MPS publications page

Title: Publications

2020

1. Sofia Meacham, Vaclav Pech, Detlef Nauck

AdaptiveVLE: an integrated framework for personalised online education using MPS JetBrains domain-specific modelling environment

Abstract:

This paper contains the design and development of an Adaptive Virtual Learning Environment (AdaptiveVLE) framework to assist educators of all disciplines with creating adaptive VLEs tailored to their needs and to contribute towards the creation of a more generic framework for adaptive systems. Fully online education is a major trend in education technology of our times. However, it has been criticised for its lack of personalisation and therefore not adequately addressing individual students' needs. Adaptivity and intelligence are elements that could substantially improve the student experience and enhance the learning taking place. There are several attempts in academia and in industry to provide adaptive VLEs and therefore personalise educational provision. All these attempts require a multiple-domain (multi-disciplinary) approach from education professionals, software developers, data scientists to cover all aspects of the system. An integrated environment that can be used by all the multiple-domain users mentioned above and will allow for quick experimentation of different approaches is currently missing. Specifically, a transparent approach that will enable the educator to configure the data collected and the way it is processed without any knowledge of software development and/or data science algorithms implementation details is required. In our proposed work, we developed a new language/framework using MPS JetBrains Domain-Specific Language (DSL) development environment to address this problem. Our work consists of the following stages: data collection configuration by the educator, implementation of the adaptive VLE, data processing, adaptation of the learning path. These stages correspond to the adaptivity stages of all adaptive systems such as monitoring, processing and adaptation. The extension of our framework to include other application areas such as business analytics, health analytics, etc. so that it becomes a generic framework for adaptive systems as well as more usability testing for all applications will be part of our future work.

IEEE Access, Read paper

2. Andreas Prinz, Gergely Mezei

The Art of Bootstrapping

Abstract:

Language workbenches are used to define languages using appropriate meta-languages. Meta-languages are also just languages and can, therefore, be defined using themselves. The process is called bootstrapping and is often difficult to achieve. This paper compares four different bootstrapping solutions. The EMF environment and the Meta-Programming System (MPS) use a compiled bootstrapping for their own definition. The platforms LanguageLab and DMLA are using interpreted bootstrapping. This paper compares these kinds of bootstrapping and relates them to the definition of instantiation. Besides the structural aspects of the bootstraps, the dynamism is also elaborated. It is shown how the bootstrap is related to the execution environment. Finally, the level of changeability is also discussed. It is shown that all approaches are quite similar and provide very flexible environments.

MODELSWARD 2019. Communications in Computer and Information Science, vol 1161. Springer, Cham, Link

3. Sofia Meacham, Vaclav Pech, Detlef Nauck

Classification Algorithms Framework (CAF) to Enable Intelligent Systems Using JetBrains MPS Domain-Specific Languages Environment

Abstract:

This paper describes the design and development of a Classification Algorithms Framework (CAF) using the JetBrains MPS domain-specific languages (DSLs) development environment. It is increasingly recognized that the systems of the future will contain some form of adaptivity therefore making them intelligent systems as opposed to the static systems of the past. These intelligent systems can be extremely complex and difficult to maintain. Descriptions at higher-level of abstraction (system-level) have long been identified by industry and academia to reduce complexity. This research presents a Framework of Classification Algorithms at system-level that enables quick experimentation with several different algorithms from Naive Bayes to Logistic Regression. It has been developed as a tool to address the requirements of British Telecom's (BT's) data-science team. The tool has been presented at BT and JetBrains MPS and feedback has been collected and evaluated. Beyond the reduction in complexity through the system-level description, the most prominent advantage of this research is its potential applicability to many application contexts. It has been designed to be applicable for intelligent applications in several domains from business analytics, eLearning to eHealth, etc. Its wide applicability will contribute to enabling the larger vision of Artificial Intelligence (AI) adoption in context.

IEEE Access (Volume: 8), Read paper

2019

1. Andreas Prinz, Alexander Shatalin

How to Bootstrap a Language Workbench

Abstract:

Language workbenches are designed to enable the definition of languages using appropriate meta-languages. This makes it feasible to define the environments by themselves, as the meta-languages are also just languages. This approach of defining an environment using itself is called bootstrapping. Often, such bootstrapping is difficult to achieve and has to be built deeply into the environment. The platform Meta-Programming System (MPS) has used bootstrapping for its own definition. In a similar way, the environment LanguageLab is using bootstrapping for its definition. This paper reports the implementation of LanguageLab in MPS thereby also porting the bootstrapping. From the experiences general requirements for bootstrapping language workbenches are derived.

MODELSWARD 2019: Proceedings of the 7th International Conference on Model-Driven Engineering and Software Development, Link

2. M. Voelter, K. Birken, S. Lisson, A. Rimer

Shadow Models - Incremental Transformations for MPS

Abstract:

Shadow Models is an incremental transformation framework for MPS. The name is motivated by the realization that many analyses are easier to do on an model whose structure is different from what the user edits. To be able to run such analyses interactively in an IDE, these "shadows" of the user- facing model must be maintained in realtime, and incremen- tality can deliver the needed short response times. Shadow Models is an incremental model transformation engine for MPS. In the paper we motivate the system through example use cases, and outline the transformation framework.

SLE 2019 , PDF

2018

1. Markus Voelter

Fusing Modeling and Programming into Language-Oriented Programming

Abstract:

Modeling in general is of course different from programming (think: climate models). However, when we consider the role of models in the context of "model-driven", i.e., when they are used to automati- cally construct software, it is much less clear that modeling is different from programming. In this paper, I argue that the two are conceptually indistinguishable, even though in practice they traditionally emphasize different aspects of the (conceptually indistinguishable) common approach. The paper discusses and illustrates language-oriented programming, the approach to {modeling| programming} we have successfully used over the last 7 years to build a range of innovative systems in domains such as insurance, healthcare, tax, engineering and consumer electronics. It relies on domain-specific languages, modular language extension, mixed notations, and in particular, the Jetbrains MPS language workbench.

ISoLA 2018 - UVMP Track, PDF

2. Markus Voelter, Bernd Kolb, Federico Tomassetti, Patrick Alff, Laurent Wiart, Andreas Wortmann, Arne Nordmann

Using Language Workbenches and Domain-Specific Languages for Safety-Critical Software Development

Abstract:

Language workbenches support the efficient creation, integration, and use of domain-specific languages. Typically, they execute models by code generation to programming language code. This can lead to increased productivity and higher quality. However, in safety-/mission-critical environments, generated code may not be considered trustworthy, because of the lack of trust in the generation mechanisms. This makes it harder to justify the use of language workbenches in such an environment. In this paper, we demonstrate an approach to use such tools in critical environments. We argue that models created with domain-specific languages are easier to validate and that the additional risk resulting from the transformation to code can be mitigated by a suitably designed transformation and verification architecture. We validate the approach with an industrial case study from the healthcare domain. We also discuss the degree to which the approach is appropriate for critical software in space, automotive, and robotics systems.

SOSYM Online; Issue TBA, PDF

2017

1. Daniel Ratiu, Vaclav Pech, Kolja Dummann (P&I)

Experiences with Teaching MPS in Industry -- Towards Bringing Domain Specific Languages Closer to Practitioners

Conference paper (ACM/IEEE 20th International Conference on Model Driven Engineering Languages and Systems (MODELS'17)): PDF

2. S. M. Guttormsen, A. Prinz and T. Gjøsæter

Consistent Projectional Text Editors

Abstract:

For modelling and domain-specific languages, projectional editors have become popular. These editors implement the MVC pattern and provide a direct connection to the underlying model. In particular, projectional editors allow much more freedom in defining the concrete syntax than traditional grammars. The downside is that it is possible to define presentations that are of bad quality, and that this is not easily visible. In this article, we identify some of the problems with projectional editors and propose ways to resolve them. We also demonstrate a proof-of-concept solution, showing how problematic editor presentations could be identified automatically.

Paper: Available from: https://www.researchgate.net/publication/313900449_Consistent_Projectional_Text_Editors.

3. Markus Voelter, Bernd Kolb, Tamás Szabó, Daniel Ratiu, Arie van Deursen (SoSyM)

Lessons learned from Developing mbeddr: A Case Study in Language Engineering with MPS

Conference paper (ACM/IEEE 20th International Conference on Model Driven Engineering Languages and Systems

(MODELS'17)): Link

Abstract:

Language workbenches are touted as a promising technology to engineer languages for use in a wide range of domains, from programming to science to business. However, not many real-world case studies exist that evaluate the suitability of language workbench technology for this task. This paper contains such a case study. In particular, we evaluate the development of mbeddr, a collection of integrated languages and language extensions built with the Jetbrains MPS language workbench. mbeddr consists of 81 languages, with their IDE support, 34 of them C extensions. The mbeddr languages use a wide variety of notations—textual, tabular, symbolic and graphical—and the C extensions are modular; new extensions can be added without changing the existing implementation of C. mbeddr's development has spanned 10 person-years so far, and the tool is used in practice and continues to be developed. This makes mbeddr a meaningful case study of non-trivial size and complexity. The evaluation is centered around five research questions: language modularity, notational freedom and projectional editing, mechanisms for managing complexity, performance and scalability issues and the consequences for the development process. We draw generally positive conclusions; language engineering with MPS is ready for real-world use. However, we also identify a number of areas for improvement in the state of the art in language engineering in general, and in MPS in particular.

4. Daniel Ratiu, Markus Voelter, Domenik Pavletic

Automated testing of DSL implementations—experiences from building mbeddr

Paper: Read on-line

Abstract:

Domain-specific languages promise to improve productivity and quality of soft-ware development by providing problem-adequate abstractions to developers. Projectionallanguage workbenches, in turn, allow the definition of modular and extensible domainspecific languages, generators, and development environments. While recent advances inlanguage engineering have enabled the definition of DSLs and tooling in a modular and cost-effective way, the quality assurance of their implementation is still challenging. In this paper,we discuss our work on testing different aspects of the implementation of domain specificlanguages and associated tools, and present several approaches to increase the automationof language testing. We illustrate these approaches with the Jetbrains MPS language work-bench and our experience with testing mbeddr, a set of domain specific languages and toolson top of C tailored to embedded software development. Based on the experience gainedfrom the mbeddr project, we extract generic lessons for practitioners as well as challengeswhich need more research.

2016

1. Andreas Prinz

Multi-level Language Descriptions

Paper: PDF

Abstract: Language descriptions are a multi-level issue. In particular, the de nition and handling of instantiation semantics connects three lev- els. This paper looks at two language workbenches without support for multi-level modelling and their handling of the multi-level part of lan- guage descriptions. From these observations, the importance of runtime instantiation in terms of an underlying machine is established.

2. Andreas Wortmann, Martin Beet

DOMAIN SPECIFIC LANGUAGES FOR EFFICIENT SATELLITE CONTROL SOFTWARE DEVELOPMENT

Conference paper - PDF

Proc. ‘DASIA 2016’, DAta Systems In Aerospace Tallinn, Estonia, 10-12 May 2016 (ESA SP-736, August 2016)

3. Přemysl Vysoký

Grammar to JetBrains MPS Convertor

Diploma thesis: Read on-line

Abstract:
JetBrains MPS is a language workbench focusing on domain-specific languages. Unlike many other language workbenches and IDEs, it uses a projectional editor for code. The developer directly manipulates the program in its tree form (AST) and not by editing a text source code. This brings many advantages, but on the other hand requires time-consuming and complicated MPS language definition. The thesis elaborates on the possibility of automating the process of creating MPS language definition from its grammar description. It introduces the MPS editor, evaluates approaches of related projects and describes author's efforts to implement an MPS plugin that allows this import. The chosen approach and the selection of tools used for implementation are justified in the thesis. We point out important problems that any similar project might deal with and we introduce some possible solutions. Furthermore, the thesis contains examples of imported languages, showing the potency of the chosen approach. The thesis also aims to lay groundwork for future extensions and suggest possible improvements.

4. Jonáš Klimeš

Domain-Specific Language for Learning Programming

Diploma thesis - PDF, source code

In the scope of this thesis, we designed a language for programming education. At first, we described eight existing tools for learning programming and identified key features in the learning process. Second, we designed an educational domain-specific language Eddie. Eddie is suitable for teenagers and adults who want to learn programming. It uses a domain based on Karel the Robot language, where users can control a robot character in a two-dimensional grid. We implemented a prototype of Eddie using the MPS Language Workbench and its projectional editor. The Eddie language gradually introduces loops, conditionals, variables, functions, and objects. Eddie programs can be created, executed and visualized in the Eddie Studio IDE.

2015

1. S. Erdweg, T. van der Storma, M. Völter, L. Tratt et al.

Evaluating and Comparing Language Workbenches - Existing Results and Benchmarks for the Future

Journal paper - PDF

Abstract: Language workbenches are environments for simplifying the creation and use of com- puter languages. The annual Language Workbench Challenge (LWC) was launched in 2011 to allow the many academic and industrial researchers in this area an opportunity to quantitatively and qualitatively compare their approaches. We first describe all four LWCs to date, before focussing on the approaches used, and results generated, during the third LWC. We give various empirical data for ten approaches from the third LWC. We present a generic feature model within which the approaches can be understood and contrasted. Finally, based on our experiences of the existing LWCs, we propose a number of benchmark problems for future LWCs.

2. M. Voelter, A. van Deursen, B. Kolb, S. Eberle

Using C Language Extensions for Developing Embedded Software - A Case Study

Conference paper - PDF

Abstract: We report on an industrial case study on developing the embedded software for a smart meter using the C programming language and domain-specific extensions of C such as components, physical units, state machines, registers and interrupts. We find that the extensions help significantly with managing the complexity of the software. They improve testability mainly by supporting hardware-independent testing, as illustrated by low integration efforts. The extensions also do not incur significant overhead regarding memory consumption and performance. Our case study relies on mbeddr, an extensible version of C. mbeddr, in turn, builds on the MPS language workbench which supports modular extension of languages and IDEs.

3. D. Pavletic, M. Voelter, S. A. Raza, B. Kolb, T Kehrer

Extensible Debuggers for Extensible Languages (Ada Europe 2015)

Conference paper - PDF

Language extension enables integration of new language constructs without invasive changes to a base language (e. g., C). Such extensions help to build more reliable software by using proper domain-specific abstractions. Language workbenches significantly reduce the effort for building such extensible languages by synthesizing a fully-fledged IDE from language definitions. However, in contemporary tools, this synthesis does not include interactive debugging for programs written with the base language or its extensions. This paper describes a generic framework for extensible debuggers that enables debugging of the language extensions by definig mappings between the base language and the language extensions. The architecture is designed for extensibility, so debug support for future extensions can be contributed with little effort. We show an implementation of our approach for mbeddr, which is an extensible version of the C programming language. We also discuss the debugger implementation for non-trivial C extensions such as components. Finally, the paper discusses the extent to which the approach can be used with other base languages, debugger backends and language workbenches.

2014

1. M. Voelter, J. Siegmund, T. Berger, B. Kolb.

Towards User-Friendly Projectional Editors

Conference paper - PDF

Abstract: Today's challenges for language development include language extension and composition, as well as the use of diverse notations. A promising approach is projectional editing, a technique to directly manipulate the abstract syntax tree of a program, without relying on parsers. Its potential lies in the ability to combine diverse notational styles -- such as text, symbols, tables, and graphics -- and the support for a wide range of composition techniques. However, projectional editing is often perceived as problematic for developers. Expressed drawbacks include the unfamiliar editing experience and challenges in the integration with existing infrastructure. In this paper we investigate the usability of projectional editors. We systematically identify usability issues resulting from the architecture. We use JetBrains Meta Programming System (MPS) as a case study. The case study discusses the concepts that MPS incorporates to address the identified issues, evaluates effectiveness of these concepts by surveying professional developers, and reports industrial experiences from realizing large-scale systems. Our results show that the benefits of flexible language composition and diverse notations come at the cost of serious usability issues -- which, however, can be effectively mitigated with facilities that emulate editing experience of parser-based editors.

Citation: M. Voelter, J. Siegmund, T. Berger, B. Kolb. Towards User-Friendly Projectional Editors. Proc. of the 7th Intl. Conf. on Software Language Engineering (SLE 2014), 20 pages, 2014

2. M. Voelter, S. Lisson.

Supporting Diverse Notations with MPS’ Projectional Editor

Workshop paper - PDF

Abstract: To be able to build effective DSLs, these DSLs must not just use language concepts that are aligned with their respective domain, but also use notations that correspond closely to established domain notations -- and those are often not purely textual or graphical. The underlying language workbench must support these notations, and combining different notations in a single editor must be supported as well in order to support the coherent definitions of systems that use several DSLs. In this paper we provide an overview over the notations supported by JetBrains MPS. MPS is a language workbench that uses a projectional editor, which, by its very nature, can deal with many different notational styles, including text, prose, math tables and graphics. The various supported notations are illustrated with examples from real-world systems.

Citation: M. Voelter, S. Lisson. Supporting Diverse Notations with MPS’ Projectional Editor, 2nd International Workshop on The Globalization of Modeling Languages, MODELS 2014

3. Markus Voelter

Generic Tools, Specific Languages

PhD thesis - http://voelter.de/books/

http://www.amazon.com/Generic-Specific-Languages-Markus-Voelter/dp/1500359432/

Abstract: Generic Tools, Specific Languages (GTSL) is an approach for developing tools and applications in a way that supports easier and more meaningful adaptation to specific domains. To achieve this goal, GTSL generalizes programming language IDEs to domains traditionally not addressed by languages and IDEs. At its core, GTSL represents applications as

documents/programs/models expressed with suitable languages. Application functionality is provided through an IDE that is aware of the languages and their semantics. The IDE provides editing support, and also directly integrates domain-specific analyses and execution services. Applications and their languages can be adapted to increasingly specific domains using language engineering; this includes developing incremental extensions to existing languages or creating additional, tightly integrated languages. Language workbenches act as the foundation on which such applications are built.

4. Fabien Campagne

The MPS Language Workbench, Volume I

Book - http://books.campagnelab.org

Abstract: The MPS Language Workbench, Volume I. The first volume of the series is both a simple introduction to the JetBrains MPS language workbench and a complete reference manual.

Citation: Fabien Campagne. The MPS Language Workbench, Volume I. March 2014.

5. Federico Tomassetti

Polyglot software development

PhD thesis - http://porto.polito.it/2537697/1/TOMASSETTI.FEDERICO.THESIS.pdf

Abstract: In the first part of this dissertation we study the adoption of modeling and domain specific languages. On the basis of an industrial survey we individuate a list of benefits attainable through these languages, how frequently they can be reached and which techniques permit to improve the chances to obtain a particular benefit. In the same way we study also the common problems which either prevent or hinder the adoption of these languages. We then analyze the processes through which these languages are employed, studying the relative frequency of the usage of the different techniques and the factors influencing it. Finally we present two case-studies performed in a small and in a very large company, with the intent of presenting the peculiarities of the adoption in different contexts.

6. M. Voelter, B. Kolb, J. Warmer

Projecting a Modular Future

Journal paper - PDF

Abstract: We describe two innovations in programming languages: modularity and projectional editing. Language modularity refers to the ability to combine independently developed languages without changing their respective definitions. A language is not anymore a fixed quantity, instead it can be extended with domain-specific constructs as needed. Projectional editing refers to a technique of building editors and IDEs that avoid the need for parsers. They support a wide range of tightly integrated notations including textual, symbolic, tabular and graphical. In addition, by avoiding parsers, the well-known limitations of grammar composition are avoided as well. The article illustrates the consequences of these two innovations for the design of (programming) languages with three examples. First, we discuss a set of modular extensions of C for embedded programming that enables efficient code generation and formal analysis. Second, we discuss a language for requirements engineering that flexibly combines structured and unstructured (prose) data. Third, we illustrate a language for defining insurance rules that makes use of mathematical notations. All examples rely on the open source JetBrains MPS language workbench.

Citation: M. Voelter, B. Kolb, J. Warmer. Projecting a Modular Future. IEEE Software, in press

2013

1. Manuele Simi, Fabien Campagne

Composable languages for bioinformatics: the NYoSh experiment

Paper - PDF

Abstract: Language WorkBenches (LWBs) are software engineering tools that help domain experts develop solutions to various classes of problems. Some of these tools focus on non-technical users and provide languages to help organize knowledge while other workbenches provide means to create new programming languages. A key advantage of language workbenches is that they support the seamless composition of independently developed languages. This capability is useful when developing

programs that can benefit from different levels of abstraction. We reasoned that language workbenches could be useful to develop bioinformatics software solutions. In order to evaluate the potential of language workbenches in bioinformatics, we tested a prominent workbench by developing an alternative to shell scripting. To illustrate what LWBs and Language Composition can bring to bioinformatics, we report on our design and development of NYoSh (Not Your ordinary Shell).

Citation: Simi M, Campagne F. (2014) Composable languages for bioinformatics: the NYoSh experiment. PeerJ 2:e241 http://dx.doi.org/10.7717/peerj.241

2. M. Voelter, D. Ratiu and F. Tomassetti

Requirements as First-Class Citizens: Integrating Requirements with Implementation Artifacts

Workshop paper - PDF

Abstract: Requirements often play second fiddle in software development projects. The tools for managing requirements often just support "numbered lists of prose paragraphs", and they don't integrate well with the tools used for implementing the system. This leads to all kinds of challenges in terms of versioning and traceability. Moreover, because they are mainly prose text, they cannot easily be checked for consistency and completeness, limiting their usefulness. In this paper we describe an alternative approach, where requirements are (at least partially) formalized to support consistency checking, where parts of requirements can be used directly as the implementation, and where requirements are managed with the same tools that are used for system development. The approach is illustrated with the mbeddr system, a comprehensive IDE for embedded software development based on an extensible version of C and domain-specific languages.

Citation: M. Voelter, D. Ratiu and F. Tomassetti. Requirements as First-Class Citizens: Integrating Requirements with Implementation Artifacts. 6th International Workshop on Model-Based Architecting/Construction of Embedded Systems (ACES-MB) 2013, 10 pages, MODELS 2013

3. M. Voelter

Integrating Prose as a First-Class Citizen with Models and Code

Workshop paper - PDF

Abstract: In programming and modeling we strive to express structures and behaviors as formally as possible to support tool-based processing. However, some aspects of systems cannot be described in a way that is suitable for tool-based consistency checking and analysis. Examples include code comments, requirements and software design documents. Consequently, they are often out-of-sync with the code and do not reflect the current state of the system. This paper demonstrates how language engineering based on language workbenches can help solve this problem by seamlessly mixing prose and program nodes. These program nodes can range from simple references to other elements over variables and formulas to embedded program fragments. The paper briefly explains the language engineering techniques behind the approach as well as a number of prose-code integrated languages that are part of mbeddr, an integrated language and tool stack for embedded software engineering.

Citation: M. Voelter. Integrating Prose as a First-Class Citizen with Models and Code. 7th International Workshop on Multi-Paradigm Modeling (MPM'13), 10 pages, MODELS 2013

4. Václav Pech, Alexander Shatalin, Markus Voelter

JetBrains MPS as a Tool for Extending Java

Paper PDF

Abstract: JetBrains MPS is an integrated environment for language engineering. It allows language designers to define new programming languages, both general-purpose and domain-specific, either as standalone entities or as modular extensions of already existing ones. Since MPS leverages the concept of projectional editing, non-textual and non-parseable syntactic forms are possible, including tables or mathematical symbols. This tool paper introduces MPS and shows how its novel approach can be applied to Java development. Special attention will be paid to the ability to modularize and compose languages.

Citation: V. Pech, A. Shatalin, M. Voelter, PPPJ '13 Proceedings of the 2013 International Conference on Principles and Practices of Programming on the Java Platform: Virtual Machines, Languages, and Tools, Pages 165-168

5. M. Voelter, D. Ratiu, B. Kolb, B. Schaetz, mbeddr

Instantiating a Language Workbench in the Embedded Systems Domain

Journal paper - PDF

Abstract: Tools can boost software developer productivity, but building custom tools is prohibitively expensive, especially for small organizations. For example, embedded programmers often have to use low-level C with limited IDE support, and integrated into an off-the-shelf tool chain in an ad-hoc way. To address these challenges, we have built mbeddr, an extensible language and IDE for embedded software development based on C. mbeddr is a large- scale instantiation of the Jetbrains MPS language workbench. Exploiting its capabilities for language modularization and composition, projectional editing and multi-stage transformation, mbeddr is an open and modular framework that lets third parties add extensions to C with minimal effort and without invasive changes. End users can combine extensions in programs as needed. To illustrate the approach, in this paper we discuss mbeddr's support for state machines, components, decision tables, requirements tracing, product line variability and program verification and outline their implementation. We also present our experience with building mbeddr, which shows that relying on language workbenches dramatically reduces the effort of building customized, modular and extensible languages and IDEs to the point where this is affordable by small organizations. Finally, we report on the experience of using mbeddr in a commercial project, which illustrates the benefits to end users.

Citation: M. Voelter, D. Ratiu, B. Kolb, B. Schaetz. mbeddr - Instantiating a Language Workbench in the Embedded Systems Domain. Journal of Automated Software Engineering, September 2013, Volume 20, Issue 3, pp 339-390, 51 pages, 2013

6. M. Voelter

DSL Engineering

Book: http://dslbook.org

Abstract: The book provides a thorough read on the subject, introducing the reader into the core aspects of DSL design and implementation. It explains the fundamental principles of using DSLs as part of modern development cycle showcasing JetBrains MPS as one of the tools that aim to make DSL development approachable.

7. Philipp Zech, Michael Felderer, Matthias Farwick, Ruth Breu

A Concept for Language--oriented Security Testing

Paper: PDF

Citation: Zech, P., Felderer, M., Farwick, M., & Breu, R. (2013). A concept for language-oriented security testing. In Proceedings - 7th International Conference on Software Security and Reliability Companion, SERE-C 2013 (pp. 53--62). Ieee. doi:10.1109/SERE-C.2013.16

8. M. Voelter, D. Ratiu, B. Kolb, B. Schaetz.

mbeddr: an Extensible C-based Programming Language and IDE for Embedded Systems

Conference paper - PDF

Abstract: While the C programming language provides very good support for writing efficient, low-level code, it does not offer adequate means for defining higher-level abstractions relevant to embedded software. In this paper we present the mbeddr technology stack that supports extension of C with constructs adequate for embedded systems. In mbeddr, efficient low-level programs can be written using the well-known concepts from C. Higher-level domain-specific abstractions can be seamlessly integrated into C by means of modular language extension regarding syntax, type system, semantics and IDE. In the paper we show how language extension can address the challenges of embedded software development and report on our experience in building these extensions. We show that language workbenches deliver on the promise of significantly reducing the effort of language engineering and the construction of corresponding IDEs. mbeddr is built on top of the JetBrains MPS language workbench. Both MPS and mbeddr are open source software.

Citation: M. Voelter, D. Ratiu, B. Kolb, B. Schaetz. mbeddr: an Extensible C-based Programming Language and IDE for Embedded Systems. Proc. of the 3rd annual conference on Systems, programming, and applications: software for humanity (SPLASH), ACM, pp 121-140, 20 pages, 2013

Older

1. Tomáš Fechtner

MPS-based Domain-specific Languages for real time Java development

Diploma thesis - PDF

Abstract: The Real-time Specification of Java (RTSJ) is an intention to introduce Java as a language for developing real-time system. However, the complexity of their development and a non-trivial programming model of RTSJ with its manual memory management often lead to programming errors. To mitigate the development of RTSJ systems it would be beneficial to provide an internal domain-specific language (DSL) extending the Java language which would allow to develop the systems in more intuitive and safer way. However, it is needed to find compromise between solution's power and level of usability, because this two attributes go often against each other. One possible way of DSLs creation concerns the Meta-Programming System (MPS). It allows to develop new domain- specific languages and corresponding projectional editors enabling different views on code. This thesis proposes a design and implementation of the DSL on the top of the MPS platform and corresponding code generator enabling development of RTSJ systems. Furthermore, the thesis provides a simple case-study to evaluate a proposed DSL. Additionally, the thesis assesses the suitability of MPS as a DSL-development platform.

2. M. Voelter.

Language and IDE Modularization and Composition with MPS

Conference paper - PDF

Abstract: JetBrains MPS is an Open Source language workbench. It uses projectional editing, which means that no parser is involved, and textual, tabular and graphical notations are handled with the same approach. As a consequence, it is an ideal language and IDE development environment. This tutorial has three parts: In part one, I will give a very short introduction to projectional editing. Part two provides examples of existing (modular) languages developed with MPS, showcasing the benefits of the tool and its possibilities. In part three, the longest one, I will show how to define languages in MPS, reusing/extending existing language modules. The tutorial will be mostly live demos; only part one will use a small number of slides.

Citation: 5. M. Voelter. Language and IDE Modularization and Composition with MPS. International Summer School on Generative and Transformational Techniques in Software Engineering, GTTSE 2011, LNCS 7680, pp 383-430, 47 pages, 2011

3. Markus Voelter, Eelco Visser.

Product Line Engineering with Projectional Language Workbenches

Paper - PDF

Abstract: This paper investigates the application of domainspecific languages in product line engineering (PLE). We start by analyzing the limits of expressivity of feature models. Feature models correspond to context-free grammars without recursion, which prevents the expression of multiple instances and references. We then show how domain-specific languages (DSLs) can serve as a middle ground between feature modeling and programming. They can be used in cases where feature models are too limited, while keeping the separation between problem space and solution space provided by feature models. We then categorize useful combinations between configuration with feature model and construction with DSLs and provide an integration of DSLs into the conceptual framework of PLE. Finally we show how use of a consistent, unified formalism for models, code, and configuration can yield important benefits for managing variability and traceability. We illustrate the concepts with several examples from industrial case studies.

4. Markus Voelter

Embedded Software Development with Projectional Language Workbenches

Paper - PDF

Abstract: This paper describes a novel approach to embedded software development. Instead of using a combination of C code and modeling tools, we propose an approach where modeling and programming is unified using projectional language workbenches. These allow the incremental, domain-specific extension of C and a seamless integration between the various concerns of an embedded system. The paper does not propose specific extensions to C in the hope that everybody will use them; rather, the paper illustrates the benefits of domain specific extension using projectional editors. In the paper we describe the problems with the traditional approach to embedded software development and how the proposed approach can solve them. The main part of the paper describes our modular embedded language, a proof-of-concept implementation of the approach based on JetBrains MPS. We implemented a set of language extensions for embedded programming, such as state machines, tasks, type system extensions as well as a domain specific language (DSL) for robot control. The language modules are seamlessly integrated, leading to a very efficient way for implementing embedded software.

5. M. Voelter

Implementing Feature Variability for Models and Code with Projectional Language Workbenches

Workshop paper - PDF

Abstract: It is often considered a binary decision whether something is domain specific or not. Consequently, there are domain specific languages (DSL) and general purpose languages (GPL), there are domain specific and non-domain specific modeling tools, there are domain specific and non-domain specific methodologies etc. In this paper we argue, that domain specificity is not a hard decision, but rather one extreme on a continuum. We also argue that many systems can be more efficiently described with a mix of domain specific and non-domain specific abstractions. This view of the world has consequences for languages, tools and methodologies, specifically the ability to modularize and compose languages. Additionally we outline these consequences and provide an extensive example based on embedded systems.

Citation: M. Voelter. Implementing Feature Variability for Models and Code with Projectional Language Workbenches. 2nd International Workshop on Feature-Oriented Software Development (FOSD) 2010, pp 41 – 48, 8 pages, 2010

Submit new article proposals

Have you read a new article about MPS? Have you written one? Let us know so we could add it to this list.