Modell-basierte Programmgenerierung und Methoden des Übersetzerbaus — Zwei Seiten derselben Medaille?

Wolf Zimmermann

Institut für Informatik
Martin-Luther-Universität Halle-Wittenberg
Von-Seckendorff-Platz 1
06120 Halle (Saale), Germany
wolf.zimmermann@informatik.uni-halle.de

In der vergangenen Dekade hat Modell-basierte Entwicklung in der Softwaretechnik zunehmende Bedeutung gewonnen, vgl. z.B. [TS07, SRC+12, JLM+12]. Aus Modellen, die in einer formalen Sprache definiert sind (Domänen-spezifische Sprache) wird Code generiert, der die Modelle implementiert. Da Domänen-spezifische Sprachen nicht selten starken Änderungen und Erweiterungen unterworfen sind, haben sich Werkzeugkästen wie beispielsweise das Eclipse Modeling Framework (kurz: EMF) etabliert [SBMP08], mit deren Hilfe die Codegeneratoren selbst generiert werden können. Zur Spezifikation Domänen-spezifischer Sprachen wird ein Metamodell definiert, aus dem dann mittels Modelltransformationen die Transformationsregeln in die Zielsprache spezifiziert werden. Neben dem Codegenerator für eine Domänen-spezifische Sprache werden beispielsweise mit der EMF-Technologie Editoren für die Domänen-spezifische Sprache generiert, die auch durch die Eclipse-Technologie in Programmier- und Anwendungsumgebungen eingebettet werden können.

Als grundlegende Technologie werden Metamodelle meist in graphischer Form (z.B. UML-Klassendiagrammen) oder durch eine kontextfreie Grammatik [EEK+12] definiert. Mittels OCL können Konsistenzbedingungen angegeben werden, die alle Modelle einer Domänen-spezifischen Sprache erfüllen müssen. Meist führen diese zu Laufzeitprüfungen. Oft werden zusätzlich im Code des Modell-basierten Codegenerators noch manuelle Ergänzungen und Änderungen vorgenommen. Gründe können Effizienzsteigerungen in der Codegenerierung oder weitere Überprüfungen sein. Eine Weiterentwicklung einer Domänen-spezifischen Sprache erfordert daher nicht selten eine grundsätzliche Revision und Überarbeitung des Codegenerators.

Die Aufgabenstellung für Übersetzer ist nahezu identisch: ein Programm in höherer Programmiersprache wird in ein Programm einer Maschinensprache oder einer anderen Hochsprache (Cross-Compiler) transformiert. In einem Übersetzer wird der Quelltext auf korrekte Syntax hin analysiert und in eine interne Datenstruktur, den abstrakten Syntaxbaum transformiert. Anschließend werden Konsistenzbedingungen wie beispielsweise Typkorrektheit analysiert. Im Falle eines Cross-Compilers wird der abstrakte Syntaxbaum in das Zielprogramm transformiert, ansonsten in eine Zwischensprache, die weiter in den

Maschinencode übersetzt wird. Hier haben sich z.T. seit Mitte der 1970er Jahre Werkzeuge [Joh75, KHZ82, DUP+82] und Werkzeugkästen zur Generierung von Übersetzern etabliert. Aktuelle Werkzeugkästen sind beispielsweise ANTLRv3 [Par07] oder Eli [KWS07]. Die Syntaxanalyse wird durch Angabe der Lexik als reguläre Ausdrücke, durch eine kontextfreie Grammatik für die konkrete Syntax und der Datenstruktur für die abstrakte Syntax angegeben. Die Konsistenzprüfungen können aus geordneten Attributierten Grammatiken generiert werden, die Transformation in die Zielsprache bzw. Zwischensprache wird aus Baumtransformationen generiert. Im generierten Übersetzer müssen keine Änderungen mehr vorgenommen werden, da Hilfsfunktionen Bestandteil der Spezifikationen sind. Hilfsfunktionen sind bereits in der Wirtssprache (der Sprache, in der der Übersetzer implementiert ist) definiert. Sie können deshalb problemlos eingebunden und geändert werden. Dadurch ist es nicht notwendig, den generierten Code anzufassen, und man kann agil einen Übersetzer Sprachkonzept um Sprachkonzept anreichern.

Als Fazit ergibt sich eine deutliche Korrespondenz zwischen der Generierung Modellbasierter Codegeneratoren und der Generierung von Übersetzern (vgl. auch [Jör11]): Die Begriffe Metamodell und Abstrakte Syntax sind konzeptuell identisch. Konsistenzbedingungen werden durch unterschiedliche Technologien wie OCL bzw. attributierte Grammatiken definiert und Modelltransformationen entsprechen konzeptuell Baumtransformationen. Für die praktische Entwicklung von Modell-basierten Codegeneratoren kann daher ohne Weiteres Übersetzerbautechnologie verwendet werden. Ein möglicher Vorteil durch die Verwendung von attributierten Grammatiken an Stelle von OCL wäre eine statische Prüfung der Konsistenz der Modelle anstatt dies auf Laufzeitprüfungen zu verlagern. Durch die Definition von Hilfsfunktionen in der Wirtssprache anstelle derer nachträglichen manuellen Integration in den Codegenerator wäre eine agilere und kostengünstigere Entwicklung von Modell-basierten Codegeneratoren möglich. Die für die industrielle Praxis wichtige Generierung der Editoren für Domänen-spezifische Sprachen und deren Einbettung in Programmier- und Anwendungsumgebungen müsste allerdings noch erfolgen, da dies durch Übersetzertechnologie bisher wenig Beachtung gefunden hat.

References

[DUP+82] S. Drossopoulou, J. Uhl, G. Persch, G. Goos, M. Dausmann, and G. Winterstein. An attribute grammar for Ada. *ACM SIGPLAN Notices*, 17(6):334, 1982.

[EEK+12] S. Efftinge, M. Eysholdt, J. Köhnlein, S. Zarnekow, R. von Massow, W. Hasselbring, and M. Hanus. Xbase: implementing domain-specific languages for Java. In *Proceedings of the 11th International Conference on Generative Programming and Component Engineering*, pages 112–121. ACM, 2012.

[JLM+12] S. Jörges, A.L. Lamprecht, T. Margaria, I. Schaefer, and B. Steffen. A constraint-based variability modeling framework. *International Journal on Software Tools for Technology Transfer (STTT)*, 14:511–530, 2012.

[Joh75] S.C. Johnson. *Yacc: Yet another compiler-compiler*. Bell Laboratories, 1975.

[Jör11] S. Jörges. *Genesys: A Model-Driven and Service-Oriented Approach to the Construction and Evolution of Code Generators*. PhD thesis, Technical University of Dortmund, 2011.

[KHZ82] U. Kastens, B. Hutt, and E. Zimmermann. *GAG: A practical compiler generator*. Lecture Notes in Computer Science 141. Springer, 1982.

[KWS07] U. Kastens, W.M.C. Waite, and A.M. Sloane. *Generating Software from Specifications*. Jones & Bartlett Learning, 2007.

[Par07] T. Parr. *The definitive ANTLR reference: building domain-specific languages*. 2007.

[SBMP08] D. Steinberg, F. Budinsky, E. Merks, and M. Paternostro. *EMF: eclipse modeling framework*. Addison-Wesley Professional, 2008.

[SRC+12] I. Schaefer, R. Rabiser, D. Clarke, L. Bettini, D. Benavides, G. Botterweck, A. Pathak, S. Trujillo, and K. Villela. Software diversity: state of the art and perspectives. *International Journal on Software Tools for Technology Transfer (STTT)*, 14:477–495, 2012.

[TS07] Sven Efftinge und Arno Hasse Thomas Stahl, Markus Völter. *Modellgetriebene Softwareentwicklung: Techniken, Engineering, Management*. dpunkt.verlag, 2007.