Capabilities of the UML Profile for Schedulability Performance and Time (SPT)

Murray Woodside and Dorina Petriu
Dept of Systems and Computer Engineering, Carleton University, Ottawa Canada
{cmw | petriu} @ sce.carleton.ca

April 2004

The authors are performance specialists with experience in software engineering, and contributors to the definition of the performance sub-profile of the UML SPT Profile [6]. Our viewpoint is that the profile should be usable in everyday practice by software developers. We have used it in a dozen examples and have found it largely effective. Our domain is distributed systems for telecom and business, with variable (stochastic) workloads and statistical real-time performance requirements, such as a deadline for a stated percentile of response completions. However, we have also considered schedulability of more deterministic systems. Here we comment on the strengths of the Profile, and additional capabilities that could be added to the next version.

1. The fundamental viewpoint of the SPT Profile

Performance annotations of a UML specification define two categories of information. Performance *parameters* describe the workload, the resource use and the behaviour of the program (they are inputs to a performance evaluation). Performance *measures* describe the performance itself, such as response delays, throughputs, utilization, or percentage of lost packets. They may be given as specified values, coming from the requirements analysis, or they may be performance predictions (the output of a performance evaluation). Specified and predicted values may both be defined for the same measure, and there could be more than one specified value (normal service, premium service) and more than one prediction (by different analysis methods, for instance).

In the SPT Profile, a *Scenario* is the unit of operation for which performance specifications and predictions are to be given; the duration of the Scenario defines what a performance engineer would call a *response*. Performance specifications are tags attached to a Step stereotype, which may be a Scenario, or a Step within it. Workload intensity parameters, and demands for resource usage, which are used in creating predictive models, and also attached to Steps.

Scenarios use the services of *Resource* entities, which have parameters such as service policy, multiplicity, and operation time, and measures such as utilization.

Performance analysis applies to instances rather than classes. Instances of objects are deployed, and execute. Different instances of the same class may have different behaviour depending on their role, or on the data they process, and the same object instance may have different behaviour in different scenarios.

2. Capabilities and limitations

We have found it straightforward to annotate a monolithic description of a response specified in a single sequence or activity diagram. Limitations may arise due to the semantics of the UML diagram, for instance sequence diagrams are weak in defining alternative and parallel branching of the scenario. Behaviour diagrams in UML 2 appear better for defining scenarios. In our examples (e.g., [9] [12] we modeled the following features of systems:

- deterministic sequences of Steps, via both Sequence (SD) and Activity (AD) diagrams,
- a sub-diagram to refine a Step, in Activity diagrams
- a repeated Step (loop), using a variable $N for the loop count, in a SD,
- probabilistic and parallel branching,
- processor resources indicated by the deployment diagram,
- process thread resources (single and multiple threaded) indicated on a SD
- a logical resource (a buffer pool) indicated on a deployment diagram, with the acquire and release stereotyped on messages in a SD,
- external operations (`<<PAextOp>>`) for network delays, disk I/O and database operations, and other interface operations (a video camera in [9][12], for instance).

We found that it is clearer to attach the Step stereotype to an action/execution-occurrence than to a message, since the occurrence is extensive in time, and so is a Step.

Sometimes it is required to attach a delay measure to a Scenario fragment that cannot be modeled as a Step, perhaps because it cuts across other delays modeled as Steps or Scenarios. This requires an ability missing in the current Profile version, to let the user define a delay measure between an arbitrary pair of events.

The Profile does not address annotations for the size of messages passed between processes, which sometimes introduce important delays related to message length. Another drawback mentioned by other authors is the lack of annotations on state machine behaviours [1][2].

Additional experience is described in papers [13] - [18]

Refinement

It is essential to be able to refine a Step by a separate diagram (easier in UML 2). This supplies detail for a high-level occurrence representing a complex operation by a subsystem or component. It is sometimes also necessary to refine a message, to describe the operations and workload of the middleware which handles it, as in platform specific operations in MDA. We have taken an approach to add the refinement to the performance model, under the name of "Model Completions" [11].

When there are alternative top-level scenarios, each one can be described separately in its own PAPerformanceContext diagram. To join them and describe the relative frequency, three approaches suggest themselves. (1) A high-level performance context could be defined which refines to alternative diagrams; this is easy in UML 2. (2) Use Cases could be linked to performance contexts and then a primary actor with a Workload stereotype could choose alternative Use Cases [4] (3) The scenarios can be combined in the performance model, leaving the relative frequencies outside the UML.

Quantities: Parameters and Measures

As mentioned above, *parameters* are the inputs to the analysis (known or assumed), and *measures* are the outputs or requirements on them. Parameter values describe workload intensity (e.g. arrival rate), behaviour (e.g. branching probabilities), and resource demands (e.g. CPU demand or the number of I/O operations required by a Step). Examples of performance measures include required, budgeted and estimated values of delays, throughputs and utilizations. The performance sub-profile uses the tag type `PAPerformanceValue` for most delays, whether it is a parameter or a measure, with modifying fields to indicate

- whether the value is assumed (for a parameter), estimated (for a measure) or measured (for either), and
- whether the value being given is a mean, variance or confidence interval, etc.

The modifying fields are an economical, powerful, and flexible mechanism. Unfortunately they are not provided for other measures besides delay. The new QoS Profile addresses the definition of these values, and should be exploited. However the open structure of modifying fields to provide interpretations of the numbers is still needed.

Variables for Parameters and Measures

A single diagram with fixed values for parameters is not enough for many analysis needs; there are typically many variations in the potential system which are most easily studied by solving the model with different parameter values. For this reason the SPT Profile supports symbolic variables, expressed as `$name`, as well as values for parameters. The same convention is used to support names for measures, to be filled by the analysis. This is extremely useful, and could be strengthened to accommodate:

- management of multiple cases with alternative values and results, represented as arrays, or tables. In our work (e.g. [12]) we have taken the parameter symbols into the performance tools, and created tables there. However it would be better to bring the results back, with suitable "case" identifiers.
- scoping of variable names to a given performance context or class of objects, to support more structured analysis. One advantage would be to allow separately defined scenarios to be brought together without name clashes for parameter names. If parameter names can be defined at the class level, then when many instances of the class interact, they could have different values referenced as `instance.$name`.

Designers will also wish to analyze variations in the selection of components and infrastructure; this is a challenging problem. This may be resolved in MDA by a platform-specific transformation layer.

Formal Definition of Performance Quantities

Performance analysis has its own terminology, and it is not perfectly standardized. For example three definitions of the response delay to a stimulus are:

- until the server first starts to work on the request,
- until the first response (such as the start of a web page download),
- until the initiator receives the complete response.

The QoS Profile has created an ontology for performance measures, based on some widely accepted definitions. Using the QoS Profile, each design object can be associated with its own QoSValue object, which has all its measures as `QoSSlotDimension` attributes. Given the open and evolving nature of the field, the definitions of measures should be kept open, so that new problem-specific measures can be used where they are needed. This argument is similar to the one that led to the introduction of standard extension mechanisms in UML.

In some systems it is necessary to define problem-specific measures. This can be done simply in terms of *delays* between *events*. In almost every case, a performance measure is either a delay between two defined events, or it can be calculated from delay measures. For example, between a start and end event, the delay is a response time or a resource holding time; between two successive repetitions of the same event, it is an inter-event time. Calculation of the inverse of such delays gives the throughput rate, and throughput multiplied by holding time gives resource utilization. Statistical measures such as mean, variance, coefficient of variation, percentiles, and various jitter measures are also defined by equations. The QoS Profile document itself defines some measures in this way, through events. We propose to allow the modellers to define derived performance measures by expressing them as functions of directly defined measure.

An open measure definition could be based on the `Duration` metaclass in UML 2, which identifies a start event, an end event, and a delay value. Parameter and measure values in the Profile could optionally be referenced explicitly or implicitly to one or more `Duration`, with mathematical expressions for derived measures.

Schedulability vs Performance Profiles

The sub-Profiles for schedulability and performance are separate, but it seems possible to combine them. The same Scenario structure underlies the analysis, and the same measures are used, such as duration for the delay of a Step. Schedulability analysis could use modifiers on some parameters and measures, such as: (1) worst-case values (as in, "worst-case execution time"), (2) special parameters of a task, such as its release time, its relative and absolute deadlines and laxity, and (3) special measures such as blocking time, pre-empted time. A combined Profile could also include relevant parameters for an action such as "is-atomic", and a description of scheduling. The stochastic behaviour parameters added for the Performance sub-Profile might be useful in recently developed approaches to stochastic schedulability.

3. Resource Descriptions

Resources are treated in considerable detail in the SPT Profile. For simplicity, unprotected resources could be treated as a special asymptotic case of protected resources through the "multiplicity" tag: an infinite multiplicity can be defined as unprotected.

Resource acquisition and release are key events in performance analysis, and sometimes need to be made explicit (as in [9][12]). This can be done with additional stereotypes on Steps (we used <<GRMresource Acquire>> and release in [9][12]).

4. The Central Role of Scenarios

Scenario-based analysis has been successful in all our studies, and reflects the standard practices in software performance engineering by professionals like Smith and Williams, who have developed their own scenario language for performance analysis [10]. Scenarios abstract away from the total system behaviour, to clearly identify the specific behaviour for which performance estimates are desired. It is straightforward to create analytic models from scenario specifications, as shown in [7], [8], [10] and many other works.

However in some situations the total behaviour may be required, for instance to show that all possible paths terminate before a certain time. Total behaviour is better defined by the state machines of the interacting objects, and [2] has investigated this use of performance annotations. These annotations are defined per class; instances have identical behaviour. A use of variable names for instances has been suggested above.

An advantage of combining state machines and scenarios, shown in [2], is that the state machine annotations can be used to refine a scenario. State machine submodels were composed with a scenario submodel, to add detail. Since the detail can be defined once and then combined with different scenarios, it can provide consistent behaviour refinement.

5. Conclusions

There appear to be opportunities to both generalize and simplify the SPT profile when adapting it to UML 2. Simplicity could be addressed by merging the schedulability and performance sub-profiles, and by harmonizing the definition of measures with the QoS Profile.

References

[1] S. Balsamo and M. Marzolla, "Simulation Modeling of UML Software Architectures," in Proceedings of the European Simulation Multiconference, Nottingham - UK, June 2003, pp. 562-567

[2] S. Bernardi, S. Donatelli, and J. Merseguer, "From UML sequence diagrams and statecharts to analysable Petri net models," in Proc. 3rd Int. Workshop on Software and Performance, Rome, July 2002, pp. 35-45.

[3] C. Canevet, S. Gilmore, J. Hillston, M. Prowse and P. Stevens, "Performance modelling with UML and stochastic process algebras", UK Performance Engineering Workshop, 2003.

[4] V. Cortellessa , R. Mirandola, "Deriving a queueing network based performance model from UML diagrams", Proc. 2nd Int. Workshop on Software and performance, p.58-70, Sept 2000, Ottawa, Canada

[5] G. P. Gu , D. C. Petriu, XSLT transformation from UML models to LQN performance models, Proc. 3rd Int. Workshop on Software and Performance, July 24-26, 2002, Rome, Italy

[6] Object Management Group, "UML Profile for Schedulability, Performance, and Time Specification," OMG Adopted Specification ptc/02-03-02, July 1, 2002.

[7] D. C. Petriu , H. Shen, "Applying the UML Performance Profile: Graph Grammar-Based Derivation of LQN Models from UML Specifications", Proc. 12th Int. Conf. on Computer Performance Evaluation, Modelling Techniques and Tools, p.159-177, April 14-17, 2002

[8] D.B. Petriu and M. Woodside, "Software Performance Models from System Scenarios in Use Case Maps," in Proc. 12th Int. Conf. on Modelling Tools and Techniques (TOOLS 2002), London, England, April 2002.

[9] D. C. Petriu and C. M. Woodside, "Performance Analysis with UML," in *UML for Real*, B. Selic, L. Lavagno, and G. Martin, Eds. Kluwer, 2003, pp. 221-240.

[10] C. U. Smith and L. G. Williams, *Performance Solutions*. Addison-Wesley, 2002.

[11] M. Woodside, D.B. Petriu, and K. Siddiqui, "Performance-related Completions for Software Specifications", Proc Int. Conf. on Software Engineering (ICSE 2002), May 2002.

[12] J. Xu, M. Woodside, and D. Petriu, "Performance Analysis of a Software Design using the UML Profile for Schedulability, Performance and Time," in Proc. 13th International Conference on Modelling Techniques and Tools for Computer Performance Evaluation (TOOLS 03), Urbana, USA, Sept. 2003.

Additional references to uses of the Profile:

[13] Cortellessa, V., D'Ambrogio, A., Iazeolla, G., Automatic derivation of software performance models from CASE documents, Performance Evaluation, 45, (2001) 81-105.

[14] T. Werdikt, B. Dhoedt, F. Gielen, "Incorporating SPE into MDA: Including Middleware performance details into System Models", Proc. 4rd Int. Workshop on Software and Performance, pp.120-124, Jan.14-16, 2004, Redwood Shores, CA, USA.

[15] A. DiMarco and P. Inverardi. "Starting from message sequence chart for software architecture early performance analysis". In 2nd Int. Workshop on Scenarios and State Machines: Models, Algorithms, and Tools, Portland, Oregon, USA, May 2003.

[16] James Skene and Wolfgang Emmerich, Model Driven Performance Analysis of Enterprise Information Systems, Electronic Notes in Theoretical Computer Science 82 No. 6 (2003)

[17] Zonghua Gu, Shige Wang and Kang G. Shin, “Issues in Mapping from UML Real-Time Profile to OSEK”, Proc SVERTS: Workshop on Specification and Validation of UML models for Real Time and Embedded Systems, Workshop at UML 2003, October, 20, 2003.

[18] Kirsten Berkenkötter, Using UML 2.0 in Real-Time Development A Critical Review, Proc SVERTS 2003