

Software Engineering for Self-Adaptive Systems

by the RE Group

Betty H.C. Cheng, Anthony Finkelstein Jeff Kramer,
Jeff Magee, Sooyong Park, Schahram Dustdar
Jon Whittle, Nelly Bencomo (coordinators)

Once upon a time in Dagstuhl...



Software Engineering for Self-Adaptive Systems

from the perspective of Requirement Engineering (RE)



Self-adaptive systems

- A self-adaptive system **is able to modify its behaviour according to changes in its environment**
- As such, a self-adaptive system **must continuously monitor changes in its context and react accordingly**

Questions

- what aspects of the environment should the self-adaptive system monitor?
 - *clearly, the system cannot monitor everything*
 - *... so, what aspects of the environment are relevant?*
- ... and exactly what should the system do if it detects a less than optimal pattern in the environment?
 - presumably, the system still needs to maintain a set of high-level goals
 - .. but, non critical goals could well be relaxed, thus allowing the system a degree of flexibility

Even more questions

- **evolution**
 - made us think that requirements may change as the system evolves (adapting)
- .. but...which requirements are allowed to vary or evolve at runtime and which must always be maintained?
- .. *we were sure about something* 😊
 - *about uncertainty*

Uncertainty

- RE for self-adaptive systems must deal with (degrees of) uncertainty
- or may necessarily be specified as incomplete

The requirements specification should cope with:

- the incomplete information about the environment
- ... and the resulting incomplete information about the respective behaviour
- the evolution of the requirements at runtime

State-of-the-art

- people are working hard!
(see some references in report)
 - specification and verification of adaptive software
 - run-time monitoring of requirements conformance
 - goal models as a foundation for specifying the autonomic behaviour and requirements of adaptive systems
 - and others

Research Challenges

short-term and long-term (more visionary ideas)

- A new requirements language
- Mapping to architecture
- Managing uncertainty
- Requirements reflection
- Online goal refinement
- Traceability from requirements to implementation

A new requirements language

From shall to maybe, sometimes...

- Current languages for RE do not explicitly support uncertainty and adaptivity
- Traditional RE:
“the system shall do this”
- Adaptive RE:
“the system might do this”
“but it may do this...” ... as long as it does this”
“the system ought to do this... .” but, “if it cannot, it shall eventually do this ...”
- definition of terms and their relations (?)

Mapping to architecture

RE specification



large gap

(semi) automation

Architecture

Support for implementing reconfigurability
(component-based, AOP, SPL, etc technologies)

Managing uncertainty

- how much uncertainty we will manage:

a system cannot start out as a transport robot and self-adapt into a robot chef!

- certain requirements will not change (invariants)
- and others should permit a degree of flexibility

Requirements reflection

- self-adaptation deals with requirements that vary at runtime
- **reflection** enables a system to observe its own structure and behaviour
- requirements reflection would enable systems to be aware of their own requirements at runtime
- a model of the requirements to be available @runtime
- could a system dynamically observe its requirements?
- can we make requirements runtime objects?

Online goal refinement

- to automate and run on-line what we are doing currently off-line (RE/Design)

Traceability from requirements to implementation

- a constant challenge in all the topics shown above is *dynamic traceability*
 - operators of a new RE specification language should be easily traceable down to architecture, design, and beyond
 - if the RE process is performed at runtime we need to assure that the final implementation or behaviour of the system matches the requirements
- different from the traditional requirements traceability

(Some questions to answer)

We will have fun!

- How can graphical models, formal specifications, policies, etc. be used as the basis for the evolutionary process of adaptive systems versus source code?
- How can we maintain traceability among relevant artifacts including the code?
- How can we maintain assurance constraints during and after adaptation?
- How much should a system be allowed to adapt and still maintain traceability to the original system?
-

Final remarks

- challenges span RE activities during the development phases and runtime
 - monitor adherence and traceability to the requirements during runtime
 - acknowledge and support the evolution of requirements at runtime
 - software artifacts must be more abstract