

Building Dynamic Models of Service Compositions With Simulation of Provision Resources

Dragan Ivanović¹, Martin Treiber²,
Manuel Carro¹, Schahram Dustdar²

¹*Universidad Politécnica de Madrid*

²*Technische Universität Wien*

*Work presented at the 29th International Conference on
Conceptual Modeling, ER-2010, Vancouver, CA, 13 Nov 2010*

3rd S-Cube Review, Essen 4-5 May 2010

Work within the realm of workpackage:

JRA-2.2: Adaptable Coordinated Service Compositions

and more specifically in deliverable CD-JRA-2.2.5: *Derivation of QoS and SLA specifications*: it derives a continuous-time model specification of orchestrations and uses it in a simulation.

Also related to workpackage

JRA-1.3: End-to-End Quality Provision and SLA Conformance

and is described in deliverable CD-JRA-1.3.5: *Integrated set of concepts for specifying end-to-end quality and negotiating SLAs*.

- **Service Orchestrations** describe centralized control flow combining services in order to achieve more complex tasks.
 - Exposed as services to the outside world — **composability**.
 - Cross organizational boundaries — **loose coupling**.
 - Usually designed around the notion of **business processes**.
 - Events, **asynchronous** exchanges, **stateful** conversations.
 - Often expressed in a **specialized language**: BPEL, YAWL, etc.
- **Service-Level Agreements (SLAs)**: constraints on values of **QoS attributes** that **provider** and **client** are expected to comply with.
- From the **client perspective**, attributing perceived QoS to **logic**, **components**, and/or **infrastructure resources** is **difficult**.
- **Providers** often implement some type of **elasticity of provision resources** to ensure SLA and meet demand in **different scenarios for load/request rates**.

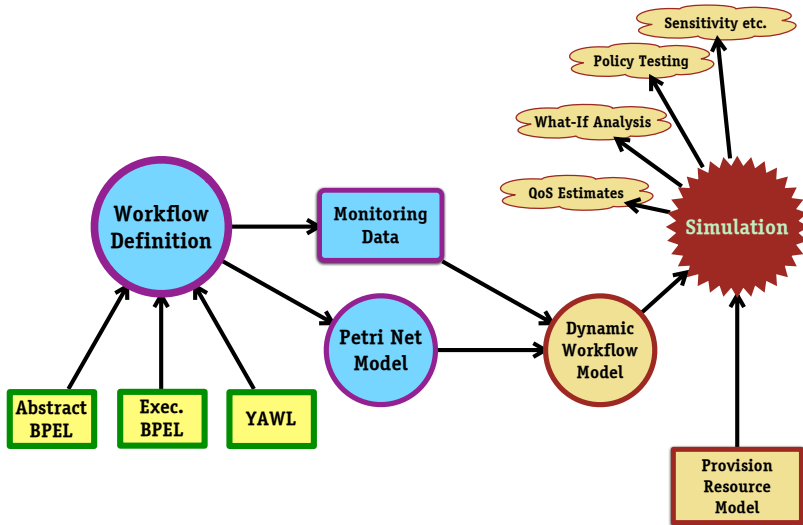
- Actual QoS offered/provided depends on both:
 - logic of the orchestration provided, and
 - load/capacity of the provision resources.
- Determining reasonable SLA offering: providers need to study expected behavior of provision system under different input scenarios, and select appropriate resource scaling policies.

Goal of this work

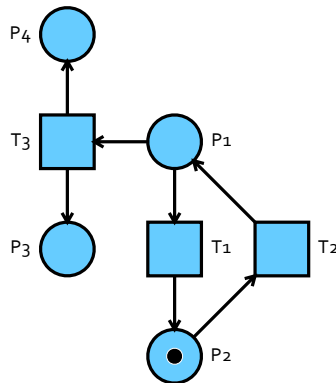
Develop dynamic (time-dependent) simulation models of service provision with QoS attributes, taking into account characteristics of the particular composition being served.

- Step 1: Develop model specifying dynamic behavior of orchestration.
- Step 2: Combine with resource model into simulation model.
- Step 3: Simulate behavior under different scenarios.

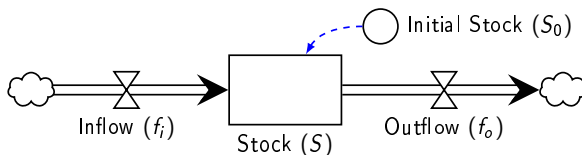
Overview of the Approach



- We use **Petri Nets** — well understood and widely used in workflow modeling.
 - Intermediate representation, can be used with several front-end languages.
 - We impose some constraints w.r.t. the shape of the net.
- **Places** (circles) stand for conditions in workflow execution (including *start* and *finish*).
- **Tokens** (dots) denote running instances and **mark** places.
- **Transitions** (squares) stand for workflow activities and fire when all input places (preconditions) are marked (**stochastic choice**).
- A **step** through a Petri net fires all transitions that can fire and moves tokens to new places.



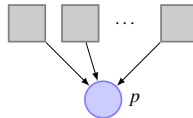
- The basic PT-nets **oversimplify** behavior because transition times are not taken into account (only discrete steps).
- In reality, transitions (activities) **take definite time**, while tokens (orchestration instances) **do not stay in places** for significant periods of time.
- To recover that dynamics, we use **continuous-time models**.



$$S|_{t=0} = S_0, \quad \frac{d}{dt}S = f_i - f_o \Leftrightarrow S = S_0 + \int_0^t (f_i - f_o) dt$$

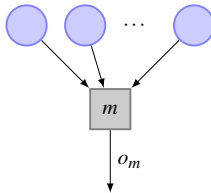
- Transitions \mapsto **stocks** [number of instances of the activity]
- Places \mapsto **flows** [number of instances transiting per second]

- CT scheme for a place:
non-input places aggregate
outputs from one or more
activities.



$$p(t) = \sum_{m \in \bullet p} \{o_m(t)\}$$

- A bit more
complex for
transitions with
several input
places.



$m_p(t)$: a new stock $\forall p \in \bullet m$

$$q_t = \operatorname{argmin}_{p \in \bullet m} \{m_p(t)\}$$

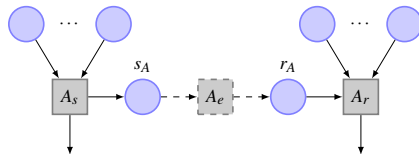
$$m(t) = m_{q_t}(t)$$

$$\frac{d}{dt} m_p(t) = p(t)w_{pm} - o_m(t), \forall p \in \bullet m$$

$$o_m(t) = \begin{cases} m(t)/T_m & T_m > 0 \\ q_t(t)w_{q,m} & T_m = 0 \end{cases}$$

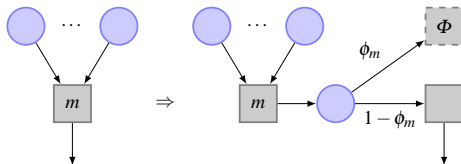
- The model derivation process can be **fully automated**.

- Modeling asynchronous message exchange between services is easy.

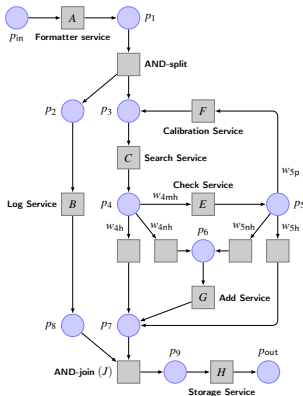
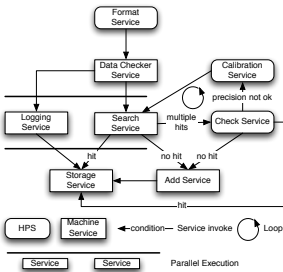


- We “open” the transition boxes that represent component or partner services and connect their CT-models.

- Accounting for failure: introduce failure probabilities, and a common failure sink.



Example: Data Cleansing Service



$$\frac{d}{dt}A(t) = p_{in}(t) - p_1(t)$$

$$p_1(t) = A(t)/T_A$$

$$p_2(t) = p_1(t)$$

$$\frac{d}{dt}B(t) = p_2(t) - p_8(t)$$

$$p_8(t) = B(t)/T_B$$

$$p_3(t) = p_1(t) + o_F(t)$$

$$\frac{d}{dt}C(t) = p_3(t) - p_4(t)$$

$$p_4(t) = C(t)/T_C$$

$$\frac{d}{dt}E(t) = p_4(t)w_{4nh} - p_5(t)$$

$$p_5(t) = E(t)/T_E$$

$$\frac{d}{dt}F(t) = p_5(t)w_{5p} - o_F(t)$$

$$o_F(t) = F(t)/T_F$$

$$p_6(t) = p_4(t)w_{4nh} + p_5(t)w_{5nh}$$

$$\frac{d}{dt}G(t) = p_6(t) - o_G(t)$$

$$o_G(t) = G(t)/T_G$$

$$p_7(t) = p_4(t)w_{4h} + o_G(t) + p_5(t)w_{5h}$$

$$\frac{d}{dt}J_{p_8}(t) = p_8(t) - p_9(t)$$

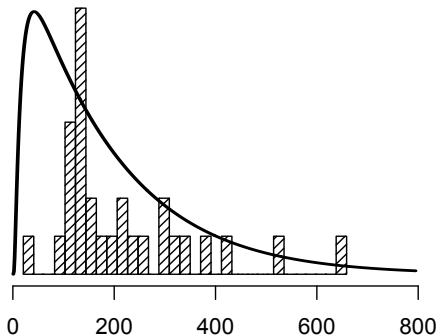
$$\frac{d}{dt}J_{p_7}(t) = p_7(t) - p_9(t)$$

$$p_9(t) = \begin{cases} p_8(t) & J_{p_8}(t) \leq J_{p_7}(t) \\ p_7(t) & J_{p_7}(t) < J_{p_8}(t) \end{cases}$$

$$\frac{d}{dt}H(t) = p_9(t) - p_{out}(t)$$

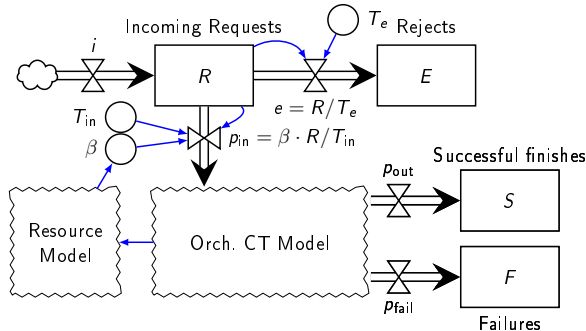
$$p_{out}(t) = H(t)/T_H$$

- CT models can be used to produce distribution of expected running times by feeding it with single unit spike (a Dirac pulse).
- Assessed by comparing discrete simulations with predictions by CT models in a range of simple workflow patterns.
 - Good fitting between median times.
- Data cleansing service orchestration simulated and compared with CT model prediction: good correspondence between model output and empiric data.
- **Calibration** of the model **with empirical data** crucial.

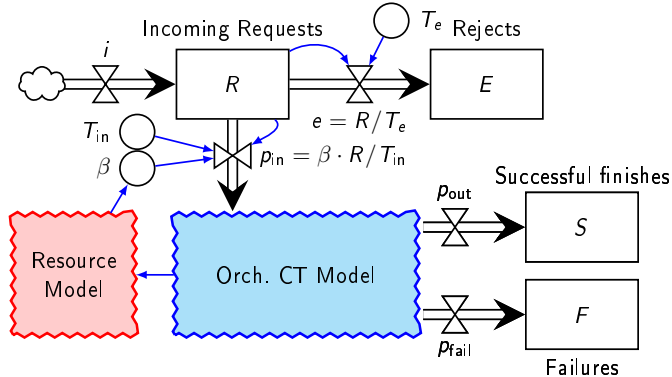


Framework Simulation Model for Several Orchestration SCUBE

- How to deal with several orchestrations which interact through resource usage?
- Plug CT model(s) of orchestration(s) into simulation framework.

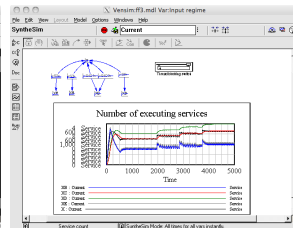
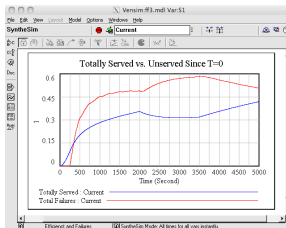
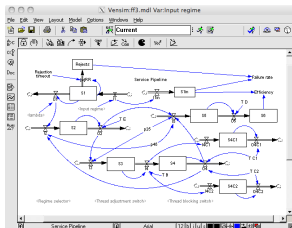


- **Resource model** aggregates values from orchestration CT model to **model provision infrastructure capacity and load**.
- **Blocking factor** (β) quenches request inflow when capacities exceeded.



- Basic accounting for rejects (E), successes (S) and failures (F).
- Provides for **separation of concerns** – separated development of:
 - **Orchestration model** (resource independent).
 - **Resource model** (orchestration independent).

- Simulation model fed to existing standard dynamic simulation tool:



- Simulating **concurrently served orchestrations** under **different input regimes**.
- Running time and reliability QoS “for free”.
 - Others (like cost) easily derivable.
- Typical simulations: **what-if**, **goal-seeking**, **sensitivity**.
 - Currently working on custom simulation tool better tailored for services.

- Automatically derived dynamic models of service orchestrations with provision resources can be used as handy tool for testing scenarios and assessing expected QoS and infrastructure management policies.
- Working on **integration with tools** for automatic derivation of PT-net models from executable orchestration specs.
- Aiming at modeling **elasticity in cloud infrastructures**.
- Challenge: cover other, more advanced and **data-dependent service composition descriptions e.g.:**
 - Concrete abstract and executable process specifications (e.g., BPEL),
 - Colored Petri-nets,
 - Other formalisms. . .

Building Dynamic Models of Service Compositions With Simulation of Provision Resources

Dragan Ivanović¹, Martin Treiber²,
Manuel Carro¹, Schahram Dustdar²

¹*Universidad Politécnica de Madrid*

²*Technische Universität Wien*

*Work presented at the 29th International Conference on
Conceptual Modeling, ER-2010, Vancouver, CA, 13 Nov 2010*

3rd S-Cube Review, Essen 4-5 May 2010