Open-perspective Modeling of Software Systems

Murray Woodside
Carleton University, Ottawa, Canada
1-613-520-5721
cmw@sce.carleton.ca

ABSTRACT
This paper investigates the question of improving a performance model by exploiting all the information that is known about a system. The modeling perspective is to be opened up to encompass the entire available dataset, which could make performance models a useful adjunct in big data analysis of performance. The scope and content of the model would not be limited by preliminary decisions made by a performance expert, but would be determined by the data itself. For this purpose it may be necessary to create multiple sub-models in various formalisms (in order to capture different kinds of mechanisms in the system), and to associate them. We consider how a decision can be reached to add another sub-model, and how it can be associated with the existing model. The paper is exploratory only; a principal goal is to identify fruitful research questions in this area.

General Terms
Measurement, Performance.

Keywords
Performance data, performance models, hybrid modeling.

1. INTRODUCTION
Performance models are required to predict the performance of a system that does not exist yet, or of a proposed deployment or reconfiguration of an existing system. However by their nature models abstract away from the full knowledge of the system and the available data on it, so that they ignore some features of this knowledge and data. As just one example, queueing models do not provide a representation of system data that can predict cache hit rates from the cache size, although these may be important for performance.

This paper considers a kind of "open-perspective" modelling process which can expand its base of assumptions and model formalisms to accommodate knowledge and data that is, or becomes, available. It identifies some possible approaches, some questions of feasibility, and some research problems to be addressed.

The benefits of a more open perspective are many. The use of models is inhibited by the limitations imposed by a narrow perspective, which depends on decisions made by someone who may not know best. A process that would build the best possible model given the available data, would not only give more accurate predictions, but would increase confidence that model represents the data and the prior knowledge of the system, rather than the biases and preferences of some modelling expert.

ICPE '17 Companion, April 22-26, 2017, L'Aquila, Italy

DOI: http://dx.doi.org/10.1145/3053600.3053651

2. RELATED WORK
This section examines various ways in which models of different types have been combined for performance analysis.

Curve-fit modeling has been used to model the dependence of parameters of queueing models, on other parameters. Vetland et al. [12] increased the fidelity of queueing models of software by incorporating "resource functions" that model the computing demand of a software operation, based on parameters of the operation (for example, the demand of a "sort" operation depends on the size of the data to be sorted). These functions are a kind of empirical complexity function for the operation, and may take the form of a regression or regression spline function as in [13]. The computing demands are calculated first, and inserted in the queueing model. Pllana et al. [9] similarly used mathematical expressions as resource functions in a simulation model. With the same goal but a different method, Balbo et al used Petri Net submodels to find the service times of queues in [1], and Nelson et al. used queueing models to determine probability distributions for service delays, which were then used in a simulation model of a complex I/O system [8].

Some multi-formalism or "hybrid" performance models combine separate performance models with complementary capabilities (for instance, a queueing model and a finite-state-machine model), with a fixed-point iteration between them. Wu combined Petri nets representing exception mechanisms with layered queues [15]. Verdickt et al. combined layered queues of a software application with a simulation of a network and its protocols [11]. Marco et al combined simulation and queueing models in [7]. The iteration approach has also been used to combine separate models made within the same formalism (especially for Petri net models), to simplify the solution and overcome solver scalability limitations.

General tools for hybrid modeling have been described. The SMART toolset combines simulation with state-based models [4]; the SHARPE toolset [10] combines a variety of analytic model formalisms.

A more narrowly defined multi-formalism is the basis of the Mobius toolset [6], which employs a set of different but closely related formalisms that all define finite-state Markovian models (Mobius is also capable of combining additional model types by fixed-point iteration).

3. THE INFORMATION BASE FOR PERFORMANCE PREDICTION
Prediction using a performance model is based on two broad categories of information: knowledge of the software architecture

and deployment, and of the hosting hardware (processors and networks), which we will call *system knowledge*, and data from tests and operations, which we will call "data". Data includes:

1. Settings in the hardware and software environment that might influence performance (buffers, timeouts, priorities) (*input data* to the model-making).
2. Event traces (part of *operating data*)... a large category
   a. externally generated events (e.g. user requests, instrumentation events, failures)
   b. message-related events (e.g. sending, receiving, replying)
   c. execution-related events (e.g. beginning, suspending, resuming, ending)
   d. memory related events (e.g. allocate, de-allocate, cache hit/miss)
   e. exception events (e.g. time-outs)
3. State traces, meaning samples of subsystem states at time intervals or defined instants (busy/idle, number of active threads, heap size, occupancy of cache, size of buffer or executable); averages related to state occupancy (also part of *operating data*).
4. Sizes of messages, and data and program entities: samples and averages (*operating data)*.
5. Statistics drawn from operating data, such as average event counts for user requests, instrumentation events, or messaging events, mean occurrences of a given state or state sequence, or statistics of data and message sizes (additional *input data* to model-making).
6. Some of the operating data and statistics can be identified as p*erformance measures* important to the prediction process, such as response times, message delays, queue lengths, or failures to meet a QoS specification.

For analytic models the performance measures and input data will all be statistics over the measurements (such as averages), but for simulation modeling trace data may also be used, as in trace-driven simulation. The system knowledge usually determines the structure of the performance model, and the input data is used to calibrate its parameters in making a performance prediction, as illustrated in Figure 1.

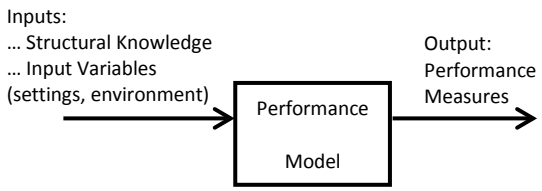

Figure 1 Data and Models

4. CLASSES OF MODELS

The models we are considering predict some performance measures as functions of some input variables as illustrated in Figure 1. The models fall into these six broad classes:

Data reports such as scatter plots which represent the observed outcomes for the measures, in a multidimensional space of input variables. To predict performance, one looks up the input value and takes the set of observations at those values, or its mean or median, as the prediction. These are the closest to the original data, but only predict based on previous experience of a configuration..

Simulation, meaning an instrumented execution of an abstraction of the system mechanisms. These recreate a version of the operating data.

Fitted curves and surfaces using standard forms such as polynomials and splines, including linear and nonlinear regression functions, adaptive regression splines (MARS) and regression trees. These give a single prediction of performance measures for each input value, and using extrapolation they can predict performance for input values which have not been measured.

State-based models, representing deterministic or stochastic transitions between states (including stochastic automata, Petri Nets with time (which have many forms), Stochastic process algebras, and any other finite-state model with time, transitions, and a method such as stochastic transitions for generating execution traces). These models may in turn be simulated, or transient or steady-state solutions for performance measures can be computed numerically.

Queueing models, representing jobs using resources and making transitions between resources, and including many kinds of extended queueing models such as layered queueing networks. These models may be solved by simulation, or transient or steady-state solutions can be found analytically or with numerical methods.

Fluid models, representing numbers of entities in a given state as continuous variables with flows between states modeled as flows of a fluid. For example there are fluid versions of both Petri Nets and queueing models.

The list is organized in increasing order of abstraction away from the original system and its measurements. Additional classes of models may also exist.

Figure instead? The tradeoff is immediately apparent. Data reports can represent any observed data, and are used in big data analysis for performance, but they cannot be extrapolated. Thus they summarize experience, and are totally faithful to the data within this limitation, but have limitations as predictors. Fitted surfaces can be used for interpolation, for input values that have not been measured but are surrounded in some sense by measured values, but are weak for extrapolation. The remaining models are all causal models, meaning that they represent the mechanisms in the system in some way. They offer possibilities for extrapolation, but the quality of extrapolation depends on how well the model captures the mechanisms of the system.

The process of creating a model requires first the selection of a model class, second creation of a structure from system knowledge, and finally calibration of the model parameters to match the operating data including some of the performance measures.

5. OPEN PERSPECTIVE MODEL CONSTRUCTION

Normally one class of models is chosen for a modeling study, which limits the system mechanisms that can be modeled. To obtain a more open perspective, *there must be an approach to expanding the model classes, when there is a need and opportunity to do so*.

We can see a move in this direction, in extended queueing and more generally in hybrid modeling, where an additional model type may be added to an existing model, to describe a mechanism that cannot be encompassed in the original. For still wider

perspective, we consider here how the data itself can drive the opening of perspective. If we can do this, we can arrive at modeling that is truly data-driven, rather than modeller-driven, although expert judgment would still be part of the process. The overall process of making models and improving them is summarized, in a very condensed way, as:

Step 1. From system knowledge, choose a model class and formalism from those available.

Step 2. Fit it to data and observe the goodness of fit, to give an initial model M. Optionally generate stress tests based on the model, to improve the accuracy of the model parameters and extend the domain of the model.

Step 3. If (a) the fit is inadequate or (b) M ignores a mechanism or some data that is felt to be important for extrapolation, attempt a perspective expansion, else stop.

This paper is concerned with the last step.

5.1 Perspective expansion

If we assume that M already makes maximum use of the class of models it is based on, then a perspective expansion should seek opportunities to involve additional classes of model. The following process is an initial proposal for doing this, and is the core of the present paper. There are four stages: *Screen* for variables, *Choose* a model class, *Associate* it with the existing model, and *Fit* the parameters of the combined model.

Screen: Consider the input variables that were ignored in fitting the model. Screen them for correlation with the model fitting errors, to create a candidate set of variables that could help expand the model.

To investigate correlation a sequence of inputs and predictions are used. For input variables that are statistics over an operational period, there are two options:

Break the operational period into subperiods, find the statistics, measured performance and the model prediction for each subperiod (the model could be re-fitted to the sub-period data), and find the correlation between the values for each subpeiod.

Input variables that are traces can be considered a special case of the same approach, in which each trace point is treated as a "sub-period". For trace data and a simulation model the predictions may be available for every trace point; in other cases the model prediction can be based on the entire data (i.e. on the original model).

The output of this stage is a set of data variables that are good candidates for explaining the model errors. The next problem is to create a new model element, called a "candidate submodel" C, that captures this incremental effect.

Choose (1) the input variables with high correlation; (2) candidate system mechanisms that might express the effect of these variables on performance, using system knowledge; (3) potential candidate submodels for these mechanisms.

Associate the candidate sub-model C to the model M. If it is expressed in the same formalism as M. then C can simply be added to M; the extended model is just an enhancement of M. Otherwise, a suitable form of association must be found. A form which is suitable in many cases has often been described for hybrid modeling, called here Loose Association, is described below. The result is a single model (M + C) which computes predictions based on an enlarged set of input variables.

Fit (M + C) to the measured data. The parameters of M must in general be re-fitted because of the effect of C.

A model-fitting process with sufficient generality for all model types is the nonlinear regression process described in [14], which only uses the combined model to compute predictions for sets of input variable values. However it is not suitable for a model that can only be solved by simulation.

5.2 Loose Association of Sub-models

In a loose association the sub-models are kept separate and are coordinated by passing variables through their input interfaces. This is convenient for models of different formalisms, since the solver software is typically separate and not eady to combine in other ways. Loose association has four forms, depending on the form of interdependence between M and C, represented as follows:

M <≠> C, or *independent submodels*: neither model depends on the other, so the two models are used together. Example: a reliability model C added to a performance model M for non-failed operation only. C gives failure probabilities, M gives performance when not failed (a rather primitive combination).

C => M, or *M embedded in C*. M depends on knowing the solution of C but not vice versa. C is solved first, and the solution is used to calibrate or modify M. Example: a dependability model in which the probability of the different failure states is calculated first by a failure model C, then the performance in each of the failure states is found by a version of adapted to the failure state, with the dependability measures found by combining the information..

M => C, or *C embedded in M*. C depends on knowing the solution of M, but not vice versa. M is solved first, and then C. Example: An analytic model of one system determines the rate of arrivals to a separate sub-system which requires a state-based model.

M <=> C, or *interdependent models*. Each model depends on the solution of the other. An initial set of outputs for one model is assumed, and the other is solved. Then the models are solved in turn in a fixed-point iteration (eg. [7]), which succeeds if it converges (not guaranteed, but in many cases successful).

The data flow in the four kinds of loose association is illustrated schematically in Figure 2.

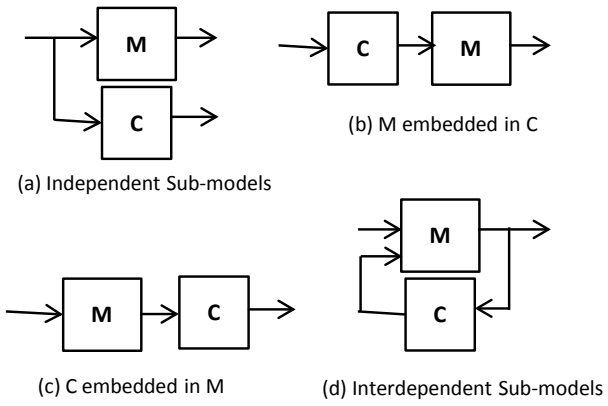

Figure 2. Forms of the Loose Association in (M + C)

For cases (a) (b) and (c) in Figure 2, Loose Association is sufficient for any combination of sub-model classes. For interdependent sub-models loose association can be applied, but it has weaknesses.

5.3 Other Associations for Interdependent Sub-models

Loose association may sometimes give less-than-ideal modeling accuracy when the mechanisms behind the sub-models M and C are tightly coupled. In these cases the exchange of results is too coarse a way to capture their interactions. However this is a subject that has not been studied much.

As an example consider a case in which M is a mean-value queueing model that predicts response times, given an arrival rate. The service time is fitted to response time data using the approach of [14]. C is a state-based model for an adaptive mechanism that allocates a second CPU core to the application, to execute the server when the system is overloaded. Using the average extra power added to the system to modify the model for all time periods appears to be a poor way to represent the performance effect of this mechanism, since the full extra power is available when it is needed..

A second approach to association which could be suitable for this example, is to first upgrade M to a more detailed modeling formalism which represents the states of the queueing system, in which the overloaded state corresponds to a set of queueing states, and combine C and M within the same more detailed formalism.

Additional methods for associating sub-models of different formalisms appears to be a significant candidate for research.

6. AN EXAMPLE

A small and artificial example will illustrate the main points of this suggested procedure. The execution of a computer program is modelled by the simplest possible queueing model, M is an M/M/1 queue (single server, exponential service distribution, random (Poisson) arrivals). The mean service time is fitted to a trace of arrivals and response times by taking averages of A and R over a number of trace sub-intervals, and using the nonlinear regression approach of [14]. The model prediction equation is then:

$$R = S/(1 - AS)$$

where R is the average response time, S the fitted average service time, and A the average arrival rate of requests to execute the program. Data was generated for a set of 20 intervals, for a system represented by such a queue with actual S = 10 ms. However the actual system also had additional application programs sharing the processor. The best fit found for S was 0.014, 40% too large. This value could be misleading for capacity analysis, as it will predict the system has a lower capacity than it really does.

In considering a more open perspective, suppose the existence of the other programs is known, and the available data includes the number K of other application processes which are active at each trace point. The correlation coefficient of the model prediction error for R, with K, was found to be -0.8, indicating strong negative correlation. A scatter plot for this data is shown in Figure 3. To make the plot, the data was divided into sub-intervals during which K was constant, and averages were taken over these subintervals to give R and A.

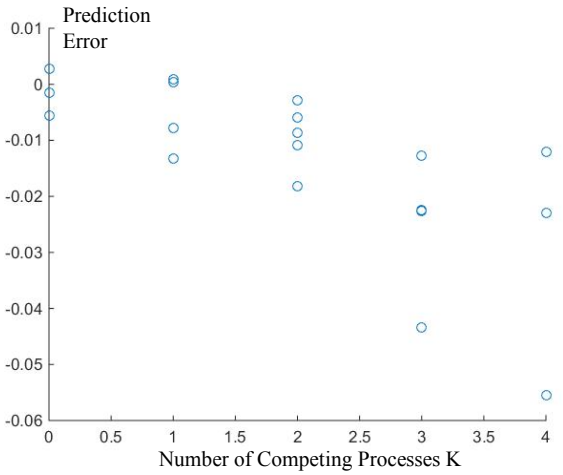

Figure 3 Scatter Plot of Prediction Error vs K

The additional mechanism being modeled is the extra contention, expressed through additional utilization in the model, not due to the arrival stream that is monitored. The form of the sub-model C for this mechanism is chosen to be a fitted linear function giving the additional utilization U(K) as a function of K:

$$U(K) = a + bK$$

Other curve-fits are equally possible. To make a performance prediction for a given K, U(K) will be calculated first and then M will be solved using service time S and U:

$$R = S/[1 - AS - U(K)]$$

This embeds M in C, like the configuration in Figure 2(b).

The parameters a, b and S in the combined model are finally fitted to the data in the scatter plot, using the overall average arrival rate. The fit is much better, as shown by the comparison of the sum-of-squared-error vs S, for M and for M + C, in Figure 4. The fitted values of S are the values at the minimum of the two curves, and are around 13.7 ms for M and around 10.2 ms for M + C.

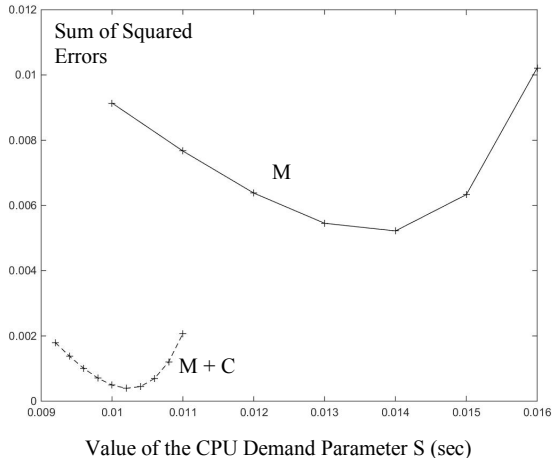

Figure 4. Comparison of Goodness of Fit of M and M + C

The fitted value of S for M + C is more nearly correct, with some statistical fitting error, and the sum-of-squared-error is less by a

factor of nearly 20 (in Figure 4 the exact values for the function for C were used).

This example just shows the steps to be taken, and indicates the potential for improvement, using the steps for perspective-opening described above. Application to real system data is the next step.

7. CONCLUSIONS

A systematic approach suitable for basing a performance model on large datasets, and for augmenting the model to exploit all the relevant data, appears to be practical. Such an approach would add performance models to the methods usable in "big data" analysis of performance. The main benefit of using these models would be an ability to extrapolate from the measured conditions. This work has only drafted out an approach, and has identified some gaps for which research would be beneficial. A principal gap is a need for ways to combine interdependent sub-models in different formalisms, in cases where the sub-models are heavily inter-twined.

8. ACKNOWLEDGMENTS

This research was supported by the Natural Sciences and Engineering Research Council of Canada through its Discovery Grant program.

9. REFERENCES

[1] G. Balbo, S.C. Bruell, et al., "Combining queueing networks and generalized stochastic Petri nets for the solution of complex models of system behavior", IEEE Trans. on Computers, v 37(10), pp 1251-1268, 1988.

[2] H. Beilner, Mater J., Wysocki C., "The Hierarchical Evaluation Tool HIT", Short Papers of the 7th Int. Conf. on Modelling Techniques and Tools for Computer Performance Evaluation, pp 6-9, 1995.

[3] G. Bolch, S. Greiner, H. de Meer, KS. Trivedi, :Queueing Networks and Markov Chains", Wiley 1998. (see Section 12.4 on SHARPE).

[4] G. Ciardo, Miner A. S. (1996). "SMART: simulation and Markovian analyzer for reliability and timing", Proc. IEEE Int. Computer Performance and Dependability Symposium, 1996.

[5] G. Ciardo, Jones I. R. L., et al., "Logical and stochastic modeling with SMART", Performance Evaluation, v 63(6): 578-608, 2006.

[6] T. Courtney, S. Derisavi, S. Gaonkar, M. Griffith, V. Lam, M. McQuinn, E. Rozier, and W. H. Sanders, "The Möbius Modeling Environment: Recent Extensions", Proc. 2nd Int. Conf. on the Quantitative Evaluation of Systems (QEST 2005), Torino, Italy, Sept. 2005, pp. 259-260.

[7] G. Marco, Mazzocca N., et al., "Multisolution of complex performability models in the OsMoSys/DrawNET framework" Second Int. Conf. on the Quantitative Evaluation of Systems (QEST'05), pp 85-94, 2005.

[8] B.L. Nelson, W. S. Keezer, T.F. Schuppe, "A hybrid simulation-queueing module for modeling UNIX I/O in performance analysis", Proc. 1996 Winter Simulation Conference, pp 1238-1246, 1996.

[9] S. Pllana, S. Benkner, F. Xhafa, L. Barolli, "Hybrid Performance Modeling and Prediction of Large-Scale Computing Systems", Proc. Int. Conf. on Complex, Intelligent and Software Intensive Systems, March 2008.

[10] R. A. Sahner, K. Trivedi, "Reliability modeling using SHARPE", IEEE Trans. on Reliability, R-36(2), pp 186–193, June 1987.

[11] T. Verdickt, B. Dhoedt, F. D. Turck, and P. Demeester, "Hybrid Performance Modeling Approach for Network Intensive Distributed Software," in Proc. 6th Int. Workshop on Software and Performance (WOSP'07), Buenos Aires, 2007, pp. 189-200.

[12] Vidar Vetland, Peter H. Hughes, Arne Sølvberg: A Composite Modelling Approach to Software Performance Measurement. SIGMETRICS 1993: 275-276

[13] M. Woodside, V. Vetland, M. Courtois, S. Bayarov, "Resource Function Capture for Performance Aspects of Software Components and Sub-Systems", in "Performance Engineering: State of the Art and Current Trends", ed. Reiner Dumke, Springer LNCS 2047, pp 239-256, 2001.

[14] M. Woodside, "The Relationship of Performance Models to Data", Proc SPEC Int. Workshop on Performance Evaluation (SIPEW), Darmstadt, Springer Lecture Notes in Computer Science, Vol. 5119, pp 9 - 28, June 2008.

[15] Pengfei Wu, "Extending Layered Queueing Network with Hybrid Submodels Representing Exceptions and Decision Making", PhD thesis, Carleton University, May 2013.