Managing Design-Time Uncertainty

Michalis Famelis
Université de Montréal
famelis@iro.umontreal.ca

Marsha Chechik
University of Toronto
chechik@cs.toronto.edu

Any software system is the accumulated result of many design decisions taken by its developers. During the course of development, however, developers are often uncertain about how to make these decisions. This uncertainty reflects lack of knowledge about the design of the system, rather than about the environment in which the system is intended to operate. It is therefore called *design-time uncertainty*, and is different from *environmental uncertainty* [1]. Addressing environmental uncertainty requires using strategies such as self-adaptation [2], which result in fully functional software systems, capable of operating under uncertain conditions, i.e., *uncertainty-aware software*. In contrast, design-time uncertainty (henceforth, also simply "uncertainty") cannot be "coded away". Rather, it must be tackled as part of the process of software development, i.e., using *uncertainty-aware software development methodologies*.

Existing methodologies, languages and tools assume that their inputs do not contain any uncertainty. Thus, uncertainty is rendered an undesirable characteristic that developers should either avoid or remove altogether before resuming their work. This results in either costly delays or potentially premature – and therefore risky – resolutions of uncertainty as developers make provisional decisions and attempt to keep track of them in case they need to be undone. We present an alternative strategy: the explicit management of design time uncertainty as part of the course of software development [3].

Specifically, we build on previously published work for encoding alternative design decisions in *partial models* which can subsequently be used for tasks such as reasoning, refinement and transformation [4]. These techniques had been implemented as *partial model operators* in MU-MMINT, an interactive modelling tool [5]. We combine these point solutions into a coherent, tool-supported methodology for tackling design-time uncertainty. This combination allows deferring the resolution of uncertainty for as long as necessary while the development work can continue.

To create a coherent methodology, we must address some fundamental questions, such as: *What does it mean to "tackle" uncertainty? How does it get added to software artifacts? When does it get removed?* To gain a more detailed understanding of how uncertainty is manifested and manipulated at different stages, as well as the different inputs, outputs, pre- and post-conditions of partial model operators, we developed the *"Design-Time Uncertainty Management"* (DETUM) model, shown in Fig. 1. The DETUM model consists of three stages: During the *Articulation* stage, uncertainty increases as developers encode alternative designs in partial models. During the *Deferral* stage, the goal is to avoid making premature decisions, while still being able to make use of software engineering techniques. To do that, developers use partial models as primary development artifacts in combination with techniques that are appropriately *lifted* so that they are applicable to partial models. During this stage *the degree of uncertainty in the model remains unchanged*. During the *Resolution* stage, the degree of uncertainty in the partial model is reduced to reflect newly acquired information. The ultimate result is a model without any uncertainty. The DETUM model allows us to to contextualize the various partial modelling operators and identify gaps through lessons learned from non-trivial scenarios.

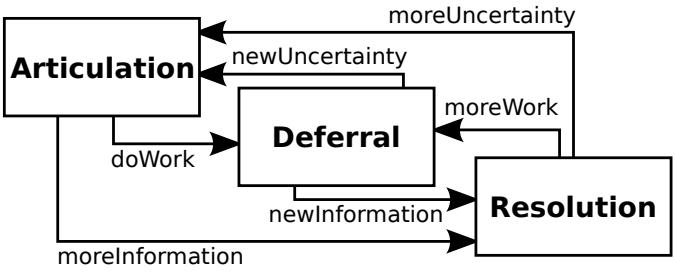

Fig. 1: The *Design-Time Uncertainty Management* (DETUM) model: an idealized timeline of uncertainty management.

This work makes the following contributions: (a) the DETUM model; (b) the mapping of various partial model operators to the DETUM model; (c) based on the above, a methodology for managing design-time uncertainty, and (d) a validation of the usability and effectiveness of the methodology based on two non-trivial uncertainty management scenarios.

REFERENCES

[1] A. Ramirez, A. Jensen, and B. Cheng, "A Taxonomy of Uncertainty for Dynamically Adaptive Systems," in *Proc. of SEAMS'12*, 2012, pp. 99–108.

[2] N. Esfahani and S. Malek, "Uncertainty in Self-Adaptive Software Systems." in *Software Engineering for Self-Adaptive Systems II*, ser. LNCS, R. de Lemos, H. Giese, H. Müller, and M. Shaw, Eds., 2012, vol. 7475, pp. 214–238.

[3] M. Famelis and M. Chechik, "Managing Design-Time Uncertainty," *Software & Systems Modeling*, March 2017. [Online]. Available: https://doi.org/10.1007/s10270-017-0594-9

[4] M. Famelis, "Managing Design-Time Uncertainty in Software Models," Ph.D. dissertation, University of Toronto, 2016.

[5] M. Famelis, N. Ben-David, A. Di Sandro, R. Salay, and M. Chechik, "MU-MMINT: an IDE for Model Uncertainty," in *Proc. of ICSE'15 Formal Demonstrations*, 2015, pp. 697–700.