Modelling Variability, Evolvability, and Adaptability in Service Computing

(a vision for future research)

M.H. ter Beek
ISTI–CNR, Pisa, Italy

joint work with

A. Fantechi
University of Florence

S. Gnesi
ISTI–CNR

G. Zavattaro
University of Bologna

ACoTA @ ASE 2010

Antwerp, Belgium
20 September 2010

Outline

A vision for future research

Emerging topic in software engineering

Synergy between Software Product Line Engineering (SPLE) and Service-Oriented Computing (SOC)

Aim & foresight

Rigorous modelling techniques and analysis tools for the systematic, large-scale provision and market segmentation of *software services*

Design techniques for *software service line organizations* to develop novel classes of applications well adaptable to customer requirements as well as to changes in the context in which, and while, they execute

Outcome

Techniques and support tools to assist organizations to plan, optimize, and control the quality of software service provision at design-/run-time

A vision for future research

Emerging topic in software engineering

Synergy between Software Product Line Engineering (SPLE) and Service-Oriented Computing (SOC)

Aim & foresight

Rigorous modelling techniques and analysis tools for the systematic, large-scale provision and market segmentation of *software services*

Design techniques for *software service line organizations* to develop novel classes of applications well adaptable to customer requirements as well as to changes in the context in which, and while, they execute

Outcome

Techniques and support tools to assist organizations to plan, optimize, and control the quality of software service provision at design-/run-time

A vision for future research

Emerging topic in software engineering

Synergy between Software Product Line Engineering (SPLE) and Service-Oriented Computing (SOC)

Aim & foresight

Rigorous modelling techniques and analysis tools for the systematic, large-scale provision and market segmentation of *software services*

Design techniques for *software service line organizations* to develop novel classes of applications well adaptable to customer requirements as well as to changes in the context in which, and while, they execute

Outcome

Techniques and support tools to assist organizations to plan, optimize, and control the quality of software service provision at design-/run-time

Concrete proposal

First focus on the definition of the formal modelling framework:

1. Extend (semi)formal *existing* notations and languages for SOC with notions of variability to achieve increased levels of flexibility and adaptability in software service provision
2. Define *rigorous* semantics of variability over behavioural models of services to support design- and run-time analysis techniques
3. Develop formal analysis and verification techniques and *tools* that remain effective over specifications with variability points, including situations with variability triggered at run-time

Superposing variability mechanisms on current languages for service design, based on policies and strategies defined by service providers, makes it possible to identify variability points that can be triggered at run-time to increase adaptability and optimize the (re)use of resources

Beyond the state-of-the-art

SPLE/SOC software development approaches share a common goal:

Encourage an organization to reuse existing assets and capabilities rather than repeatedly redevelop them for new software systems

We will go beyond the state-of-the-art of the SPLE/SOC connection:

1. by providing a full formal model of existing connections, with verification techniques based upon them
2. by addressing run-time adaptability by extending the scope of the flexibility that can be achieved through the introduction of the notion of variability in service definitions

Overall strategy

Primary objective

Add variability and adaptability to the SOC principles by inheriting from SPLE mechanisms to include variability notions in a software artifact

Crosscutting concern

Guarantee basic *correctness* assumptions of the provided services, in terms of certain desired qualitative and quantitative properties, by means of formal modelling of variability and adaptability

Rationale

As current service-oriented languages do not support (or in a limited form) the possibility to *configure*, *adapt*, and *reconfigure* a system, they must be suitably extended to deal with variability and adaptability (likewise the formal verification techniques already available for them)

Overall strategy

Primary objective

Add variability and adaptability to the SOC principles by inheriting from SPLE mechanisms to include variability notions in a software artifact

Crosscutting concern

Guarantee basic *correctness* assumptions of the provided services, in terms of certain desired qualitative and quantitative properties, by means of formal modelling of variability and adaptability

Rationale

As current service-oriented languages do not support (or in a limited form) the possibility to *configure*, *adapt*, and *reconfigure* a system, they must be suitably extended to deal with variability and adaptability (likewise the formal verification techniques already available for them)

Overall strategy

Primary objective

Add variability and adaptability to the SOC principles by inheriting from SPLE mechanisms to include variability notions in a software artifact

Crosscutting concern

Guarantee basic *correctness* assumptions of the provided services, in terms of certain desired qualitative and quantitative properties, by means of formal modelling of variability and adaptability

Rationale

As current service-oriented languages do not support (or in a limited form) the possibility to *configure*, *adapt*, and *reconfigure* a system, they must be suitably extended to deal with variability and adaptability (likewise the formal verification techniques already available for them)

Modelling variability and evolvability

Provide a formal model of service lines that is able to capture the notion of variability, i.e. able to express the variations that certain characteristics of services can be subject to

Ongoing research [VaMoS09,VaMoS10,iFM10]

MTS: model behaviour of product family in one single model, namely an LTS with *may* and *must* transitions

Deontic logic DHML: natural way to formalize notions like *violation*, *obligation*, *permission*, and *prohibition*

Initial goal: obtain a common reference model that develops these concepts to take into account the peculiarities of SOC and SPLE

Only when the modelling of variability in SOC is consolidated, the reference model will be extended to address run-time adaptability

Extending service-oriented languages

Study extensions/modifications of current service-oriented languages to support variability: need to revisit *choreography*, *orchestration*, and *behavioural contract* languages

Ongoing research [TGC10]

JoRBA: extend service orchestration language **Jolie** with *rule-based adaptation rules*

Idea is to include *evolution hooks*: information on part of the system on which modifications could be applied, allowing the programmer to specify points that could be affected by future system reconfigurations

The dynamic variability logic should be developed separately, e.g. as a set of *evolution rules* which could be created/changed after the application has been deployed without affecting the running application

At run-time, *evolution managers/servers* should check the environment and user needs, control whether modifications must be applied to the application, and exploit its evolution hooks to reconfigure it

Develop verification techniques/tools

Combine/extend ideas underlying SOC verification techniques/tools with those of SPLE to support *design-time verification* and *validation*, *run-time monitoring*, and *verification* of *flexible and adaptable* services

Ongoing research [FASE08,SCP10,TCS07,VaMoS09,VaMoS10,iFM10]

CMC/UMC: adapt existing model checker for COWS/UML to deal with deontic/stochastic logics over L^2TS/M^2TS

Requires particular care, since the resulting analysis and verification techniques should still be effective over specifications with variability points, including situations with variability triggered at run-time

First concern: analysis of properties at family *and* at product level

Second concern: prove correct derivations of products *from* a family

Third concern: introduction of run-time adaptability, a big challenge for off-line verification by model checking requiring innovative techniques