Towards Generative Metaprogramming

Maider Azanza\textsuperscript{1}, Salvador Trujillo\textsuperscript{2}, Oscar Diaz\textsuperscript{1}
maider_azanza@ehu.es, strujillo@ikerlan.es, oscar.diaz@ehu.es

\textsuperscript{1} ONEKIN Research Group
University of the Basque Country
San Sebastian, Spain
\textsuperscript{2} IKERLAN Research Centre
Mondragon, Spain

Abstract. Metaprogramming is the concept that program synthesis is a computation. Generative programming is about developing metaprograms that synthesize other programs. Our intention goes one step beyond: we aim to generate metaprograms, which when executed, will generate a target program of a product line. This work describes the Generative metapro\textsuperscript{2}rogramming for Variable structure (GROVE) approach to synthesize metaprograms. A portlet product line case study is used to illustrate this approach.

Rationale. A main insight of this work is that recent advances in Software Engineering have reduced the cost of coding programs at the expense of increasing the complexity of program synthesis, i.e. the process of coming up with the final program. Model Driven Development (MDD) and Software Product Lines (SPL) are two cases in point. MDD conceives metaprograms as a pipeline of model transformations, where most popular transformations are top-down. Although MDD brings important benefits (e.g. reduced costs, improved quality, reduced development time), now the synthesis of the final program becomes more complicated. The traditional cycle of configuring-linking-compiling, now needs to be enriched with additional transformation steps that move the model down to the code. The synthesis metaprogram becomes more complex.

On the other hand, SPLs aim at building a set of related products out of a common set of core artifacts. Unlike MDD, now the stress is not so much on the abstraction level at which the software is specified, but on conceiving programs as pre-planned variations from common artifacts. The synthesis of a program starts with the SPL’s common artifacts where variation points are gradually instantiated to establish the features that will be eventually exhibited by the end program. However, the description of how a program is synthesized out of a set of artifacts is far from trivial. Again, program synthesis becomes more elaborated.

MDD and SPL are now well-established approaches for reuse. Both together will result in even more important gains. However, the description of the process to synthesize final programs out of models and features ends up in complex scripts. A combined use of these two techniques was reported in [5]. For this case, a typical script to realize a synthesis metaprogram consists of around 500 LOC of batch processes that use 300 LOC of ANT makefiles and 2 KLOC of Java...
code, taking around 4 person/day to complete. So far, these bulky metaprograms were based on laborious, hand-crafted scripting.

**The Idea.** Generative programming is about metaprograms that synthesize other programs. This work describes an approach to generate metaprograms, which when executed, will synthesize a target program of a product line. We name this process generative metaprogramming. Our intention is to accelerate the development of synthesis metaprograms by generating them from abstract specifications. Doing so, a synthesis metaprogram is declaratively specified rather than programmatically implemented (GROVE Tool Suite supports our generative approach to metaprogramming [4]).

In order to generate metaprograms from these abstract models we define a three step process. First, for a given domain, a template is specified defining the different abstraction layers and the transformations through which a program specification goes, until code artifacts are obtained. This template is valid for every program that belongs to the domain. Second, a certain SPL is defined in terms of instances of the created template (This work does not focus on the creation of the SPL capability. We consider that the SPL is created using existing approaches [1, 2]). Third, a specific program can be declaratively specified using the template instances. The point is that from this specification, the synthesis metaprogram code is generated.

This three steps process refers solely to the generation of product synthesis metaprograms. The automation of such generation is feasible because there are no people involved in metaprogram execution. This is not always the case, there are tasks where different people have to take part (e.g. core asset development). **Software Process Engineering Metamodel Specification (SPEM)** [3] is an specification that defines software and systems development processes and their components. Our next aim is to enrich our tools with the ability to generate SPEM processes that will guide users in such tasks.

**Conclusions.** This work described ideas to synthesize metaprograms, which when executed, will synthesize a target program of a product line. Specifically, we elaborated on the generation of metaprograms from abstract specifications, introducing our GROVE approach. Those interested in more details can visit [http://www.onekin.org/grove](http://www.onekin.org/grove).

**Acknowledgments.** Maider Azanza enjoys a doctoral grant from the Basque Government under the Researchers Training Program.

**References**