A Proactive Approach for Runtime Self-Adaptation Based on Queueing Network Fluid Analysis

Emilio Incerto[1] Mirco Tribastone[2] Catia Trubiani[1]

[1]Gran Sasso Science Institute, L'Aquila, Italy
[2]IMT Institute for Advanced Studies, Lucca, Italy

1st International Workshop on Quality-Aware DevOps (QUDOS) September 1, 2015

Outline

Motivations

- In software development process the fulfillment of performance requirements is a very important goal
- In most application domains performance evaluation is critical even at design time
- Furthermore, run-time variability makes the process of devising the needed resources challenging

Motivations

- In software development process the fulfillment of performance requirements is a very important goal
- In most application domains performance evaluation is critical even at design time
- Furthermore, run-time variability makes the process of devising the needed resources challenging

- **Research question:** How to fulfill performance requirements while considering run-time variability?

The Idea

- Self-adaptation is a promising technique
- It consists in finding at run-time the most suitable system configuration that preserves the functional behavior while meeting performance requirements
- We propose a **proactive** run-time self-adaptation approach based on **fluid approximation** of **queuing networks**

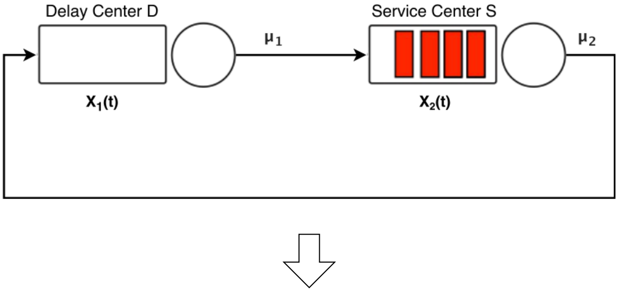
The Idea

- Self-adaptation is a promising technique
- It consists in finding at run-time the most suitable system configuration that preserves the functional behavior while meeting performance requirements
- We propose a **proactive** run-time self-adaptation approach based on **fluid approximation** of **queuing networks**
- The idea is to devise at run-time the most suitable system configuration relying on efficient transient analysis of a QN model, fed with the actual system parameters

Background: Fluid approximation

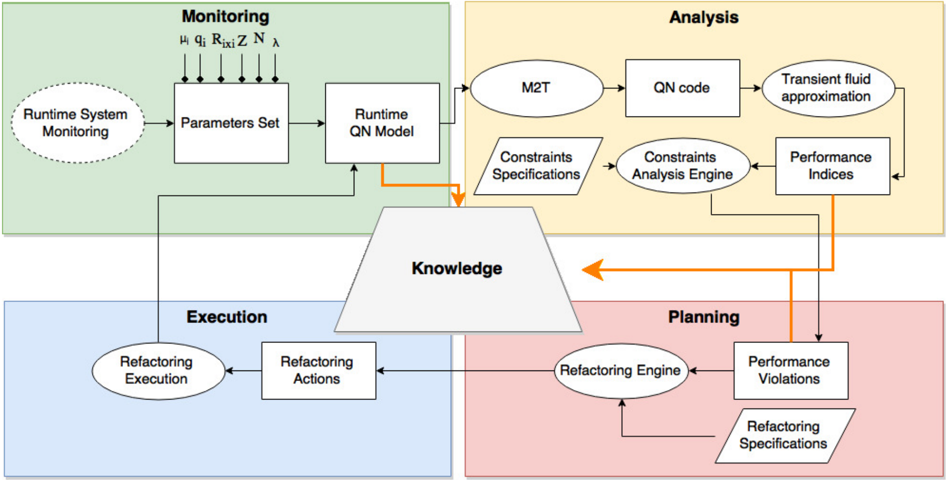
- The goal of this technique is to speed up the analysis of transient dynamics of queueing networks models
- Basically it consists in translating a QN model in a system of Ordinary Differential Equations (ODEs)
  - Each equation analytically describes the evolution of the queue length at each service center
  - Then, solving these equations, we are able to derive the performance indexes of interest

Background: Fluid approximation

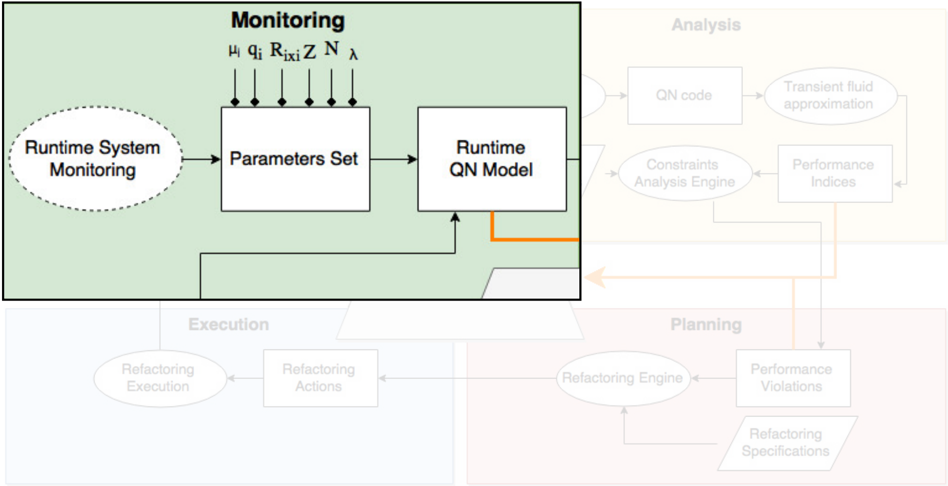


$$\frac{dx_1(t)}{dt} = -\mu_1 x_1(t) + \mu_2 \min(1, x_2(t))$$
$$\frac{dx_2(t)}{dt} = +\mu_1 x_1(t) - \mu_2 \min(1, x_2(t))$$

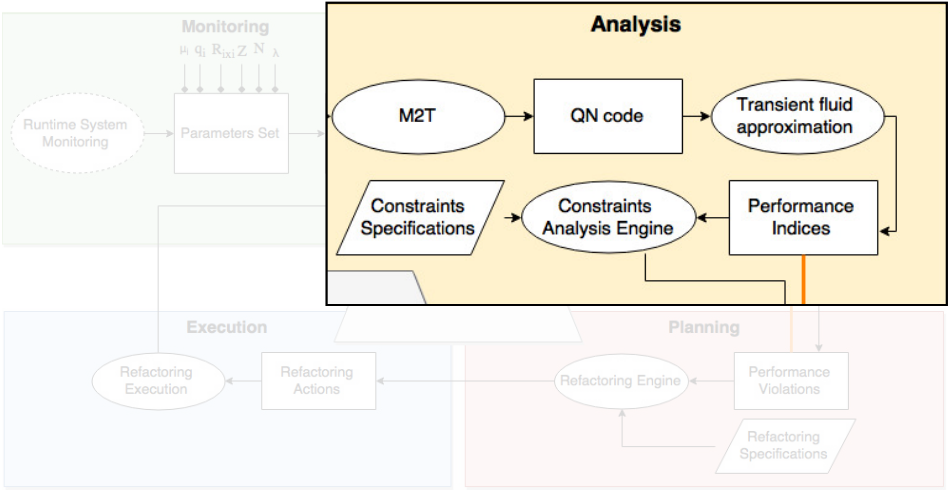
Our Approach



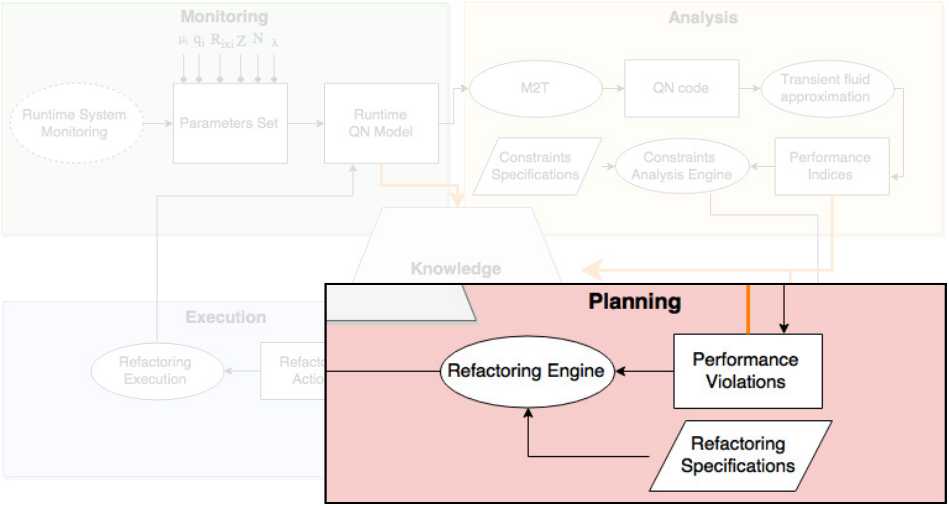
Our Approach: Monitoring Phase



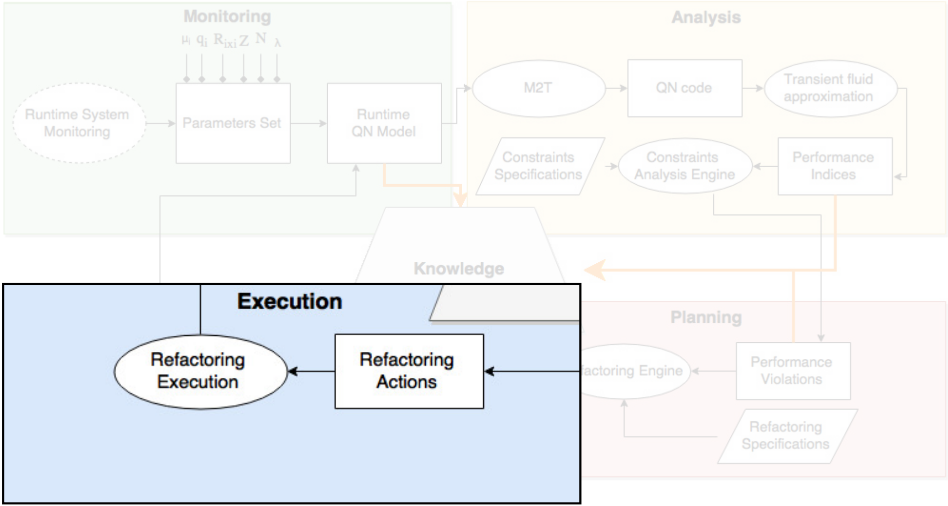
Our Approach: Analysis Phase



Our Approach: Planning Phase



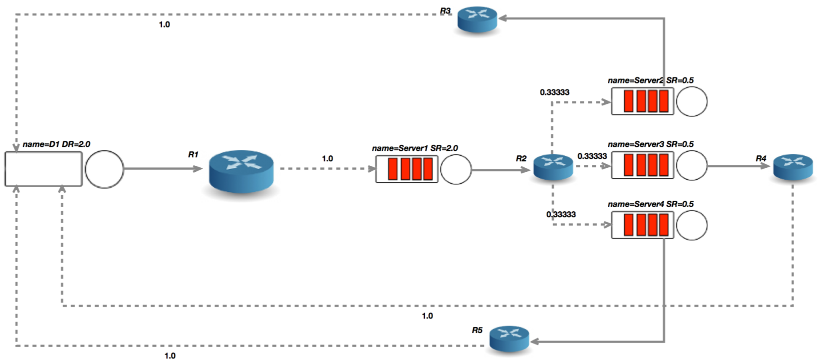
Our Approach: Execution Phase



Illustrative Example

- We consider a constraint model requiring that the percentage of jobs in the queue of every center does not exceed 0.5% of the total jobs population
- We developed an Eclipse based tool for QN models definition and M2T transformation execution. `http://sourceforge.net/projects/qnml/`

Illustrative Example: Monitoring



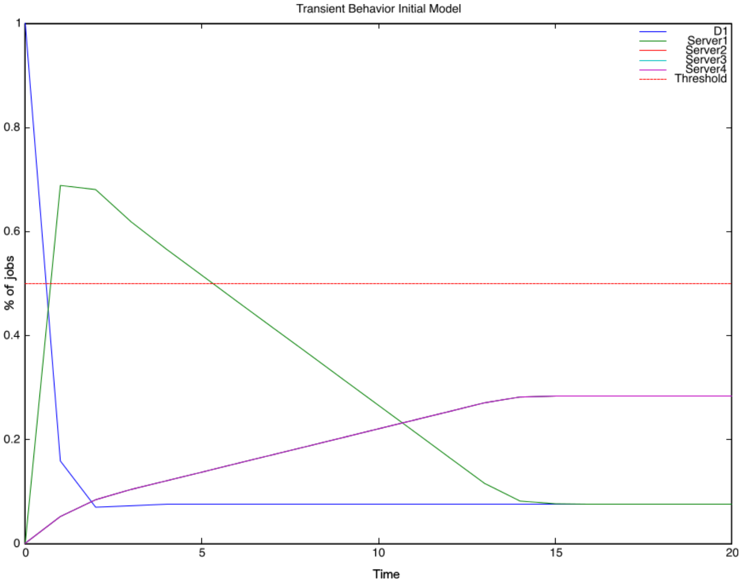
$$\begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1/3 & 1/3 & 1/3 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Station	Init. Pop.	μ_i	Z
D1	10	n.d.	0.5
Server1	0	2.0	n.d.
Server2	0	0.5	n.d.
Server3	0	0.5	n.d.
Server4	0	0.5	n.d.

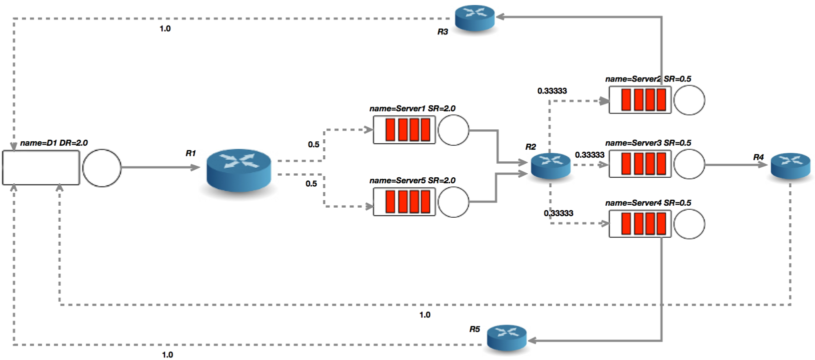
Illustrative Example: Analysis

$$
\begin{aligned}
\frac{dx_1}{dt} &= -\mu_1 x_1(t) + \mu_3 \min(x_3(t), 1) + \mu_4 \min(x_4(t), 1) \\
&\quad + \mu_5 \min(x_5(t), 1); \\
\frac{dx_2}{dt} &= +\mu_1 x_1(t) - \mu_{2,1} \min(x_2(t), 1) - \mu_{2,2} \min(x_2(t), 1) \\
&\quad - \mu_{2,3} \min(x_2(t), 1); \\
\frac{dx_3}{dt} &= +\mu_{2,1} \min(x_2(t), 1) - \mu_3 \min(x_3(t), 1); \\
\frac{dx_4}{dt} &= +\mu_{2,2} \min(x_2(t), 1) - \mu_4 \min(x_4(t), 1); \\
\frac{dx_5}{dt} &= +\mu_{2,3} \min(x_2(t), 1) - \mu_5 \min(x_5(t), 1);
\end{aligned}
$$

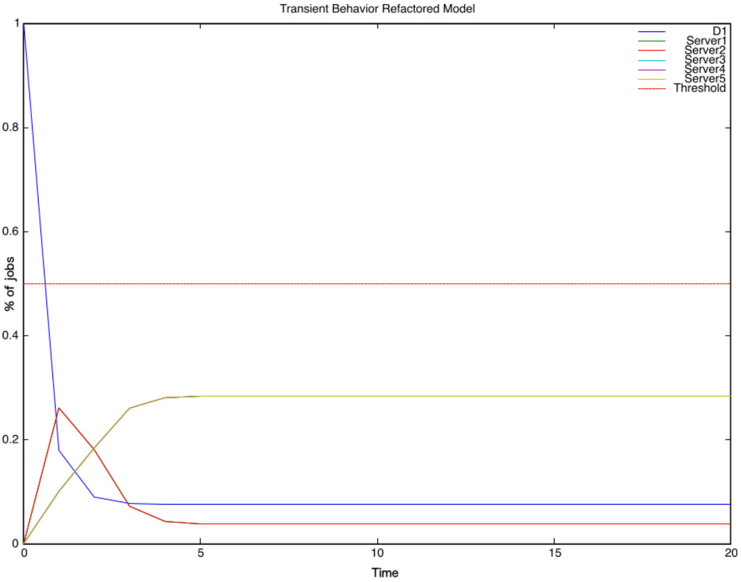
Illustrative Example: Analysis



Illustrative Example: Planning & Execution



Illustrative Example: Planning & Execution



Conclusion and future work

- We presented a proactive approach that provides self-adaptation capabilities to software systems in order to guarantee the fulfillment of performance requirements
- **Key Idea:** exploit the analysis of transient dynamics through QNs fluid approximation technique
- Our Research Agenda:
  - Formal specification of the constraints analysis and refactoring engine
  - Language definition for constraints and refactoring specifications
  - Symbolic modeling and optimization for the planning phase
  - Systematic comparison between our approach and other simulation techniques

Feedback and Thought provoking

- Feedback and Discussion:
  - What are the run-time variabilities in your domain of expertise?
  - How do you manage such variabilities?
  - What are the most critical performance/quality/cost requirements in your domain of expertise?
  - How do you evaluate the fulfillment of such requirements?
- Thought provoking statement:
  - Is it always convenient to refactor software systems?!
  - What if run-time variability is too fast?!
  - How to plan refactorings that are *"fast enough"* to cope with run-time variability?!