, shown in figure 3. This model is at a coarser level of granularity since the underlying application use case is more complex to model. We use this model to define the discrete event model used by the discrete event simulator reported in section 6. We use the 'PaCommStep', 'GaCommHost' and 'PaLogicalResource' profiles to represent our stochastic model parameters. The 'PaLogicalResource' specifies 'poolSize' for max number of users, 'utilization' for mean number of concurrent users and 'throughput' for requests per second. The 'GaCommHost' specifies server 'throughout' for requests per second, 'blockT' for request processing time, 'packetT' for LAN delay (mean) and jitter (standard deviation) and 'capacity' for LAN bandwidth. The 'PaCommStep' allows input and output network links to be characterised, with 'blockT' representing the network delay and 'msgSize' the average size of messages sent.

5. Methodology - stochastic algebra for performance estimation

For the post production use case we use PRISM [1], a probabilistic model checker that supports three types of probabilistic models: discrete-time Markov chains (DTMCs), continuous-time Markov chains (CTMCs) and Markov decision processes (MDPs). We use a pre-processing step to create PRISM models from serialized UML (XML format) based on the MOFScript [5] open source tool. Once generated our PRISM models provide estimated performance metrics and can be run iteratively to optimize hardware resource

specifications based on cost and quality of service returned. We outline our approach and its implications on the UML models in this paper; algorithm details and results from modelling and optimization experiments can be found in [10].

MOFScript is a powerful tool to process serialized UML files to generate source code (PRISM in our case). Using an automatic UML to PRISM translation step allows users to work at the human understandable UML level, and leaves the modelling system to create an executable model.

For simple state machine diagrams (up to 5 state machines, each with up to 5 states) this approach works very well and model solutions are fast to compute. However PRISM models do not scale well, both for model complexity and execution time. The complexity problem with models results partly from the lack of object oriented capabilities in MOFScript. In order to process recursive state machines, transition, guards and other elements commonly found it is necessary to make use of complex data structures that are missing in MOFScript. The PRISM syntax itself has a few restrictions when processing complex states machines also.

The first problem is that it is not possible to trigger a state transition and update a global variable in a single step; the solution is to add extra states but this adds to the complexity of the model (which increases model complexity, compute time and memory requirements). The second problem is the lack of a global/local variable scope definition (not currently supported by UML and the Value Specification Language) which forces us to use complex data structures in the PRISM code. The third and last issue is related to stochastic transitions for arbitrary probability distributions. There is no MARTE mechanism to define a custom distribution that does not follow one of the standard profiles in MARTE (e.g. normal, uniform etc). In reality modelling stochastic user choice / behaviour will almost always require a non-standard distribution and resource behaviour often follows hyper-exponential, coxian and/or mixed distributions.

These problems mean that this UML to PRISM approach appears practical only for small-scale models (the colour correction model itself is a simplified version of the full post-production workflow). These small-scale models are however fast to execute and could be applied to scenarios where near real-time model output is required.

6. Methodology - discrete event simulations for performance estimation

For our e-Learning use case we use a Monte-Carlo based discrete event simulation approach to model performance metrics. We model each deployed component (web server, mobile and laptop devices and network connections) and stochastically characterize these components to provide us with our model parameters. The event simulation can then be run and probabilistic quality of service computed for a specific hardware profile. This simulation can be run many times to optimize a hardware provisioning profile based on cost and quality of service delivered.

The model itself has been created in Matlab from the UML deployment diagram shown in section 4. At the moment model creation is manual, but we intend to create a pre-processing step to transform the UML deployment diagrams (with MARTE stereotypes) into a set of event triggers suitable for input into a generalized event simulator (e.g. using the Matlab Simulink toolkit). We will use a combination of a XSL transform and Matlab pre-processing code. Our model is focussed on tasks performed by the application user (e.g. we model the processing of a sequence of user requests for mobile content sent to a web server) as well as specifying the deployed compute and network resource characteristics.

Our initial experiments have shown that the discrete event simulation approach allows us to model much more complex scenarios, and avoid the limitations we see in PRISM. We are currently working at the UML deployment level but will soon look at specifying for each UML deployment node a state machine for finer grained modelling. Our models are

slower to run than PRISM models, especially if the number of iterations are high to provide the best accuracy, but scale well and are still useful for offline performance estimates suitable for optimization of cloud resource provision options.

7. Results - strengths, weaknesses and practical experience

We have found that UML 2.0, with the MARTE profile, is very expressive and allows us to represent well in UML deployment diagrams and state machines models stochastic annotations for user behaviour patterns, network load profiles and processing time profiles. The MARTE stereotypes we consider most useful for our interactive application modelling are the PaStep, PaCommStep, PaLogicalResource and GaCommHost.

The only thing we need that is missing from MARTE is a way to represent custom (i.e realistic) value distributions to represent profiles of expected user behaviour (e.g. download behaviour as a factor of time of day) and resource behaviour.

We found the PapyrusUML editor (version 1.12.3) easy to use but unstable, requiring frequent backups of the UML diagrams to prevent loss of data; the editor is currently being supported and we expect future versions will resolve this problem.

The use of MOFScript to create PRISM models has worked well, but there are real challenges handling models with a reasonable level of complexity. We were only able to model simplified post-production workflows, focussing on small elements such as colour correction. To model larger post-production workflows we manually edited the PRISM (a time consuming process) to avoid restrictions that come with automatic mode translation.

Whilst we have not automatically transformed UML deployment and state diagrams into event triggers we believe this can be achieved simply by use of an XLS transform (from serialized UML XML to an event trigger configuration file) and a Matlab pre-processing code.

Our choice between using a stochastic model (e.g. PRISM Markov models) or discrete event simulation model (e.g. Monte Carlo simulations) depends on the complexity of the model, accuracy required and compute time needed. If the model is simple, and you have enough memory, the 'perfect' numerical solution offered by PRISM will probably be the best. If the modelling memory is not sufficient to hold the entire numerical matrix then numerical solution will not be possible, and if virtual memory is utilised the execution times will be prohibitively slow. We feel the simulation model is usually the correct choice if the model is complex, and is a good default option since it is an approach that scales well.

For a concrete comparison we have below execution times running our colour correction model for a numerical solution and Monte Carlo solution at 0.01 error with 95% confidence. These figures were generated using the PRISM model checker for direct comparison.

Table 1: Comparison of compute times for stochastic model and event simulation

	UML states (inc clocks)	Markov chain states	Transitions executed	Compute time	Error
Stochastic model	13	641241	829869	3 sec	n/a numerical solution
Discrete event simulation	13	641241	829869	20 sec	0.01 at 95% confidence

Compute hardware: CPU Intel Core2 Duo 2.4GHz, RAM 4GB, Windows XP

8. Conclusions

We have in this paper looked at practical ways we can use the UML 2.0 standard MARTE profile to model stochastic interactive application workflows. We use the PapyrusUML editor to create and serialized our UML to XML, making use of the PaStep, PaCommStep,

PaLogicalResource and GaCommHost MARTE stereotypes. We have investigated both stochastic models and discrete event simulation models, automatically serializing UML deployment and state machine diagrams and used this to automate the creation of our models. To create PRISM stochastic models from serialized UML we use an open source tool MOFScript. To create from serialized UML event triggers for a Monte Carlo discrete event simulation we propose using an XSL transform and pre-processing Matlab code. We give concrete examples of how we use MARTE to help other practitioners in the field.

Our choice between using a stochastic model (e.g. PRISM Markov models) or discrete event simulation model (e.g. Monte Carlo simulations) depends on the complexity of the model, accuracy required and compute time needed. In summary PRISM models are fast to execute if small and produce numerically accurate results. Discrete event simulation models are slower to execute but scale much better and are a good default solution for models of unknown complexity.

For future work we are looking next to implement the XSL transformation approach to automate the UML to event simulation interface, avoiding the costly manual model translation step. We also hope to investigate compound errors resulting from the aggregation of several deployment node state machines, as appears in many of our realistic use cases. Lastly we are looking at using instrumentation logs from our use case applications for real-time feedback, allowing the introduction of a learning algorithm to incrementally improve our models accuracy over time.

Acknowledgements

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

References

[1] Boniface, M. Nasser, B. Papay, J. Phillips, S. Servin, A. Yang, X. Zlatev, Z. Gogouvitis, S.Katsaros, G. Konstanteli, K. Kousiouris, G. Menychtas, A. Kyriazis, D. "Platform-as-a-Service Architecture for Real-Time Quality of Service Management in Clouds", Fifth International Conference on Internet and Web Applications and Services, ICIW '10 (2010)

[2] Digital Film Technologies Bones Digital Dailies (2010) http://www.dft-film.com/software/bones_dailies.php

[3] OMG. "Unified Modeling Language : Superstructure (version 2.1.1) Object Management Group, Inc.", Needham, MA 02494 (2007) OMG document number: formal/2007-02-05

[4] PapyrusUML. "Open Source Tool for Graphical UML 2 Modeling" (2010) http://www.papyrusuml.org

[5] MOFScript. "Open Source Tool for model to text transformation" (2010) http://www.eclipse.org/gmt/mofscript/

[6] PRISM. "Probabilistic model checker" (2010) http://www.prismmodelchecker.org/

[7] MARTE. "Modeling and Analysis of Real-time and Embedded systems" (2010) http://www.omgmarte.org/

[8] Mazzetti, A. "Virtual Cities of Art and Immersive Experience in Geo-referenced Communities", TiceMED - UBIQUITOUS LEARNING, Milano-Bicocca (2010)

[9] StarUML. "The Open Source UML/MDA Platform" (2008), http://staruml.sourceforge.net/en/

[10] Addis, M. Zlatev, Z. Mitchell, B. Boniface, M. "Modelling Interactive Real-time Applications on Service Oriented Infrastructures", NEM summit 2009