Foreword: First Workshop requirements@run.time

Pete Sawyer, Nelly Bencomo, Jon Whittle
Computing Department
Lancaster University, UK
[sawyer, nelly, whittle]@comp.lancs.ac.uk

Daniel M. Berry
Computer Science Department,
University of Waterloo, Canada
dberry@uwaterloo.ca

Anthony Finkelstein
Department of Computer Science
University College London, UK
a.finkelstein@cs.ucl.ac.uk

***Abstract* — The first edition of the Workshop requirements@run.time was held at the Eighteenth International Conference on Requirements Engineering (RE 2010) in the city of Sydney, NSW, Australia on the 28th of September 2010. It was organized by Pete Sawyer, Jon Whittle, Nelly Bencomo, Daniel Berry, and Anthony Finkelstein. This foreword presents a digest of the presentations and discussions that took place during the workshop.**

Keywords- Requirements, reflection, run-time, self-adaptive systems

I. INTRODUCTION

Requirements@run.time was conceived to explore the issues related to software systems that are aware of and reason about their requirements [1] as they run. It is anticipated that such capabilities would bring benefits, not only for the design of self-adaptive and autonomous systems [2, 4], but also for the development of self-explaining systems [7] and for the testing and debugging of such systems.

Eight papers were submitted. Six papers were accepted for publication in these proceedings, and five were selected for presentation. Every submitted paper was reviewed by at least three program committee members.

The workshop aims were to

1. integrate and combine research ideas from RE, monitoring [5, 6], computational reflection [1], model-driven engineering (including models@run.time [3]) and autonomic, self-* systems;
2. provide a state-of-the-research assessment to guide research in the area;
3. stimulate the creation of a network of researchers in the area, and
4. plan and promote further events on these topics.

II. SESSION SUMMARIES

In the opening presentation, Sawyer outlined the format of the workshop. He then summarized the workshop aims. Following Sawyer's scene-setting, the paper presentation sessions followed. Paper presentations were divided into the two paper sessions and adopted the REFSQ model of appointing discussants to act as icebreakers for the discussion sessions that followed each paper. The presented papers are summarized:

SESSION 1: goals@design-&run-time

1. "Goal-Oriented Requirements Modelling for Running Systems", presented by Yun, introduces a typology explaining how different types of systems, with varying levels of dynamism, cope with requirements changes at run time with varying amounts of dynamic adaptation. The four types of systems are static, reactive, adaptive, and collaborative.

2. "Towards a Continuous Requirements Engineering Framework for Self-Adaptive Systems", presented by Qureshi, proposes a goal- and user-oriented framework for building self-adaptive systems (SASs). This framework includes the possibility of continuous adaptive requirements engineering (CARE) at run time. The CARE for a system is to be specified in the Techne modeling language. A companion paper, "Continuous Adaptive Requirements Engineering: An Architecture for Self-Adaptive Service-Based Applications", describes an architecture for building CARE into a SAS, but it was not presented.

SESSION 2: Statically and Dynamically Generated Requirements Data

3. "Run-Time Monitoring of System Performance", presented by Hill, proposes using a softgoal interdependency graph and a simulation of the architecture of a system to monitor the run-time performance of the system.

4. "Adaptive Monitoring of Software Requirements", presented by Ramirez, presents Plato-RE, a computation-based approach to monitoring satisfaction at run time and to dynamically adapt the monitoring to minimize monitoring costs. That is, besides a system being dynamically adaptive, the part of the system that monitors the need to adapt is adaptive.

5. "Using Requirements Traceability Links At Runtime — A Position Paper", presented by Paech, suggests a way that trace information maintained during the development of a dynamically adaptive system can be used to assist in the system's own run-time RE.

Each session concluded with a wrap-up discussion that pulled together the themes to emerge. These discussions led to a list of issues that we felt merited more research.

For the bulk of the afternoon session, we split into two groups. Each group chose a subset of issues to emerge form the paper sessions that they wanted to discuss.

Group one chose the following issues:

- *What does requirements@run-time mean?* For requirements@run.time to work, we need more than just a representation of a behavioural specification. An adaptive system needs to be aware of the current state of all of its models and of its entities, goals, adaptations, tracing links, entities, and preferences.
- *What can change [in the requirements models]*? There is a fundamental difference to be drawn between conventional and adaptation requirements. The former deal with the conventional behavior of a system, while the latter deal with how a system must adapt. Each can change at runtime.
- *At what level of granularity must change be applied?* A distinction should be made between changes that affect the requirements model and should be applied globally, and changes that affect the configuration of the system without affecting the requirements model. There are types of adpatation that can be specified at design time and those that can be derived only at run time. In some systems, all adaptation scenarios can be predicted and specified at design time. At the other extreme, there are systems where new requirements can be inferred from the presence of undesirable conditions in the environment. In either case, it is essential that the way adaptations are specified be independent of the system on which an adaptation must be performed. For any system in which humans are in the loop and are able to inject new requirements at run-time, users need guidance on how to define new requirements and the constraints under which they can do so.
- *Why GORE (goal-oriented RE) for self-adaptive systems?* Goal models are good media for tracing from high-level goals to low-level requirements. Goals allow reasoning about whether an adaptation is effective, i.e., achieve or mainains the satisfaction of the goals. Softgoals and user preferences drive the selection of adaptation strategy. However, goal models can get to be too big to be manageable. On the other hand, is there a paradigm that works better?

Group two considered the issue:

- *The levels of granularity or abstraction at which run-time requirements models could be applied.* The group noted that adaptive or autonomic capabilities of a software platform probably serve to define the levels of abstraction that are useful. The group noted also that a range of emerging technologies appeared to offer new capabilities, and we need to be aware of these since they constrain what can be achieved with run-time requirements models.

The group made two recommendations,

- that for requirements@run-time to progress, we needr to be aware of related disciplines, particularly autonomic computing, AI and machine sensing, and bio-inspired computing, and
- that we should follow the lead of other areas of SE research and develop a set of reference case studies to permit different approaches to be evaluated.

Following reporting of the two groups' findings by their rapporteurs, Lilliana Pasquale and Tom Hill, the workshop segued into the final plenary session where a number of the groups' points generated lively discussion. In particular, we wrestled with what we meant by "new requirements emerging at run time". For example, if a system adapts to an unforeseen event, then is that a new requirement or is that just a requirement that was already present but merely implicit? Following this line of reasoning, we wondered whether requirements@run.time is really about making emergent requirements explicit, for tracing, post-hoc diagnosis, etc. The conclusion was that a fundamental problem to grapple with was whether our self-adaptive systems could really deal with events unforeseen at design time, or whether we could really only sensibly specify events that could be foreseen at design time. Each is hard, but the former appears to be the harder of the two.

III. FINAL REMARKS

All the workshop's aims were achieved to a greater or lesser extent and a large number of challenging and unanswered questions were identified. There was consensus that we should aim to repeat the workshop at RE 2011.

REFERENCES

[1] N. Bencomo, J. Whittle, P. Sawyer, A. Finkelstein, and E. Letier. Requirements reflection: requirements as runtime entities. In *ICSE 32*, 199–202, Cape Town, South Africa, 2010.

[2] D. M. Berry, B. H. C. Cheng, and J. Zhang. The four levels of requirements engineering for and in dynamic adaptive systems. In *11th International Workshop on Requirements Engineering: Foundation for Software Quality (REFSQ'05)*, Porto, Portugal, 2005.

[3] G. S. Blair, N. Bencomo, and R. B. France. Models@ run.time. *IEEE Computer*, 42(10):22–27, 2009.

[4] B. H. C. Cheng, H. Giese, P. Inverardi, J. Magee, and R. de Lemos. Software engineering for self-adaptive systems: A research road map. In *Software Engineering for Self-Adaptive Systems*. Springer, 2008. LNCS, 5525.

[5] S. Fickas and M. S. Feather. Requirements monitoring in dynamic environments. In *RE Conf.*, 1995.

[6] W. N. Robinson. A requirements monitoring framework for enterprise systems. *Requir. Eng.*, 11(1):17–41, 2006.

[7] J. Whittle, W. Simm, and M.-A. Ferrario. On the role of the user in monitoring the environment in self-adaptive systems: a position paper. In *Workshop on Software Engineering for Adaptive and Self-Managing Systems, SEAMS'2010*, 69–74, New York, NY, USA, 2010. ACM.