Issues of Adaptable Software for Open-World Requirements

Carlo Ghezzi
DEI Politecnico di Milano, DeepSE Group, Italy

Software is embedded in an increasingly dynamic world. In many practical cases – from pervasive computing to enterprise-level scenarios – the surrounding environment and the requirements change continuously and the software has to change accordingly. While change requests are traditionally handled off-line, there is an increasing need for managing them on-line, as the software is running and providing service, possibly without involving a designer in the loop.

Software is also rapidly changing in the way it is developed, deployed, operated, and maintained. Applications are often built by integrating components that are produced by independent organizations, or by personalizing and adapting generalized frameworks, rather than developing new solutions in-house and from scratch. Increasingly, applications are built by integrating and composing services that are owned and managed by separate organizations, which offer them as *services* on the clients.

All these reasons demand revisiting the traditional approaches followed to engineer applications. The main challenge is how to combine extreme dynamism and flexibility with the levels of dependability that are requested by the application. In particular, the clear-cut separation between *development time* – requirements, design, implementation, modeling, analysis and verification, deployment – and *run time*, upon which traditional software engineering approaches are founded, is blurring. At development time one must live with imprecision and partial knowledge of what will actually occur at run time, and the application must be adaptable to the changing situations that will arise at run time. Verification must extend from development time to run time.

Engineering software systems that operate in this new setting requires viewing and supporting the software lifecycle as a *feedback loop*. As an application is running, the environment in which it is embedded is monitored to capture its actual behavior. Updates in the requirements and the information extracted from run time data may be used to calibrate the models that were used when the application was initially developed. Calibrated models may then drive an adaptation of the application – via some automated model-driven process – or may be an input provided to humans in the loop who can perform the necessary changes.

The lectures will progress through the following issues:

- Introduction and motivations.
- Specifications and service level agreements among different stakeholders and subsystems. Functional and non-functional qualities.
- Architecture: how do the requirements for dynamic adaptation affect software composition. Language support to dynamic adaptation.
- Modelling and analysis: how can a development time approach be integrated with a run time approach that calibrates models via machine learning of environment parameters. We will focus on quantitative probabilistic models and on non-functional properties that can be assessed by model checking. We will show how can model parameters be calibrated at run time.

References

1. C. Ghezzi, L. Baresi, E. Di Nitto. *Towards Open-World Software.* Computer, Vol. 39, No. 10; IEEE; pp. 36-43; 2006.
2. B. H. C. Cheng, R. de Lemos, H. Giese, P. Inverardi, J. Magee. *Software Engineering for Self-Adaptive Systems.* [outcome of a Dagstuhl Seminar]; Springer; 2009.
3. D. Menasce. *QoS Issues in Web Services.* IEEE Internet Computing, Vol. 6, No. 6; 2002.
4. M. Kwiatkowska. *Quantitative Verification: Models, Techniques and Tools.* ESEC/FSE; 2007.
5. I. Epifani, C. Ghezzi, R. Mirandola, G. Tamburrelli. *Model Evolution by Run-Time Parameter Adaptation.* Procs. 31st International Conference on Software Engineering (ICSE'09); IEEE Computer Society Press; pp. 623-626; 2009.