Family-Based Performance Analysis of Variant-Rich Software Systems

Matthias Kowal[1], Ina Schaefer[1], Mirco Tribastone[2]

Institut für Softwaretechnik und Fahrzeuginformatik[1], Electronics and Computer Science[2]
TU Braunschweig[1], University of Southampton[2]
Mühlenpfordtstr. 23, Highfield Campus
D-38106 Braunschweig, UK-SO17 1BJ Southampton
{m.kowal, i.schaefer}@tu-bs.de
m.tribastone@soton.ac.uk

Abstract: The analysis is meant for behavioral models of workflow-type software systems such as data centers or automation systems. We model these systems as a UML activity diagram with performance annotations to compute a performance prediction. A product-based (PB) analysis is not always viable, since we can have several variants, and each variant has its own performance model, which is why we cannot reuse the results of a single variant. Hence, we propose a family-based (FB) performance analysis relying on symbolic computation. The FB analysis can be significantly faster than PB analysis, especially for large-scale workflow model, thus enabling efficient calculation of large parameter spaces.

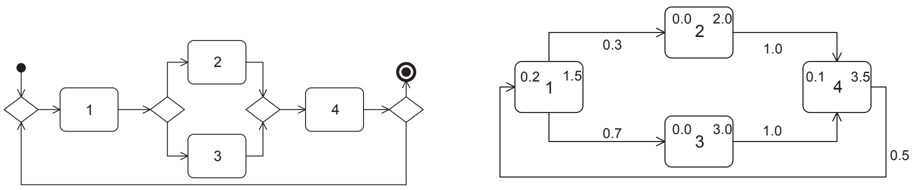
1 PAADs and PB-Evaluation

Workflow-type software systems can be visualized as Activity Diagrams (AD). Fig. 1 shows a sample and its representation as Performance-Annotated AD (PAAD). Nodes that are not supported, e.g. initial and decision nodes, are removed and the edges appropriately redirected. Each node has two additional values with arrival rate (top-left) and service rate (top-right) and edges are labeled with a probability depicting how likely it is that a job takes a certain path in the system. Such a PAAD is interpreted as continuous-time Markov chain that underlies a Jackson-type queuing network [Jac63]. The product-based evaluation is given by the following system of linear equations.

$$(I - P^T)\gamma = \lambda, \tag{1}$$

P is the routing matrix, defining with which probability a job moves from one node to another after it is processed at the present node. Once the system is solved for γ, we have fully characterized the steady-state behavior of the network and we can compute e.g. the throughput and average queue length (AQL). The steady-state behavior for Fig. 1 is given by

$$\gamma = \begin{bmatrix} 0.50 & 0.15 & 0.35 & 0.60 \end{bmatrix}^T.$$



(a) Sample Activity Diagram (b) Annotated PAAD

Figure 1: Running example

Solving the above system must be done for each variant separately in PB evaluation, which is time consuming.

2 Variability and FB-Evaluation

A FB approach requires that variability is included into the PAADs. As a solution, we apply the principle of delta modeling (DM) to the modeling process to manage variability [Sch10]. In DM, a system has a core variant and several deltas. Each delta contains information about adding, removing or modifying PAAD elements. A new variant of the system can be generated by applying a single delta or delta sequence, containing all elements to get to the desired variant, to the core variant. The following delta removes node 2 in Fig. 1 as well as the connected transitions.

$$\delta = \{rem\ (1, 0.3, 2), rem\ (2, 1.0, 4), rem\ 2, mod\ (1, 0.7, 3)\ by\ 1.0\}$$

DM enables us to construct a 150%-model by merging all deltas and the core. This super variant is not necessarily a valid configuration of the system. Each element that is changed by a delta is represented with a parameter instead of the concrete values in the 150%-model. The FB-evaluation solely relies on the 150%-model. Analogous to (1), we can solve the system of linear equations, but this time it is done symbolically and just once. The AQL for single variants is now computed by plugging the specific values into the parametrized expression. Thus, enabling us to skip the step of solving the system numerous times as in PB-evaluation leading to speed-ups of up to 2000%.

References

[Jac63] James R. Jackson. Jobshop-like Queueing Systems. *Management Science*, 10(1):131–142, 1963.

[Sch10] Ina Schaefer. Variability Modelling for Model-Driven Development of Software Product Lines. pages 85–92, 2010.