MetaEdit+ at the Age of 20

Steven Kelly, Kalle Lyytinen, Matti Rossi, and Juha Pekka Tolvanen

Abstract We review the initial vision underlying MetaEdit+, discuss its evolution over the last 20 years, and compare it to the state of the art today. We also note the rise of domain-specific modeling and the value that MetaEdit+ and similar tools have offered in advancing this field. We conclude with a discussion of theoretical and conceptual advances in this field that have taken place since the implementation of the tool, and a review of the future of method engineering.

1 Introduction

In the 1996 CAiSE conference we published a paper called "MetaEdit+: A Fully Configurable Multi-User and Multi-Tool CASE and CAME Environment" [8]. The paper described a state-of-the-art modeling and metamodeling environment that the ongoing project at the University of Jyväskylä had implemented. The main goals of the article were to explain the problems found with existing CASE and method engineering tools, state our vision for the MetaEdit+ environment, and describe the architecture and key principles in its design and implementation.

The MetaEdit+ tool was originally developed in a series of research projects from 1992 until 2001, building on the research behind the earlier, single user and single modeling language MetaEdit tool [22]. A spin-off company, MetaCase, was

S. Kelly (✉) • J.P. Tolvanen
MetaCase, Jyväskylä, Finland
e-mail: stevek@metacase.com; jpt@metacase.com

K. Lyytinen
Case Western Reserve University, Cleveland, USA
e-mail: kjl13@case.edu

M. Rossi
Aalto University, Espoo, Finland
e-mail: matti.rossi@aalto.fi

J. Bubenko et al. (eds.), *Seminal Contributions to Information Systems Engineering*,
DOI 10.1007/978-3-642-36926-1_10,

founded in 1991 and from 1995 research and development associated with the tool progressively shifted there and continues today.[1] The CAiSE article reflects our understanding of the necessary system functionality and its architecture in 1996, at which point most of the initial requirements elicited had been implemented to at least a working beta level.

By reflecting on the implementation and use of MetaEdit+ over the years we have gained a broad and deep appreciation of the challenges of method engineering and its changing nature as the software industry has evolved. In this paper we look at how MetaEdit+ has changed since 1996, and how it has impacted method engineering research and practice. We conclude with a summary of lessons learned and briefly discuss the future of method engineering and method engineering tools.

2 Past and Current Research Issues

In the mid-nineties CASE tools and heavyweight methods were seen as a panacea for most information systems development issues. We observed the need for more versatile tool support and integration and the ability to adapt tools and methods to specific situations. This approach was known as 'situational' method engineering, whereby standardized methods were adjusted for varying development tasks and situations [13]. The 1996 article was one of the first to articulate the challenges of situational method engineering and its tool support. That vision was explained and developed further in a series of theses [7, 11, 16, 20, 24, 29] and other publications [6, 21]. In our experience, history has been kind to that vision, and the solutions it presented are still valuable and relevant for software development.

Since CAiSE '96, large-scale methods for systems development have gradually gone out of fashion. At the same time CASE tools have become standardized work horses which can improve and support specific design and software development tasks. The commercial CASE tool market has also largely vanished whilst many powerful tools have been made open source (Eclipse) or offered for a very low fee (Visual Studio). Comprehensive and integrated methods and workbenches have been replaced with lightweight documentation and agile methods [1].

At the same time method engineering tools have found a new lease of life as language workbenches for Domain-Specific Modeling (DSM) [9]. This fits with the idea of evolutionary 'method prototyping', which was described and evaluated in Tolvanen's thesis [24]. OMG's MDA and Microsoft's Software Factories approach [4] have also driven the demand for flexible tools like MetaEdit+. The methods and tools for DSM have been honed in the OOPSLA DSM workshops[2] starting in 2001

[1] http://www.metacase.com

[2] http://www.dsmforum.org/DSMworkshops.html

[25], and the Language Workbench Challenge[3] from 2011. Several special issues have been published on DSM recently [3, 23, 27].

The move towards DSM use of MetaEdit+ emerged from its users, most notably Nokia's Jyrki Okkonen. As is often the case, research can create something interesting, but it takes industrial users to make it truly useful. DSM is however no panacea: most MetaEdit+ users have been concentrated in areas such as embedded systems (automotive, medical), consumer electronics, medical systems and telecommunications. Common themes have often included some kind of product line, a development space defined by use of an in-house platform or framework, or the configuration of complex systems from modular parts.

3 MetaEdit+ at Age 20

Since 1996 MetaEdit+ has evolved through industrial needs as well as innovation. Many of the features included in the 1996 environment have proved their worth, such as visual modeling, WYSIWYG symbol definition, incremental metamodel evolution, reporting and code generation facilities, and repository functions. In contrast, reverse engineering, hypertext, method rationale, and flexible queries and transformations have been used relatively little.

MetaEdit+ contains several browsers allowing flexible method composition from pre-defined parts. This was seen as a key feature of a method development environment at that time [28]. In practice the reuse of method components has rarely proven useful, except for large-grained units such as whole diagram types. The ability to reuse and reference individual elements has, however, proved key for integration between modeling languages. Similarly, method rationale has not been used, but hyperlinking generated code back to the model element that produced it has proved useful in practice.

MetaEdit+ was by no means a finished product in 1996 and many features have been added since then. Here we will just mention a few features we consider most important added between 1996 and the latest 5.0 release in 2012. The ability to represent complex graphical objects has been found to be vital for implementing many modeling languages, and for user acceptance of languages (See Fig. 1). The WYSIWYG Symbol Editor from 1996 has been extended significantly with features such as conditionality, dynamic templates, and SVG support. A new concept of *Port* was introduced, making GOPRR into GOPPRR. In 1996, MetaEdit+ was rather a monolithic, closed environment. Since then, support for a wide array of common image and document formats has been added. Model and metamodel information can be exported and imported as binary files or in an open XML format, and

[3] http://www.languageworkbenches.net

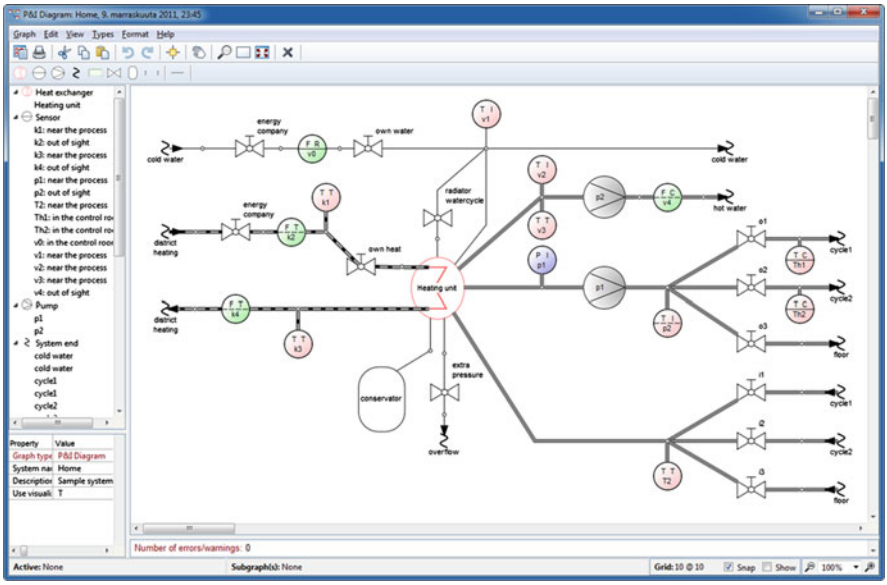

Fig. 1 Example model in MetaEdit+ 5.0 Diagram Editor

accessed and manipulated via an API. Open source plugins integrate MetaEdit+ into Eclipse and Visual Studio IDEs.

3.1 Research Impact

The MetaPHOR research group, from which MetaEdit+ was born, has produced over 10 PhD theses and ca. 50 research papers – most of them after the publication of the paper.[4] MetaEdit+ has been used as a reference tool in several tool comparisons (e.g. [10, 12]). The feature sets envisioned have also formed lists for future tools and MetaEdit+ has been used in many projects as a prototyping and development workbench in developing new software development methods [15, 17–19]. Today more than 50 universities are using MetaEdit+ to support both research and teaching. A 2008 IEEE Software article [5] identified MetaEdit+ as being at the highest level of abstraction for all software development tools, 15 years ahead of the curve. We would include the other early DSM tools such as Vanderbilt's GME [14] and Honeywell's DoME [2] in this category too.

[4] http://metaphor.it.jyu.fi/metapubs.html

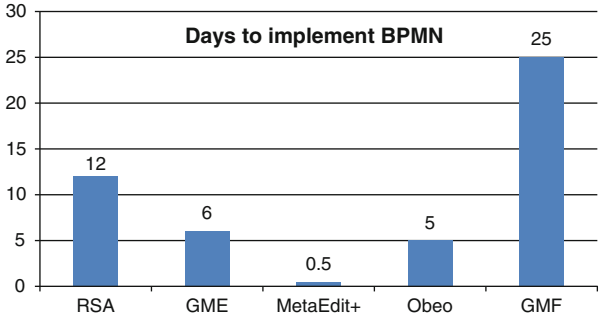

Fig. 2 Comparison of metamodeling time

3.2 Industry Reception and Practical Impact

The initial version of MetaEdit+ received recognition from BYTE magazine with a 'Best of CeBIT'95' finalist award, with later versions recognized in the Software Development Magazine Jolt awards (2004, 2005) and SDTimes top 100 (2007, 2008). MetaEdit+ has been used to develop a wide range of both software and hardware solutions. A prime example is Nokia feature phones, which have sold over a billion units running code automatically generated from a DSM language in MetaEdit+. Nokia estimated that applying DSM with MetaEdit+ increased productivity by a factor of ten [26]. Similar results have been achieved in fields as diverse as fish farming, insurance, railway systems, home automation, telecom services, and wearable sports computers. A recent article [12] by committers on the Eclipse Papyrus modeling tool compared MetaEdit+, IBM Rational Software Architect, Obeo Designer, GME and Eclipse GMF. The same language, BPMN, was modeled from scratch with each tool, recording the time taken (Fig. 2).

4 Summary

Advanced information systems engineering has changed technically significantly in the last 25 years. When we started work on metaCASE tools, there were no good graphics or persistency libraries available, so everything had to be developed from scratch. In 2013, creating tool support for modeling language engineering is technically easier, yet still conceptually challenging.

It can be argued that effective adoption and deployment of tools such as MetaEdit+ is no longer limited by the tool capabilities, but by the challenges of organizing the work through (meta)modeling and the intellectual challenges of developing original methods through DSM that can provide the necessary productivity payback. After the divergence to hundreds of languages in the 1980s, the convergence toward the dominance of UML left only a few creating their own

languages. There is currently a dearth of knowledge of the principles and benefits of high-level language creation and implementation in industry. Hopefully the recent growth of language development and uptake of DSM tools in universities can seed a new generation of language creators.

References

1. Cockburn A (2002) Agile Software development. Addison-Wesley
2. DoME Users Manual (1996). Honeywell Technology Center, Minneapolis
3. Gray J, Rossi M, Tolvanen J-P (2004) Domain-Specific Modeling with Visual Languages. Journal of Visual Languages & Computing 15 (3-4):207–330
4. Greenfield J, Short K (2004) Software Factories: Assembling Applications with Patterns, Models, Frameworks, and Tools. Wiley, Indianapolis
5. Helsen S, Ryman A, Spinellis D (2008) Where's My Jetpack? IEEE Software 25 (5):18–21
6. Jarke M, Pohl K, Weidenhaupt K, Lyytinen K, Marttiin P, Tolvanen J-P, Papazoglou M (1998) Meta Modeling: A Formal Basis for Interoperability and Adaptability. In: Krämer B, Papazoglou M (eds) Information Systems Interoperability. John Wiley Research Science Press, pp 229–263
7. Kelly S (1997) Towards a Comprehensive MetaCASE and CAME Environment: Conceptual, Architectural, Functional and Usability Advances in MetaEdit+. PhD Thesis, University of Jyväskylä, Jyväskylä
8. Kelly S, Lyytinen K, Rossi M (1996) MetaEdit+: A Fully Configurable Multi-User and Multi-Tool CASE and CAME Environment. In: Constapoulos P, Mylopoulos J, Vassiliou Y (eds) Advanced Information Systems Engineering, proceedings of the 8th International Conference CAISE'96. Lecture Notes in Computer Science. Springer-Verlag, Berlin, pp 1–21
9. Kelly S, Tolvanen J-P (2008) Domain-Specific Modeling: Enabling full code generation. Wiley-IEEE Computer Society Press
10. Kern H, Hummel A, Kühne S Towards a Comparative Analysis of Meta-Metamodels. In: Rossi M, Sprinkle J, Gray J, Tolvanen J-P (eds) Proceedings of the 11th Workshop on Domain-Specific Modeling, 2011.
11. Koskinen M (2000) Process metamodelling - Conceptual foundations and application. Dissertation, University of Jyväskylä
12. Kouhen El A, Dumoulin C, Gérard S, Boulet P (2012) Evaluation of Modeling Tools Adaptation.
13. Kumar K, Welke RJ (1992) Methodology Engineering: A Proposal for Situation Specific Methodology Construction. In: Kottermann WW, Senn JA (eds) Challenges and Strategies for Research in Systems Development. John Wiley & Sons, Washington
14. Ledeczi A, Maroti M, Bakay A, Karsai G, Garrett J, Thomason C, Nordstrom G, Sprinkle J, Volgyesi P The generic modeling environment. In: Workshop on Intelligent Signal Processing, Budapest, Hungary, 2001.
15. Leitner A, Preschern C, Kreiner C (2012) Effective development of automation systems through domain-specific modeling in a small enterprise context. Software & Systems Modeling
16. Marttiin P (1998) Customisable Process Modelling Support and Tools for Design Environment. Dissertation, University of Jyväskylä, Jyväskylä
17. Mewes K (2009) Domain-specific Modelling of Railway Control Systems with Integrated Verication and Validation Disseration
18. Preschern C, Leitner A, Kreiner C (2012) Domain-Specific Language Architecture for Automation Systems: An Industrial Case Study. Paper presented at the Workshop on Graphical Modeling Language Development

19. Qureshi T (2012) Enhancing Model-Based Development of Embedded Systems Dissertation. Disseration
20. Rossi M (1998) Advanced Computer Support for Method Engineering: Implementation of CAME Environment in MetaEdit+. Dissertation, University of Jyväskylä, Jyväskylä
21. Rossi M, Ramesh B, Lyytinen K, Tolvanen J-P (2004) Managing Evolutionary Method Engineering by Method Rationale. Journal of AIS 5 (9 article 12)
22. Smolander K, Lyytinen K, Tahvanainen V-P, Marttiin P (1991) MetaEdit-A Flexible Graphical Environment for Methodology Modelling. In: Andersen R, J. A. Bubenko jr., Solvberg A (eds) Advanced Information Systems Engineering, Proceedings of the Third International Conference CAiSE'91. Lecture Notes in Computer Science. Springer-Verlag, Berlin, pp 168–193
23. Sprinkle J, Mernik M, Tolvanen J-P, Spinellis D (2009) Special issue on Domain-Specific Modeling editorial. IEEE Software 26 (4)
24. Tolvanen J-P (1998) Incremental Method Engineering with Modeling Tools: Theoretical Principles and Empirical Evidence. Dissertation, University of Jyväskylä
25. Tolvanen J-P, Gray J, Lyytinen K, Kelly S Proceedings of 1st OOPSLA Workshop on Domain-Specific Visual Languages. In: Tolvanen J-P, Gray J, Lyytinen K, Kelly S (eds) Proceedings of 1st OOPSLA Workshop on Domain-Specific Visual Languages, 2001. Jyväskylä University Printing House
26. Tolvanen J-P, Kelly S (2000) Benefits of MetaCASE: Nokia Mobile Phones Case Study. MetaCase Consulting plc. http://www.metacase.com/papers/MetaEdit_in_Nokia.pdf. Accessed 1/7 2004
27. Tolvanen J-P, Rossi M, Gray J (2013) Theme Issue on Domain-Specific Modeling in Theory and Applications editorial. Journal of Software and Systems Modeling to appear
28. Zhang Z Defining components in a MetaCASE environment. In: CAiSE'00, Stockholm, Sweden, 2000. Springer-Verlag
29. Zhang Z (2004) Model component reuse : conceptual foundations and application in the metamodeling-based systems analysis and design environment. Dissertation, University of Jyväskylä, Jyväskylä