Session 2 Discussion
Recorded by Martina Maggio, Lund University, Sweden

Bashar Nuseibeh — Adaptation and Boundaries

Ubiquitous and mobile systems: depends a lot on the context in which they are placed and used. Seamless: it hides the boundaries between two things, it makes information flow across boundaries. Mention of security and privacy of systems and information flow.

Sometimes boundaries are a feature, sometimes they are a constraint. Considering these boundaries explicitly may be useful. Difficulty is that sometimes boundaries are unclear and changing. So we need to think about them both at design time and at runtime. 20 years ago the boundary between software and hardware was very clear. Now it is not so clear anymore and some things are implemented in hardware and software, example given is smart cities.

In the PhD spent time trying to understand how each person can specify partial knowledge about these boundaries. Every person has different perspective, the interesting part is how one perspective relates to the others.

— Boundary critique (Ulrich 2002, following Churchman 1970)
— The disappearing boundary between development time and runtime (Baresi Ghezzi 2010)
— Interaction design: making and using (Nakakoji 2011)

The boundaries are still there: tacit or explicitly. Security is also all about boundaries. Security and safety: a bird getting in an airplane engine makes it explode. Someone is throwing the bird. The first is a safety problem, the second is both a safety and a security problem. The engine should be bird resistant (safety) but also nobody should enter the perimeter and be able to throw the bird (security). Trust assumptions are important.

Interesting thing is not disappearing boundaries but changing boundaries. Boundaries between mobile devices, infrastructure and people are difficult to identify and manage. Trust assumptions that we were using to bound security problems no longer hold. Adaptive security challenge is to try to understand and cope with those changes.

Notion of topology of context, structure of the context in terms of the operational environment. Not necessarily physical, but also for example due to the network and the connections. You have a number of layers of this topology. Maps that are partly physical and partly virtual (Pasquale, SEAMS 2012, Tsigkanos RE 2014 http://lili-pasquale.lero.ie/papers/RE2014.pdf and ICSE 2015 http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=7203054). How can someone cross a digital boundary? Interplay between physical and digital gives better analysis.

John Mylopoulos — Engineering Adaptive Software System: a research agenda

Concepts, tools and techniques for building an artifact - in this case an adaptive software system.

What is special about adaptive software system and why cannot we use classical SE techniques? They are special in the sense that they have a distinctive architecture that separates concern between the base system and the adaptation mechanism. This adaptation functions (monitoring, analyzing, planning and execution) are separated from the main components.

Some class of requirements may lead to feedback loops.
Where did this MAPE function comes from and what are the requirements about those four components? What

are the additional requirements that lead to the feedback loop?

— Awareness requirements (Souza 2011)
This requirement R will not fail more than three times per month.
— Evolution requirements (Souza 2012)
If R fails more than three times per month, change it to R-.
— Adaptation requirements (Angelopoulos 2014)
Adaptation should not affect non-failing requirements.
— Contextual requirements (Ali 2010)
R is a requirement only in context C.

We need to represent runtime requirements and their state and other information about them. You have to take design time models and you have to extend them. At runtime, you have to create instances of these models, you have to keep history (for example to deal with awareness requirements and the “it will not fail more than a certain number of times”) to deal with these requirements (Dalpiaz 2013). Fuzzy runtime requirements models (Baresi 2010). Runtime requirements model for reflection (Bencomo 2010).

We need to be able to find the root causes of failures, for example using AI diagnostic techniques (Wang 2007).

System with large adaptation spaces. How large an adaptation space can we support. We are interested in making it as big as possible. A lot of the literature uses a requirement model to define an adaptation space (Souza 2011). The alternative is looking at the adaptation space as the architecture and see in how many possible ways you can change the architecture.

Lots of feedback loops in the system (each one of them introduced by a requirement like an awareness requirement or similar). Adaptation can cancel out each other. Somehow we must be able to deal with concurrent failures and deal with aggregated answers. In Rainbow (Garlan 2006), you can have rules where you can have “if R1 and R2 are both failing do A”. Enumeration of rules (they overwrite other rules). Another way to deal with that is qualitative reasoning (Angelopoulos 2014) — “if an adaptation A for problem P interferes with another problem P' don’t use it).

Optimization requirements — “minimize meeting cost while meeting some requirements”. Combination of a minimization requirement and the constraint that we have to restore another requirement (failing for example because the number of meeting rooms is not enough). The check of requirements consistency is usually done with a SAT solver but it is difficult to combine this with the minimization one. Use latest results from the Automated Reasoning community, maybe Sebastiani 2015.

Full adaptation: can we allow requirements to evolve due to failures in ways unanticipated at design time? A very preliminary answer proposes to use AI planning and domain knowledge to propose new requirements at runtime (Sabatucci, SEAMS 2015).?