An Annotated SLE Bibliography

The *SLECOURSE* project*

Version: *October 18, 2012*

Abstract

Software Language Engineering (SLE) is a particular view on Software Engineering (SE), which pays specific attention to the many software languages that are used in software development. These are not just programming languages, but also modeling languages, query and transformation languages, schema languages, and domain-specific languages. Thus, SLE is concerned with design, implementation, testing, deployment, and evolution of software languages as well as language-based software components.

The purpose of this annotated bibliography is to collect papers that could serve as background for concrete SLE courses. The bibliography may serve well beyond its purpose of supporting teaching. At this early stage of a bibliography, we may be well advised on focusing on more general papers that are likely to be useful for different SLE course designs as opposed to highly technical and specialized work. Should it happen that the bibliography contains confusingly many entries, then we can still impose some extra structure on the bibliography, e.g., based on grouping or tagging.

*http://slecourse.github.com/slecourse

References

[1] Tiago L. Alves and Joost Visser. A case study in grammar engineering. In *Software Language Engineering, First International Conference, SLE 2008, Toulouse, France, September 29-30, 2008. Revised Selected Papers*, volume 5452 of *Lecture Notes in Computer Science*, pages 285–304. Springer, 2009. Entry added by *Ralf Lämmel*. Paper publicly available online at *http://wiki.di.uminho.pt/twiki/pub/Personal/Tiago/Publications/grammar-eng.pdf*.

> In response to and prior to the publication of the "Grammaware Agenda" [11], there have been various published efforts on using grammarware engineering. The present paper is a strong example of leveraging and advancing several areas of grammarware engineering in single study. That is, the study is concerned with development of a VDM-SL grammar from its ISO standard language reference.

[2] Nikolaos Drivalos, Dimitrios S. Kolovos, Richard F. Paige, and Kiran Jude Fernandes. Engineering a dsl for software traceability. In *Software Language Engineering, First International Conference, SLE 2008, Toulouse, France, September 29-30, 2008. Revised Selected Papers*, volume 5452 of *Lecture Notes in Computer Science*, pages 151–167. Springer, 2009. Entry added by *Ralf Lämmel*. Paper non-publicly available online at *http://www.springerlink.com/content/d517t60440404548/*.

> The paper addresses the important SLE problem of traceability for related software artifacts in different software languages, at different levels of abstractions, related in different ways. The paper presents the engineering of the Traceability Metamodelling Language (TML), a metamodelling language dedicated to defining traceability metamodels.

[3] Sebastian Erdweg, Tillmann Rendel, Christian Kästner, and Klaus Ostermann. Sugarj: library-based syntactic language extensibility. In *Proceedings of the 26th Annual ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications, OOPSLA 2011, part of SPLASH 2011, Portland, OR, USA, October 22 - 27,*

2011, pages 391–406. ACM, 2011. Entry added by *Ralf Lämmel*. Paper publicly available online at *http://www.informatik.uni-marburg.de/~rendel/erdweg11sugarj.pdf*.

The paper describes SugarJ—an approach to enhance Java's library notion with syntactic sugar, hence providing an easy-to-use language extension approach. For instance, XML, Java closures, and XML schemas can be integrated into the Java language in this way. SugarJ leverages SDF for syntax definitions, Stratego for the underlying transformation framework. SugarJ provides an incremental parsing approach so that library imports can affect parsing past the import.

[4] Martin Erwig and Eric Walkingshaw. Semantics first! - rethinking the language design process. In *Software Language Engineering - 4th International Conference, SLE 2011, Braga, Portugal, July 3-4, 2011, Revised Selected Papers*, volume 6940 of *Lecture Notes in Computer Science*, pages 243–262. Springer, 2012. Entry added by *Ralf Lämmel*. Paper publicly available online at *http://web.engr.oregonstate.edu/~erwig/papers/SemanticsFirst_SLE11.pdf*.

The paper suggests a semantics-centric approach to language design as opposed to a more syntax-based one. Haskell is used as a metalanguage. General language operators are employed to adapt and grow sophisticated languages out of simple semantics concepts.

[5] Jean-Marie Favre, Dragan Gasevic, Ralf Lämmel, and Andreas Winter. Guest editors' introduction to the special section on software language engineering. *IEEE Trans. Software Eng.*, 35(6):737–741, 2009. Entry added by *Ralf Lämmel*. Paper publicly available online at *http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5353438*.

This is not a technical paper, but rather an extended introduction to a special issue on SLE. The text provides a relatively early but somewhat matured description of the notions SLE and software languages. Also, the papers of the special issue provide a good test harness for discussing these notions. See

[9] for another early informal definition of software languages and SLE.

[6] Florian Heidenreich, Jan Kopcsek, and Uwe Aßmann. Safe composition of transformations. *Journal of Object Technology*, 10:7: 1–20, 2011. Entry added by *Ralf Lämmel*. Paper publicly available online at *http://www.jot.fm/issues/issue_2011_01/article7.pdf*.

The paper provides an approach for the safe composition of (model) transformations. This problem is interesting from an SLE point of view because the transformations are in themselves linguistic artifacts, as much as the metamodels for the inputs and outputs, and finally the composition descriptions are linguistic artifacts, too. In addition, the achievement of safe composition requires technqiues that are overall informed by language design and language definition.

[7] Jakob Henriksson, Florian Heidenreich, Jendrik Johannes, Steffen Zschaler, and Uwe Aßmann. Extending grammars and metamodels for reuse: the reuseware approach. *IET Software*, 2(3):165–184, 2008. Entry added by *Ralf Lämmel*. Paper publicly available online at *http://fheidenreich.de/work/files/IET-Reuseware-2008.pdf*.

The paper captures some reusable language-design knowledge. That is, the paper supports the extension of a given core language with reuse-related expressiveness in the sense of partial programs or templates. (The approach is shown to be powerful enough to support, for example, a module system.) The observation is here that an increasing number of domain-specific languages calls for a more efficient and easier-to-use language design process where each new language can easily support some general expressiveness that is likely to be expected—such as the partial programs at hand. The approach is based on earlier work on Invasive Software Composition.

[8] Markus Herrmannsdoerfer, Sander Vermolen, and Guido Wachsmuth. An extensive catalog of operators for the coupled evolution of metamodels and models. In *Software Language Engineering - Third International Conference, SLE 2010, Eindhoven, The Netherlands, October 12-13, 2010, Revised Selected Papers*, volume 6563

of *Lecture Notes in Computer Science*, pages 163–182. Springer, 2011. Entry added by *Ralf Lämmel*. Paper publicly available online at *http://www.st.ewi.tudelft.nl/~vermolen/pmwiki/uploads/Main/HerrmannsdoerferVW10.pdf*.

The paper deals with the coupled evolution of metamodels and models (in a MOF/EMF-like context). Such coupling has been studied intensively by the SLE and MDE communities and the present paper aims at a particularly "complete" catalog of operators for coupled transformation.

[9] Anneke Kleppe. Why software language engineering? In *Software Language Engineering: Creating Domain-Specific Languages Using Metamodels*. Addison-Wesley Professional, 2008. Entry added by *Ralf Lämmel*. Paper publicly available online at *http://www.informit.com/articles/article.aspx?p=1312831*.

The paper is the motivational chapter from a textbook on SLE. The chapter describes the continuous increase of software languages and the trend of programming and modeling to blend into other. The author uses the term 'mogram' to capture programs, models, schemas, and queries. The chapter begins with the following quote by Victor Hugo: "An invasion of armies can be resisted, but not an idea whose time has come." See [5] for another early informal definition of software languages and SLE.

[10] Ralf Lämmel and Wolfram Schulte. Controllable combinatorial coverage in grammar-based testing. In *Testing of Communicating Systems, 18th IFIP TC6/WG6.1 International Conference, TestCom 2006, New York, NY, USA, May 16-18, 2006, Proceedings*, volume 3964 of *Lecture Notes in Computer Science*, pages 19–38. Springer, 2006. Entry added by *Ralf Lämmel*. Paper publicly available online at *http://homepages.cwi.nl/~ralf/ccc/paper.pdf*.

The paper describes a highly systematic and scalable approach to grammar-based testing. The grammar is essentially interpreted in a combinatorial sense and a number of well-defined

control mechanisms help with deriving test data of the intented shape, size, and number while also avoiding combinatorial explosion. The approach has been implemented in C# and used for grammar-based testing in the .NET platform. Combinatorial explorations constrasts with randomized test-data generation. The paper provides a broad discussion of related work including a reference to [14].

[11] Paul Klint and Ralf Lämmel and Chris Verhoef. Toward an engineering discipline for grammarware. *ACM Trans. Softw. Eng. Methodol.*, 14(3):331–380, 2005. Entry added by *Ralf Lämmel*. Paper publicly available online at *http://www.cs.vu.nl/grammarware/agenda/paper.pdf*.

The paper surveys SLE (or grammarware engineering) by making an inventory of grammarware, describing some problems with the proper engineering of grammarware, stating some promises of addressing grammarware engineering more seriously, listing some matured or emerging principles of grammarware engineering, and calling out a list of research challenges.

[12] Lukas Renggli, Tudor Gîrba, and Oscar Nierstrasz. Embedding languages without breaking tools. In *ECOOP 2010 - Object-Oriented Programming, 24th European Conference, Maribor, Slovenia, June 21-25, 2010. Proceedings*, volume 6183 of *Lecture Notes in Computer Science*, pages 380–404. Springer, 2010. Entry added by *Ralf Lämmel*. Paper publicly available online at *http://scg.unibe.ch/archive/papers/Reng10aEmbeddingLanguages.pdf*.

The paper describes an embedding approach for the implementation of domain-specific languages (DSLs). Specifically, DSLs are modeled as language extensions of the underlying host language. The approach addresses the challenge of providing the language extensions in a manner that they integrate well with the development tools of the host language. The paper presents the extensible system Helvetia which intercepts the compilation pipeline of the Smalltalk host language to

seamlessly integrate language extensions. See [18] for another approach on language embedding.

[13] Davide Di Ruscio, Ralf Lämmel, and Alfonso Pierantonio. Automated co-evolution of gmf editor models. In *Software Language Engineering - Third International Conference, SLE 2010, Eindhoven, The Netherlands, October 12-13, 2010, Revised Selected Papers*, volume 6563 of *Lecture Notes in Computer Science*, pages 143–162. Springer, 2011. Entry added by *Ralf Lämmel*. Paper publicly available online at *http://userpages.uni-koblenz.de/~laemmel/gmfco/paper.pdf*.

The paper studies coupled evolution in the context of Eclipse Modeling Framework (EMF) and the associated Graphical Modeling Framework (GMF). In this context, we face several coupled models; if the domain model changes, then several other models must co-change. The problem is interesting because of the heterogeneity of the involved models.

[14] Emin Gün Sirer and Brian N. Bershad. Using production grammars in software testing. In *Proceedings of the Second Conference on Domain-Specific Languages (DSL '99), Austin, Texas, USA, October 3-5, 1999*, pages 1–13. ACM, 1999. Entry added by *Ralf Lämmel*. Paper publicly available online at *http://www.cs.cornell.edu/people/egs/papers/kimera-dsl99.pdf*.

The paper shows how grammar-based test-data generation and an accompanying methodology of testing may be highly effective and scalable for testing language-based software. Previously, grammar-based testing was mainly focused on compiler testing. The paper specifically tests the Java virtual machine. See [10] for another grammar-based testing paper, which also refers to the present paper in context.

[15] Dave Thomas. The impedance imperative tuples + objects + infosets = too much stuff! *Journal of Object Technology*, 2(8):7–12, 2003. Entry added by *Ralf Lämmel*. Paper publicly available online at *http://www.jot.fm/issues/issue_2003_09/column1.pdf*.

The paper takes a critical look at data programming—specifically in the sense of CRUD (Create, Read, Update,

Delete). The discussion covers early keyed files, established means such as SQL and database access APIs, object-oriented databases, modern wrapping/mapping-based approaches (e.g., object/relational mapping). The column identifies various problems with data programming: diversity of data modeling and CRUD programming options and the practical need to mix them, difficulties of integrating different type systems and data query/transformation languages, proprietary developments, performance issues, and complexity of support technologies. The discussion also briefly touches some contenders that may address some problems with previous and established approaches. The paper is, in fact, a column, which may be a good starting point for searching for technical publications around this topic. This is one D. Thomas' columns referrenced in this bibliography [15, 17, 16] for their SLE relevance.

[16] Dave Thomas. The api field of dreams - too much stuff! it's time to reduce and simplify apis! *Journal of Object Technology*, 5(6):23–27, 2006. Entry added by *Ralf Lämmel*. Paper publicly available online at *http://www.jot.fm/issues/issue_2006_07/column3.pdf*.

The paper takes a critical look at the promises and realities of APIs (class libraries and frameworks). For instance, API designs are said to be complex (e.g., in terms of included unnecessary features) and to involve unnecessary variation (e.g., in terms of varying and copying familiar classes and methods). The methodology for producing APIs is also found problematic (e.g., in terms of the JSR process leading to many APIs with some of them serving business strategies rather than real users). The proposal is to strive for user-driven API design and uniformity. The paper is, in fact, a column, which may be a good starting point for searching for technical publications around this topic. This is one D. Thomas' columns referrenced in this bibliography [15, 17, 16] for their SLE relevance.

[17] Dave Thomas. Programming with models? modeling with code. the role of models in software development. *Journal of Object Technology*, 5(8):15–19, 2006. Entry added by *Ralf Lämmel*. Paper publicly available

online at *http://www.jot.fm/issues/issue_2006_11/column2.pdf*.

The paper complements the notion of model-driven engineering by the idea of expressing models as code. In this manner, the column provides an inspiring discussion on the differences between models and code, and the pros and cons of using designated modeling frameworks subject to manual or automated model-to-code transformation eventually, or expressive programming languages, which can express models conveniently, or mainstream languages with more encoding overhead for models. The paper is, in fact, a column, which may be a good starting point for searching for technical publications around this topic. This is one D. Thomas' columns referrenced in this bibliography [15, 17, 16] for their SLE relevance.

[18] Laurence Tratt. Domain specific language implementation via compile-time meta-programming. *ACM Trans. Program. Lang. Syst.*, 30(6), 2008. Entry added by *Ralf Lämmel*. Paper publicly available online at *http://eprints.mdx.ac.uk/5920/1/Tratt-domain_specifci_language_implementation.pdf*.

The paper describes an embedding approach for the implementation of domain-specific languages (DSLs). The approach is based on the Converge programming language and uses its compile-time meta-programming facility. The accompanying methodoogy is meant to simplify DSL implementations and improve their quality. See [12] for another approach on language embedding.