Session 2 Discussion
Recorded by Lionel Montrieux, NII, Japan

Bashar Nuseibeh: Adaptation and Boundaries

Bashar starts by warning us that most of his talk is speculation, no results, though he has some evidence to support his claims. He hopes to trigger discussions.

Bashar focuses on boundaries, especially in ubiquitous and mobile systems. These systems should be able to reconfigure themselves, and adapt to changing contexts, but somehow users should not notice. Bashar argues that this seamlessness in fact hides boundaries between systems and components. These boundaries change, and information flows across them.

Boundaries are prevalent in software engineering, and it is very instructive to think of boundaries and the flow information across them. Bashar argues that boundaries in SE (e.g. boundaries between design and run time) are not disappearing, but are often tacit. It is important to identify those tacit boundaries.

In requirements engineering, he echoes Michael Jackson's argument that the essence of requirements engineering is to find the problem's boundaries.

Security is all about boundaries. Get them wrong, and your system is not secure. Trust assumptions are the raw material of boundaries, as they affect what designers believe to be true at a certain point. These trust assumptions change over time, and so do the boundaries that follow from them.

Bashar concludes by talking about topological boundaries, and the use of topology to describe the structure of the operational environment of a system, including its physical, digital, and/or social aspects.

John Mylopoulos: Engineering Adaptive Software Systems

John had prepared 10 questions, on the tools, techniques and concepts for building adaptive software systems (ADSSs). Due to time constraints, he had to skip a few of them.

What makes ADSSs special, is their architecture that separates the base system from the adaptation system.

1/ What requirements lead to a MAPE?

4 types of requirements need a MAPE look: awareness, evolution, adaptation and contextual requirements. Tools and techniques are required to deal with each of those types of requirements.

2/ What do runtime requirements look like?

The literature includes hierarchies of goals, fuzzy runtime requirement models, or runtime requirement models for reflection. The critical issue is that monitoring becomes non-scalable and/or intractable in the modelling language is over-expressive. The description of the system's behaviour, in particular, is very important.

3/ What failures trigger adaptation?

[skipped]

4/ Diagnosing the problem

Failure is often a symptom of a problem, not the problem itself. Root cause analysis is necessary to understand what happened, and hopefully prevent it from happening again. Some answers may exist in the AI diagnostic community. Solutions are often difficult to scale, but some seem to work well in practice.

5/ Systems with large adaptation spaces

How big an adaptation space does a system support? What's the space of all possible adaptations?

6/ Dealing with multiple failures

An adaptation that deals with one problem may interfere with adaptations that deal with other problems. Some work in this area includes the use of rules, or quantitative reasoning.

Rules tend not to scale well.

It is necessary to give one coherent answer to several problems, rather than several answers, each solving a single problem, but not necessarily compatible with each other.

7/ When can you reconfigure the system?

[skipped]

8/ Optimisations for adaptation requirements

In general, there may be many adaptations satisfying all the adaptive system's constraints. How to choose the best one is an optimisation problem, which requires both SAT-based and optimisation-based reasoning. This is difficult.

9/ The identification problem for ADSSs

Finding the relationship between input and output is a well-known problem in control theory. In our case, we look at requirements and failures.

Solutions in the literature include:
+ guestimate qualitative differential relations;
+ case-based reasoning;

+ learn over time.

10/ Full adaptation

See Jeff Kramer's rather excellent SEAMS'15 keynote.
Can we adapt to failures in ways unanticipated at design time?
Some have proposed the use of AI planning and domain knowledge to propose new requirements at runtime.