Scaling Size and Parameter Spaces in Variability-aware Software Performance Models

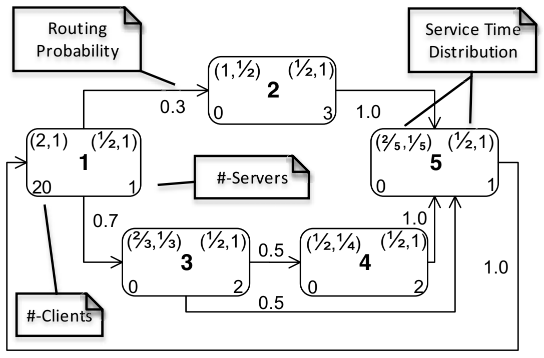
Matthias Kowal[1] Max Tschaikowski[2] Mirco Tribastone[3] Ina Schaefer[4]

Abstract: Model-based software performance engineering often requires the analysis of many instances of a model to find optimizations or to do capacity planning. These performance predictions get increasingly more difficult with larger models due to state space explosion as well as large parameter spaces since each configuration has its own performance model and must be analyzed in isolation (product-based (PB) analysis). We propose an efficient family-based (FB) analysis using UML activity diagrams with performance annotations. The FB analysis enables us to analyze all configurations at once using symbolic computation. Previous work has already shown that a FB analysis is significant faster than its PB counterpart. This work is an extension of our previous research lifting several limitations.

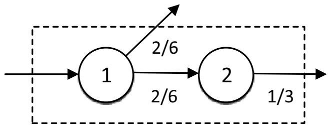
1 Coxian Distributions and PB-Evaluation of PAADs

Performance Annotated Activity Diagrams (PAAD) capture the workflow of a software system and enhance it with performance-related properties [KST14]. An example can be found in Fig. 1a. Each node represents a service center in the software system, e.g. CPUs, web server and so forth, and has the following performance annotations at its corners: vectors for the service time distribution (top left and right values), number of clients at that node during the initial condition (bottom left) and number of servers (bottom right). Edges connect the nodes and are annotated with probabilities denoting the likelihood of a job to take that path. We can construct a continuous-time Markov chain (CTMC), where the length of either vector denotes the actual number of states in the CTMC (or stages of the distribution). The left vector provides the rate of the exponential residence time at each state, while the right vector contains the probability with which a service process moves from one state to the next. The time between entering the first state and exiting from any other state gives us a non-exponential distribution for the service at the specific node. Fig. 1b shows the CTMC for such a Coxian distribution. The services will be exponentially distributed with $2/6+2/6=2/3$ in state 1 and enter state 2 with a probability of $1/2$ encountering an additional delay of $1/3$. Coxian distributions provide better representations of real-world software systems, since they can be seen as a composition of exponential stages and are able to approximate any given general distribution [St09]. In addition, we can now simulate parallelism with multiple servers that are available at a node. Both aspects remove a restriction of our previous work in [KST14]. The calculation of the steady state throughput for such Coxian-distributed multi-server nodes is a non-trivial task that involves solving the system of Ordinary Differential Equations (ODE) given by $R^T T = T$. R is the routing probability matrix and T determines the ODE throughputs. In the PB analysis, we have to solve it for each variant in isolation, which is inefficient. Mean service times as well as number of servers and clients also play role in the calculation of T, but their relation to the ODE system is omitted here.

[1] Technische Universität Braunschweig, Germany
[2] IMT Institute for Advanced Studies Lucca, Italy
[3] IMT Institute for Advanced Studies Lucca, Italy
[4] Technische Universität Braunschweig, Germany



(a) A Performance Annotated Activity Diagram



(b) Coxian CTMC for node 3.

Figure 1: Running Example

2 Variability and FB-Evaluation

A FB analysis is only reasonable if variability is included into the PAADs. Similar to our previous work, we applied the principle of delta modeling (DM) in which deltas can *add*, *remove* or *modify* PAAD elements. Given a specific *core* PAAD, we can generate any variant of the system by applying the respective deltas [Sc10]. The FB analysis relies on the construction of a 150%-model or super-variant. This model is built by merging the core and all deltas into one large model. Each PAAD element that is changed by a delta is represented as a symbol in the 150%-model and not its concrete value, e.g. removing the edge between node 3 and 5 would modify the probability between node 3 and 4 to 1.0 (cf. Fig. 1a) and result in two symbolic parameters in R. Again, we can construct the ODE system, but solve it symbolically this time, which has to be done just once. The steady state throughputs T are now calculated by plugging the concrete values for the desired variant into the parametrized expressions. The FB analysis is faster compared to the PB one and gets even more efficient for larger networks or an increasing number of variants.

References

[KST14] Kowal, Matthias; Schaefer, Ina; Tribastone, Mirco: Family-Based Performance Analysis of Variant-Rich Software Systems. In: FASE 2014. pp. 94–108, 2014.

[Sc10] Schaefer, Ina: Variability Modelling for Model-Driven Development of Software Product Lines. In: VaMoS. pp. 85–92, 2010.

[St09] Stewart, William J.: Probability, Markov Chains, Queues, and Simulation. Princeton University Press, 2009.