DSLs Should be Online Applications

Dominykas Barisas[1], Adam Duracz[2] and Walid Taha[3,4]

[1] *Kaunas University of Technology, Kaunas, Lithuania, dominykas.barisas@ktu.lt*
[2] *Halmstad University, Halmstad, Sweden, adam@duracz.net*
[3] *Halmstad University, Halmstad, Sweden, walid.taha@hh.se*
[4] *Rice University, Houston, TX, USA, taha@rice.edu*

Abstract

Domain-Specific Languages (DSLs) play an important role in both practice and education. But developing them is challenging, because a DSL must ultimately satisfy a large and complex set of user/customer requirements to fulfil its intended role, and neither requirements nor users are fully available at all times during the development process. Requirements can be elicited using agile methods but such methods assume the availability of the users. The situation is further complicated when the user base is primarily students and when enhanced learning is a key requirement. In this paper we propose developing DSLs, especially educational ones, as online applications. We analyze how this can help requirement elicitation and learning. Being online brings language development closer to the user, yielding new opportunities to improve and accelerate the language design process. It is also well-matched to agile methods, since web-based analytics provide an abundant source of data that integrates naturally into the development process. As an example, we consider applying the method to Acumen, a DSL designed to support teaching Cyber-Physical Systems.

Keywords: *Agile methods, Domain-Specific Languages (DSLs), Cyber-Physical Systems (CPS), Web-based Applications, Staging, Education.*

1. Introduction

Domain-Specific Languages (DSLs) target a particular application domain in order to improve the efficiency of the product development process, and to capitalize on the knowledge accumulated by domain experts. [1] [2] The benefits of DSLs are generally the fruit of the efforts of a DSL developer. Much of the complexity of the developer's job stems from the need to simultaneously articulate, formalise, and implement the specifications for the DSL. Unfortunately, there is scarce practical advice on how to address the unusual difficulties in the elicitation of requirements that often arise when developing DSLs. Agile software development methods can be helpful, as they respond to incomplete or evolving specifications. [3] [4] In the agile approach, requirements and solutions are developed collaboratively by members of self-organizing teams. Software is built and tested in short sequential iterations, where a version of the entire product has to be completed before the next one can begin. High priority features are developed in earlier iterations. This way, feedback from the customer can be gathered on a working prototype of the final system, and used to improve later iterations. The key features of agile methods are: 1) A full system is produced in each iteration (short release); 2) Continual (or "constant") testing; 3) Continual requirements development, testing, and code reviews, and 4) Small development teams. [5]

Taking agile development methods into account, the question is then how, and to what extent, these methods can be applied to DSL development, and how could the process be improved? There are some missing links between what the existing methods provide and what effective DSL development needs. In particular, the specific sources of complexity in DSL development are as follows:

1. Current research traditions separate language design and language implementation into two distinct sub-disciplines, but an agile method for DSL development needs to combine both;
2. The customer base is often inaccessible. Ideally, for a DSL, the customer base should be anyone working in that domain, which is often a non-trivial number of users;

3. The customer base is usually not in a good position to specify a language, not to mention the *best* language for that domain. The customer may not yet be an expert in the best practices and notations for the domain. Specifying languages requires advanced training in mathematics and logic. Additionally, because of the large input space, the customer demonstrations that normally follow each iteration in an agile project are not so straight-forward.
4. The end-user is not always able to access or start using the simulation system without a preliminary knowledge and preparation. Equally, enabling the customer to give constructive feedback on possible enhancements of the simulation system is not trivial. The challenge is to make the system intuitive, user-friendly and rewarding, so that the customer feels satisfied with each small achievement and willing to take another step to move forward.

1.1. Contributions

The main contribution presented in this paper is the insight that many benefits can be derived from developing a DSL and its development environment as a multi-user online application. In particular, developing a DSL as an online application provides a natural and efficient workflow to achieve the following:

1. End-user feedback (traditionally obtained through agile practices such as sprint reviews) is acquired continuously through the online DSL development environment, using analytics and direct feedback facilities. User feedback is enriched with appropriate context, in the form of the interaction and source revision history that led up to the feedback submission.
2. Requirements (traditionally embodied in agile practices such as user stories), such as those of the need for language idioms or development environment features, are mined from user DSL source files and revision histories, and interactions with the online user interface.
3. Support for planning (as in the prioritization of items in a sprint backlog) is provided by analytics. For example, bugs that affect many users are prioritized.

Developing DSL programs online brings the users and designers closer by erasing some of the communication boundaries associated with the standalone (client-side) model development. The agile promise of a tight feedback loop, based on interactive meetings between product owners and developers, is enhanced with non-interactive (cheaper and more frequent) feedback through monitoring and analytics. In the rest of this paper we define the proposed method and discuss its application to a modeling language being developed by some of the authors.

2. Proposed Method

To address the first source of complexity, we propose an extension to the traditional development process, focusing on the use of interpreters and translators rather than traditional compilers. This method keeps the code small, flexible and consistent with the agile development value of prioritizing code over documentation. A good interpreter can be viewed as a formal specification for a language. This unifies the problems of developing the requirements (the DSL design) and developing the formal semantics. A technique known as *staging* [6] (which is similar to the recently-proposed idea of a skeleton design method [7]) can then be used to turn interpreters into translators/compilers without significant changes to the structure or size of the code base.

To address the remaining three sources of complexity, we propose developing language implementations that are delivered as online applications, with an offline functionality provided to support use when network connectivity is not available. Online functionality is the key mechanism for reaching out to a broad user base. It also facilitates use and adds value through collaboration, openness, light-weight social networking, automatic grading and gateway/gamification [8] [9] processes. Gamification techniques can provide frequent hints to the user and increase feedback, both of which are useful in the development process. Key concepts from gamification include comprehensive navigation, keeping the user informed of what to do and explaining how answers are correct or incorrect. This makes it possible to determine where the user got stuck, where the learning curve was too steep, or simply where there is a bug in the system. It also helps identify to what degree the DSL is understandable to users and to leverage the

feedback to make the language more intuitive.

In the offline mode, we should at least have the equivalent of the traditional, standalone implementation. For example, local files should continue to be accessible and executable. Beyond this minimal functionality, caching and version control can help support higher degrees of independence and robustness to network availability. Interestingly, some of the implementation techniques used to support offline operation can also benefit the online mode. For example, caching can improve remote page load time and bandwidth consumption.

Examples of specific methods of how feedback collection affects requirements and how they can be incorporated naturally into an agile DSL development method are presented in Table 1.

Table 1. Amendments to common agile practices that aim to make language development more effective.

Agile practice	**Definition**	**Amendments for DSL development**
Iteration	Pre-defined period of time during which a set of requirements is to be implemented and integrated into a working version of the software.	Each iteration ends with a demonstration to the product owner, as well as a release of the DSL to the full user base. After a minor release, users retain access to the previous version, so that a version of the language compatible with their models is available. Porting guides, prepared for each release, help users transition to new versions when backward compatibility is broken.
Iteration Review Meeting	Meeting that follows each iteration, during which stakeholders provide feedback, based on a demonstration of the current version of the product.	Mechanized requirements elicitation. Feedback is obtained passively: 1. Through analysis of usage metrics and session logs; and interactively: 2. Using feedback forms. Here, access to passive feedback, such as session logs, can enhance the information actively provided by the user; 3. By up/down-voting of language and development environment features.
Iteration Planning Meeting	Meeting that precedes each iteration, where priorities for the iteration are established by the product owner. These are used to determine the iteration backlog, which specifies what will be done during the iteration.	Multi-dimensional prioritization. Criteria provided by the product owner is supplemented with metrics obtained through the mechanized feedback facilities. For example, features with many up-votes and few down-votes can be given a high user priority.

A key observation to take away from this table is that the proposed amendments rely on information that is readily available to DSL developers when the language is made available through an online development environment.

3. Application to the Development of Acumen

We first recognized the difficulties in DSL design outlined above in the context of ongoing work on an educational DSL called Acumen. In this section, we explain how the proposed method can be applied to improve the design and development process of Acumen.

The emergence of Cyber-Physical Systems (CPS) as a cross-disciplinary paradigm for innovation has prompted activity to develop DSLs for this domain. Developing a DSL for modeling and simulation of

CPS adds complexity stemming from the multitude of more specialized disciplines that make up the CPS paradigm. In addition to this, a DSL to support education demands further requirements relating to accessibility, both conceptual and financial. Developing an application that should meet all these expectations is clearly a daunting task, justified only by the high potential impact for such a tool. Examples include simulation tools such as Simulink [10], OpenModelica [11] SimScape [12], EnergyNavigator [13], and the Hybrid Quartz language. [14] [15] Despite this seemingly large selection, many fundamental challenges presented by the CPS domain are still not tackled adequately by any single available DSL, including: integration of models of systems from different sub-domains [16], simulation of Zeno systems [17], and the modeling of timing constraints. [18]

To address some of these research and educational problems, we are developing Acumen. [19]-[26] A particular educational challenge for CPS is its multidisciplinary nature, as it encompasses such academic disciplines as software engineering, networking, learning theory, electrical engineering and mechanical engineering. Often, advanced expertise from each of these disciplines must be integrated through close collaborations in order to develop a new Cyber-Physical System. In a research university environment, it is not unusual for an educational DSL to need to simultaneously address such a wide range of challenges. But to approach a challenge this complex, simplicity and accessibility in the language design are essential. In the case of Acumen, this is reflected in its adherence to a small core textual language and a minimalist, self-contained custom development environment. The language and environment have already been used in teaching three editions of a new course on CPS [25]-[27] with Acumen greatly influencing both the language and the teaching format of the course. Using the language in the class helps make the mathematics of the CPS theory more approachable. At the same time, exposing the designers to the real needs of the successive classes of students provided a steady influx of feedback and new requirements. [25] [26] This experience drew our attention to the particular challenge of DSL development, namely, the need for frequent and precise collection of requirements from users. This challenge arises with both application-oriented and educational DSLs, but is more challenging in the latter case because it has aims that transcend the specific product being developed in each use of the DSL.

3.1. Basic Design for an Online Version of Acumen

So far, Acumen has been developed and used as a standalone, client-side application. To apply the method described above, we need to select a specific path for development and evolution. We propose the most basic functionality, with an initial version of the Acumen platform incorporating the following:

1. Development of an online graphical user interface (editor, plotter, table, 3D simulation view), logging and analytics, and a user identification system, while ensuring reasonable functionality even in the absence of a network;
2. Unit- and property-based testing;
3. Incorporation of gateway/gamification processes;
4. Support for model sharing and co-simulation;

Once these functions are realized, we expect that there will be enough analytics about application and language usage to enable the identification of strategies for improving the system and the exercise sets it supports.

The general approach for system evaluation is to see if it can increase both the utility and the number of users of the Acumen language. This can be done in two arenas:

1. Within the CPS class taught at Halmstad University;
2. To the open user base outside the class.

Examples of specific analytics that can be considered include:

1. Usability evaluation by observing how features of the system are used, and modifying arrangements of the interface, to influence frequency of use of certain features;
2. Evaluating the effect of co-simulation on user engagement;
3. Studying analytics of language use in terms of user engagement, new users, returning users;
4. Evaluating the time and effort it takes to complete exercises sets;

In practice, the specific metrics used should be determined on the basis of several factors, including ease of implementation, interaction with support for mixed online and offline operation, effectiveness in demonstrating key results, and impact on runtime performance of the system.

4. Conclusion and Future Work

Prior experience with DSL development revealed differences and shortcomings in the efficacy of traditional development methods, compared to regular software development projects, and led to a proposal for a new method. An initial survey of DSL development methods and evaluation of feedback gathered from students using a standalone platform has shown the need for the amendment of standard agile practices, promoting better communication with system users and involving them at an early stage of development. This paper has explained how developing DSLs as online applications can facilitate their development, and has outlined a plan to evaluate the effectiveness of the method, using the Acumen language as a case study. Some of the key benefits of developing an educational DSL as an online application include:

- It facilitates the gathering of usability statistics, allowing improved user experience and possibly speeding up the education process;
- It encourages the interactive use of educational exercises and gateway-based learning processes, enabling the gathering of user-input and rewarding the user through gamification methods;
- It enables communication between users, promoting learning and helping consolidate and develop user buy-in for requirements as they evolve.

A priority for our future work will be to implement the proposed approach for quantitative evaluation. Our experience with Acumen will allow us to compare metrics such as user interest, activity, and time required to learn the language.

5. Acknowledgements

This work was supported by the US NSF CPS award 1136099 and Swedish KK-Foundation CERES Centre.

References

[1] W. Taha, "Domain-Specific Languages", Invited Paper, International Conference on Computer Engineering & Systems (ICCES'08.), Cairo, Egypt, 2008.

[2] B. Rumpe, M. Schindler, S. Völkel, I. Weisemöller, "Agile Development with Domain Specific Languages", In proceeding of: *Modelling Foundations and Applications - 7th European Conference*, ECMFA 2011, Birmingham, UK, 2011.

[3] A. Ahmed, S. Ahmad, N. Ehsan, E. Mirza, S.Z. Sarwar, "Agile software development: Impact on productivity and quality", *Management of Innovation and Technology (ICMIT)*, IEEE International Conference, Singapore, June 2010.

[4] L. Williams, "What agile teams think of agile principles", Communications of the ACM, vol 55, April 2012.

[5] S. Soomro, "New Achievements in Technology Education and Development", InTech, 2010.

[6] W. Taha, "A Gentle Introduction to Multi-Stage Programming", Domain-Specific Program Generation, LNCS, Vol 3016, 2004.

[7] S. Whitsitt and J. Sprinkle, "Model Based Development with the Skeleton Design Method", *Proceedings of the IEEE International Conference and Workshops on the Engineering of Computer Based Systems (ECBS)*, 2013.

[8] S. Deterding, M. Sicart, L. Nacke, K. O'Hara, and D. Dixon, "Gamification. Using game-design elements in non-gaming contexts", *In CHI '11 Extended Abstracts on Human Factors in Computing Systems (CHI EA '11)*. ACM, New York, NY, USA, 2011.

[9] R. Raymer, "Gamification: Using Game Mechanics to Enhance eLearning", *eLearn*, Vol. 2011, No. 9, September 2011.

[10] C. Ong, "Dynamic simulation of electric machinery: using MATLAB/SIMULINK", Prentice Hall PTR, Upper Saddle River, NJ, 1998.

[11] P. Fritzson, P. Aronsson, A. Pop, H. Lundvall, K. Nystrom, L. Saldamli, D. Broman, A. Sandholm,

"OpenModelica - A free open-source environment for system modeling, simulation, and teaching", *Proceedings of the IEEE International Symposium on Computer-Aided Control Systems Design*, 2006.
[12] S. Miller and J. Wendlandt, "Real-Time Simulation of Physical Systems Using Simscape", MATLAB News and Notes, 2010.
[13] T. Kurpick, C. Pinkernell, M. Look and B. Rumpe, "Modeling Cyber-physical Systems: Model-driven Specification of Energy Efficient Buildings", *Proceedings of the 2012 Modelling of the Physical World Workshop*, Innsbruck, Austria, 2012.
[14] K. Bauer, "A New Modelling Language for Cyber-physical Systems", Department of Computer Science, University of Kaiserslautern, Germany, January 2012.
[15] K. Schneider, "The synchronous programming language Quartz", Department of Computer Science, University of Kaiserslautern, Kaiserslautern, Germany, 2009.
[16] S. Neema, T. Bapty and J. Sztipanovits, "Multi-Model Language Suite for Cyber Physical Systems", Institute for Software Integrated Systems, Vanderbilt University, March 2013.
[17] P. Derler, E. Lee and A.L. Sangiovanni-Vincentelli, "Modeling Cyber-Physical Systems", *Proceedings of the IEEE*, vol 100, 2012.
[18] E.A. Lee, "Cyber Physical Systems: Design Challenges", *Proceedings of the 11th IEEE International Symposium on Object Oriented Real-Time Distributed Computing*, Orlando, Florida, USA, 2008.
[19] W. Taha, P. Brauner, R. Cartwright, V. Gaspes, A. Ames, and A. Chapoutot, "A Core Language for Executable Models of Cyber Physical Systems", Rice University, 2010.
[20] Y. Zhu, E. Westbrook, J. Inoue, A. Chapoutot, C. Salama, M. Peralta, T. Martin, W. Taha, M. O'Malley, R. Cartwright, A. Ames and R. Bhattacharya, "Mathematical equations as executable models of mechanical systems", *Proceedings of the 1st ACM/IEEE International Conference on Cyber-Physical Systems*, 2010.
[21] J. Bruneau, C. Consel, M. O'Malley, W. Taha, W. M. Hannourah, "Preliminary Results in Virtual Testing for Smart Buildings." Mobile and Ubiquitous Systems: Computing, Networking, and Services, Springer Berlin Heidelberg, 2012.
[22] Acumen web site. http://www.acumen-language.org, 2012. Last accessed 15 February 2014.
[23] W. Taha, P. Brauner, R. Cartwright, V. Gaspes, A. Ames, and A. Chapoutot, "A core language for executable models of cyber physical systems: work in progress report", ACM Special Interest Group on Embedded Systems, 2011.
[24] W. Taha and R. Philippsen, "Modeling basic aspects of cyber-physical systems", *Proceedings of the Workshop on Domain-Specific Languages and Models for Robotic Systems (DSLRob)*, 2012.
[25] W. Taha, R. Cartwright, R. Philippsen and Y. Zeng, "A First Course on Cyber Physical Systems", *Proceedings of the First Workshop on Cyber-Physical Systems Education (CPS-Ed)*, 2013.
[26] W. Taha, R. Cartwright, R. Philippsen, Y. Zeng, "Experiences with a First Course on Cyber-Physical Systems", Workshop on Embedded and Cyber-Physical Systems Education (WESE), 2013.
[27] Effective Modeling Group: Teaching, http://www.effective-modeling.org/p/teaching.html, February 2014. Last accessed 15 February 2014.

Authors

Principal Author: Dominykas Barisas holds a PhD degree in Software Engineering from Kaunas University of Technology. He is presently a lecturer at Kaunas University of Technology, having experience in software test automation, and is interested in cyber-physical systems and the development of domain-specific languages.

Co-author: Adam Duracz holds a MSc degree in Computer Science from Stockholm University. He is presently a PhD student at Halmstad University, working on a semantics for a hybrid systems modeling language, based on validated numerics.

Co-author: Walid Taha is a Full Professor of Computer Science at Halmstad University, and holds a Part-time Full Professor appointment at Rice University. He is interested in teaching and advancing the state of the art in Cyber-Physical Systems.