UML Profile Generation for Annotation-based Modeling*

Alexander Bergmayr[1], Michael Grossniklaus[2], Manuel Wimmer[1], and Gerti Kappel[1]

[1]Vienna University of Technology, Austria
[bergmayr|wimmer|kappel]@big.tuwien.ac.at
[2]University of Konstanz, Germany
michael.grossniklaus@uni-konstanz.de

Abstract: The capability of UML profiles to serve as annotation mechanism has been recognized in both industry and research. With *JUMP*, we have presented a fully automatic approach to generate profiles from annotation-based Java libraries. We have demonstrated the practical value of *JUMP* by contributing profiles that facilitate reverse-engineering and forward-engineering scenarios for the Java platform. Its evaluation shows that automatically generated profiles are equal or even improved in quality compared to profiles currently used in practice.

Since the introduction of the UML profile mechanism, numerous profiles have been developed, many of which are available by the OMG standardization body. Even in industry, their practical value has been recognized as today's modeling tools offer predefined profiles. They are considered as a major ingredient for model-based software engineering approaches by providing features supplementary to the standard UML metamodel. This powerful capability of profiles can also be exploited in terms of an annotation mechanism. As a result, such profiles leverage *annotation-based modeling*, where defined stereotypes show similar capabilities as annotations in programming languages such as Java.

Deriving stereotypes from available programming libraries to produce corresponding profiles at the modeling level seems desirable. They enable high-level platform-independent models (PIMs) to be refined into models specific to a platform (PSMs), where the platform refers to the library from which the profile was derived. Turning this forward engineering (FE) perspective into a reverse engineering (RE) one, existing programs can be represented as UML models that capture annotations by applying the corresponding profiles. Therefore, platform-specific profiles and their application are beneficial from both perspectives. In the RE step, model analyzers can exploit captured stereotypes to facilitate comprehension, whereas profiled UML models, i.e., models to which profiles are applied, pave the way for model transformers to generate richer program code in the FE step.

For that reason, we have presented *JUMP* [BGWK14b] that enables UML profiles to be generated automatically from Java libraries, which use annotations. We have discussed three significantly different representations of profiles in current modeling tools and highlighted the benefits of the mapping realized by *JUMP*. It allows annotations to be applied in

*This work is co-funded by the European Commission, grant no. 317859.

a controlled UML standard-compliant way as the generated stereotypes extend exactly the required UML metaclasses. From a language engineering perspective, stereotypes facilitate defining constraints and model operations because they can directly be used as explicit types similar to a metaclass in UML. *JUMP* realizes a mapping between Java's annotation language and UML's profile language. It enables the generation of specific stereotypes for corresponding annotations, which in turn leverage platform-specific profiles.

We have implemented tool support[3] for *JUMP* [BGWK14a] based on Eclipse. Its evaluation shows that automatically generated profiles are equal or even improved in quality, e.g., completeness and correctness, compared to profiles used in practice. Currently, we provide in total over 700 stereotypes comprised by 20 profiles that complement OMG's collection of standardized profiles with supplementary profiles for the Java platform.

To show the feasibility of *JUMP*, we have extensively applied it as enabling technology in the ARTIST project [BBC+13], where we work towards a cloud-oriented software modernization approach, which involves representing PSMs that refer to the platform of existing applications, e.g., the Java Persistence API (JPA), and the platform of "cloudified" applications, e.g., Objectify[4], when considering cloud datastores. For instance, JPA annotations facilitate distinguishing between plain associations and compositions and determining precise multiplicities. Moreover, annotations of Objectify enable method bodies to be generated even from a structural viewpoint and non-functional properties to be improved. These examples highlight the practical value of *JUMP* for RE and FE tools.

Ongoing work includes (i) the contribution of *JUMP* to the Eclipse-based UML Profile Repository (UPR)[5], (ii) the consideration of Java 8 features, such as repeating annotations, (iii) the generalization of generated profiles based on EMF Profiles [LWWC12] to allow their application to a wider range of modeling languages, and (iv) the extension of *JUMP's* scope to profiles that capture annotations independent of platforms, thereby shifting such annotations to a more conceptual level.

References

[BBC+13] Alexander Bergmayr, Hugo Bruneliere, Javier Cánovas, Jesús Gorroñogoitia, George Kousiouris, Dimosthenis Kyriazis, Philip Langer, Andreas Menychtas, Leire Orue-Echevarria, Clara Pezuela, and Manuel Wimmer. Migrating Legacy Software to the Cloud with ARTIST. In *CSMR*, pages 465–468, 2013.

[BGWK14a] Alexander Bergmayr, Michael Grossniklaus, Manuel Wimmer, and Gerti Kappel. Bridging Java Annotations and UML Profiles with JUMP. In *Demonstration @ MoDELS*, pages 1–5, 2014.

[BGWK14b] Alexander Bergmayr, Michael Grossniklaus, Manuel Wimmer, and Gerti Kappel. JUMP - From Java Annotations to UML Profiles. In *MoDELS*, pages 552–568, 2014.

[LWWC12] Philip Langer, Konrad Wieland, Manuel Wimmer, and Jordi Cabot. EMF Profiles: A Lightweight Extension Approach for EMF Models. *JOT*, 11(1):1–29, 2012.

[3] https://code.google.com/a/eclipselabs.org/p/uml-profile-store

[4] https://code.google.com/p/objectify-appengine

[5] https://projects.eclipse.org/projects/modeling.upr