Model-Driven Design and Organic Computing – Combinable Strategies? *

Peter Puschner† and Raimund Kirner
Vienna University of Technology, Austria
Email: peter@vmars.tuwien.ac.at

It is difficult to relate a design approach to a term that stands for a class of computing systems that exhibit certain behaviors, as they belong to different phenomenologic categories. We will, however, compare the assumptions that the two approaches are based on, and draw our conclusions about the situations in which each of the strategies is useful.

Model-driven design is an approach that aims at keeping the complexity of the design and implementation process of applications low, thus avoiding errors in application development. Model-driven design technologies use domain-specific modeling languages that provide means to describe the objects, object semantics, constraints, relations and behaviors that are relevant to the problems in the domain. Developers use these artifacts to build applications in the form of declarations that describe their design intent. In addition, model-driven design environments include software components to analyze models wrt. certain properties and to synthesize different representations (e.g., source code) and information about the model. The automatic generation of source code that is "correct by construction" from high-level models is considered one of the major advantages of model-driven design and engineering over a the traditional, error prone software development process that strongly relies on manual coding [1].

Model driven design is applicable to application areas that are well understood. The modeling mechanisms of a design environment are expected to comprise all aspects (objects, relations, behaviors) of the domain that are necessary to build working applications. Together with the evaluation components they are the basis for constructing systems that are trusted to behave according to their intent.

Organic Computing characterizes a research area that aims at building autonomous computing systems that dynamically adapt to changing environmental conditions. Organic computing systems use self-X properties (e.g., self-explaining, context awareness, self-organisation, self-healing) to assess environmental conditions and their own operational state and to re-organize themselves similarly as living organisms do. Using these strategies they are expected to perform their tasks even under unforseen situations.

In organic computing the assumption is that the knowledge about the application area is incomplete. Systems are expected to cope with situations that have not been modeled before. The hope is that these systems can deal with phenomena that are not so well understood or even unknown, or for which the analysis cost are considered as being too high.

Exclusive Domains or Combinable Approaches? Having described the characteristics of the application domains in which the discussed strategies are used we could conclude that each of the strategies is exclusive to a certain domain. On the other hand, we can immediately think of situations in which a combination of the two strategies looks promising:

Consider the case when a model-driven design fails to cover a critical situation that has been unknown so far. Could not organic computing provide a way to master this otherwise disastrous case?

When we look at autonomous organic computing systems we realize that it is difficult to predict whether they can handle certain situations or not. So could not we build models of self-X properties and behaviors and use them in model driven design, thus getting better control and more detailed insights about organic computing systems?

If we were able to answer these questions positively, we could indeed expect to construct more robust and adaptive systems. We must not expect miracles, however, because even the best modeling and evaluation tools have their limits when the state space grows, and even the most sophisticated self-X mechanisms cannot handle situations that are too far off the anticipated problem classes.

References

[1] D. C. Schmidt. Model-driven engineering. *IEEE Computer*, 39(2):25–31, Feb. 2006.

*Position statement for panel: "Model Driven Design and Organic Computing – Contradictory or Synergetic Approaches to Overcome the Embedded Software Crisis."

†This work is supported by the European Community's Seventh Framework Programme [FP7/2007-2013] under grant agreement no. 214373.