Improving the Development Tool Chain in the Context of Petri Net-Based Software Development

Tobias Betz, Lawrence Cabac, and Matthias Güttler

University of Hamburg
Faculty of Mathematics, Informatics and Natural Sciences
Department of Informatics
{2betz,cabac,3guettle}@informatik.uni-hamburg.de
http://www.informatik.uni-hamburg.de/TGI

Abstract. Modern, collaborative software development projects are highly supported by a variety of tools. Aside from the pure code development that is nowadays well supported by integrated development environments (IDEs) such as Eclipse, also other activities receive increasing attention in the matter of tool support. Recent improvements in tool support for source code management (SCM), process management and documentation management are driven by the changing demands of increasing distribution and agility of development projects. Moreover, the integration of tool support systems into integrated project management environments (IPMEs) increase the usefulness of each emerging tool support, especially for agile approaches. The Petri net-based agent-oriented software engineering (Paose) is highly influenced by agile methods and combines the agile concepts with aspects from model driven development. We have achieved increased efficiency by the adoption of IPMEs into the Paose development approach. However, the introduced tool set integration lacks in support for the model driven part of Paose, i.e. the tool integration for graphical (model-based) source code is missing. In this work we present an approach for an appropriate tool support of agile methods and model driven development. As a prototypical implementation of the concepts we present a Web service-based framework and plugin-based extension for Redmine as representative of open source IPMEs, which is currently in use in the context of the Paose approach.

Keywords: Renew, integrated project management, Paose, model-driven development, Redmine

1 Introduction

The history of software development shows a continuous increase of complexity in several aspects of the development process [1, p.10]. Furthermore, the increasing relevance of software in general and the higher demand in quality and speed contribute to the complexity of procedure. In comparison to conventional software

development, today's modern development requires more flexible approaches to manage the given circumstances. In fact, project managers have to deal with distributed teams and software systems, concurrently working developers and fast and sporadic changes of demands.

Therefore we need even more adequate tools in order to keep the development process under control and to handle the different requirements in a satisfyingly and efficient manner. In the usual software development process, there are a lot of well-established tools in use, such as integrated development environments (IDEs) to support technical aspects of programming and project management tools, assisting developers and mainly superintendents in organizational matters. The centralized or distributed storage of source code is also well supported by various source code management (SCM) tools such as Subversion[1] or Git[2].

In contrast to the classical software development, the agent-oriented software development faces developers and project managers with some new aspects of tasks and therefore with some new fields to support, possibly even with an extended tool chain. One main reason for this is the self-organized and autonomous character of agents, which we also adopted into the organization of the development team [4]. This adoption goes well with the agile methods, which highly influenced the evolution of the Paose approach. Another reason is the model driven development based on Petri nets. In the Paose approach, we do not only operate with plain text files as source code, we also work with a graphical representation of Petri nets as source code to accomplish our targets of development. In contrast to the conventional model-driven development, in which the models are used to generate the executable source code, the Paose approach directly uses the models (Petri nets) as source code.

With this work we present an approach to improve the developers tool chain in context of the Petri net-based agent-oriented software development. For that reason, we advance the development process through adjusted and new tools, which fill the gap of partially unsupported tasks. Additionally, our focus lies on the adequate integration of these tools in our existent tool chain to advance the integrated project management environment (IPME). In Section 2 we investigate the developers' tasks, especially in context of the agent-oriented and agile procedure of development. Furthermore, we point out the deficits of the existent tool chain and additionally work out some possible supporting measures. We decided to primarily improve the topic of source code management within the Petri net-based development through an extended tool set. Section 3 describes these tools with regard to the architecture, realization and integration. We also present our prototypical implementation, which allows to include visual representations of any diagram type / net, available in the Renew tool set (see `http://www.renew.de`), in an IPME. This includes the simple displaying of diagrams / nets as well as the presentation of visual differences between versions available in the SCM.

[1] Apache Subversion, Enterprise-class centralized version control for the masses (http://subversion.apache.org)

[2] Git, distributed version control system (http://git-scm.com)

In the final conclusion we summarize our results of investigation and discuss the achieved benefits through our enhancement in the development tool chain.

2 Developer Tasks in Context of Paose

The process of software development consists of many different activities and phases we traverse during the specification, implementation, verification and deployment of software artifacts [13]. Figure 1 shows the various tasks a developer has to deal with. Besides the pure writing of source code, there are more tasks to focus on. Primarily, the definition of development tasks and their timing is essential concerning the planning and definition of pending work. These activities are supported by ticketing systems or bug trackers, which are widely applied in common tool chains of software development projects. Not only the current implementation phase is supported by tickets, it is also part of the software artifact specification within the meaning of agile methods.

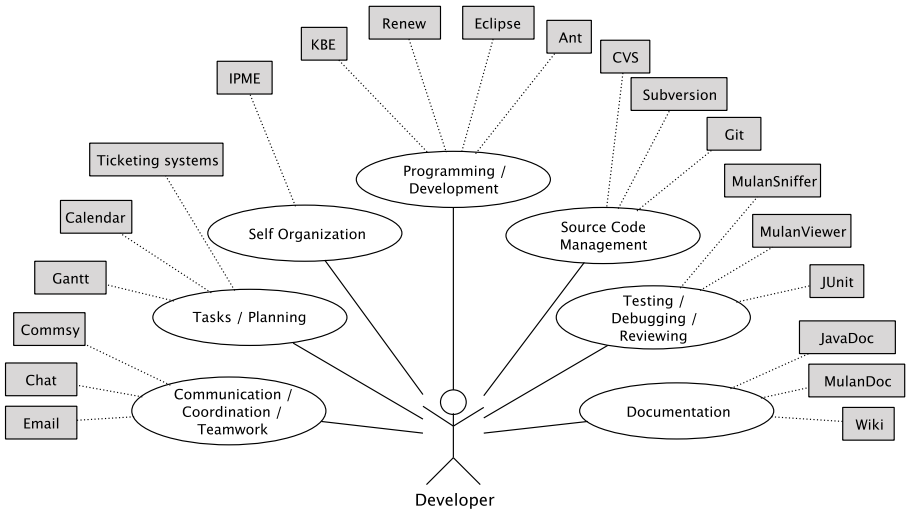

Figure 1. The developer's tasks and the supporting tools.

While communication is a prerequisite for the proper planning and coordination inside the development team[3], this activity gets more relevant within distributed teams and concurrently working developers. The terms self-organization and responsibility are outstanding in agent-oriented development [4,6], as well as in applied agile approaches [3]. Regular teamwork is already well supported

[3] In this context, we refer to development teams as individuals, who take part in the process of development and subsumes programmers, managers and contributors to the project.

by various communication platforms, but often less efficient than desired, especially in the context of agent-oriented software development[4]. Proper change notification in distributed planning is a complex task, regarding what Petrie et al. [11] investigated in this matter, and still not satisfactorily solved for modern, collaborative software development environments. In the context of agent oriented software development, especially the autonomous and self-organized aspect is predominant and consequently communication and coordination within the team becomes more important.

Furthermore, the source code management plays a significant role sharing resources within the development team. Regarding textually represented source code, a wide variety of tools exists to support the developer [9]. We already mentioned SCM tools like Subversion and Git. For graphically represented source code or models, on which the Petri net-based agent-oriented software development is based on, however, there is a lack of supporting tools to manage and visualize source code or changes made to it [9]. Finally one of the most useful features of modern SCM tools is the possibility to reveal differences between arbitrary revisions (versions) of the source code. This insufficiency within the Petri net-based development is a problem in order to accomplish the required information exchange and also the change notification between the developers of a (distributed) team, as mentioned before. Regarding Schipper et al. [12], a major concern is the depiction of changes in graphical models in a graphical way. We will present an appropriate solution for this missing feature in the following section.

Documentation is another task a developer is faced with. In agile approaches this task is less expensive, because running software has a higher value as comprehensive documentation if we follow the agile manifest [2]. However, apart from this, documentation is always a task to manage. Some IPMEs already support integrated documentation in the form of wikis for example. But also the documentation inside the source code is a part of the entire documentation and should be supported in some form by the integrated documentation system or the IPMEs as well. An IPME with integrated SCM and possibility to write and permanently save additional documentation fulfills this task sufficiently.

Finally, developers make use of a lot of tools. We aim at the integration of all used tools in the existent and usual tool chain and further connect these tools together. This means that developers are able to link the various contents and results of the tools' usages. More precisely, we target the linking of source code with task definitions, documentation and discussed topics relevant to the concrete realization. Additionally, the exchange of information between the tools should be realized as an automatic process embedded into the usual workflow. The result of this is an integrated project management environment, being able to handle many tasks belonging to different phases and areas of the complete development process and finally to make this information available through a single interface.

[4] Note that in Paose the agent-oriented paradigm is also applied to the development team and to the development process.

The free available open source project management tools Redmine[5] and Trac[6] support many of these needs out of the box and also provide a notification system via e-mail transparently embedded in the usual workflow. One missing feature, the possibility to view and compare graphical source code in a visual manner inside the IPME, is realized through our extension of the tool chain and will be presented in the following section.

3 Web Service-Based Tool Integration

In the previous section, we pointed out the requirement of a tool to visualize and compare graphical source code, particularly in context of Petri net-based development. This section deals with the architecture, implementation and integration of the outlined draft for such a tool, that fits perfectly into the distributed nature of agent-oriented software development, our existing tool chain and development workflow.

3.1 Architecture of the Tool Set

Due to the plugin-based design of the most project management tools, in our case Redmine, it is possible to improve the functionality of the IPME with respect to the integration of Petri nets and also other models such as sequence diagrams, class diagrams, etc. For this purpose, we provide an extension, which enables the IPME to (1) render net files in a graphical manner instead of showing the corresponding text representation and (2) compare two revisions of the same net file by highlighting their differences. The implementation of this extension is divided into two functional components. The first is designed as a Web service, which offers the capability to compute images of submitted nets that depict the requested functionality of the extension (detailed description in Section 3.3). The second component is located in the IPME as a plugin called Redmine Renew Plugin and integrates the functionality offered by the Web service into the web-based user interface (as described in Section 3.5). This is realized by extending the existing code view mechanisms so that net files supported by Renew (*.rnw, *.aip, *.draw, *.pnml, etc.) will be forwarded and handled by the corresponding Web service.

Each component of the tool chain can be assembled and hosted in a completely distributed environment, as depicted in Figure 2, or centralized on one host. The IPME is hosted on a web server that offers developers, managers and designers an easy access using a simple Web browser. Source code repositories are located on file servers and will be integrated into the environment with the help of the build-in source code management (SCM) connectors. The major workload of the net integration is done by the export server, which hosts the two Web services (Image Net Export and Image Net Diff).

[5] Redmine, A flexible project management web application (http://www.redmine.org)
[6] Trac, Integrated SCM & Project Management (http://trac.edgewall.org)

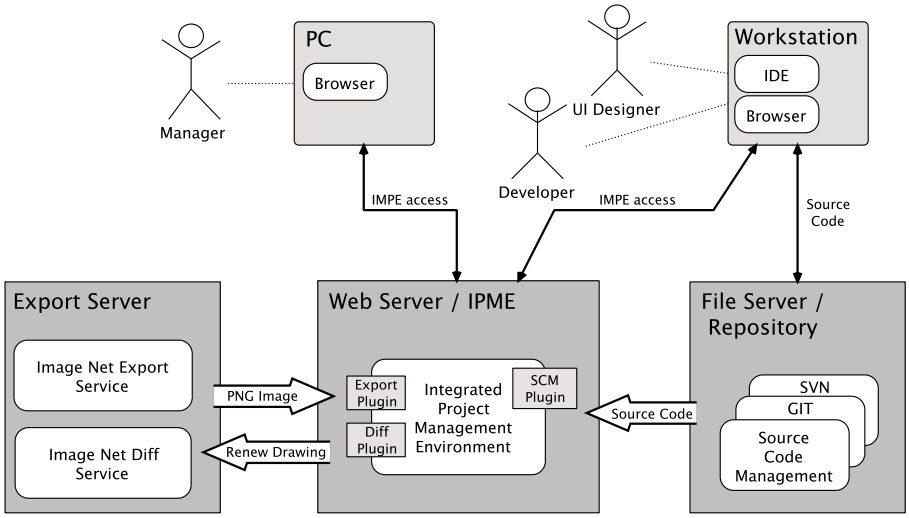

Figure 2. Web Service-based extension of the distributed tool chain environment.

3.2 Export and ImageNetDiff

Conversion of diagram / net representation into image representations is provided by two plugins available for Renew. The Export plugin is part of the distribution of Renew and makes use of the powerful Freehep libraries (see `http://www.freehep.org/`). It was first included in version 2.1 of Renew. The Export agent makes use of the PNG (Portable Network Graphics) export format, which is displayed directly in a Web browser.

The ImageNetDiff plugin for Renew is available as on optional plugin and allows producing a visual diff of two diagrams or nets. As the name suggests, the visual diff is represented as an image (compare with [7]) and is in fact produced through a comparison of two images, which are the result of an export operation. The functionality is available in Renew from the GUI or as command line argument, i.e. comparing two opened nets (or diagrams) as well as using the functionality in scripts is possible. Additionally, the comparison of a checked-out copy of a net against the revision base in a Subversion repository is directly supported. With the integration of the ImageNetDiff functionality in the IPME the comparison of arbitrary version in a repository is also easily possible. Instead of executing a diff on the textual representation and mapping the results back to the Petri net or diagram we make heavy use of the graphical (analogue) representation of the diagram. Thus the diff procedure in our implementation is not a semantic (or syntactic) diff but a purely visual one. On the one hand, this approach holds some disadvantage. Direct editing of nets on the basis of the diff representation is not possible and also our tool set does not support automatic merges. On the other hand, we can also derive some advantages from this approach. The implementation is very simple. In many relevant cases this

approach shows the desired results. The results integrate directly into the Web-based applications. Note also, that the diff images are only transitional objects during the development of models / source code. For the means of comparison, they are only useful at the moment of creation. Once the desired information is found, e.g. the error is located, the diff object is deleted.

3.3 Web Service Implementation

Both services (Image Net Export Service and Image Net Diff Service) used in the Redmine Renew Plugin are realized as RESTful Web services [10]. This approach offers a high interoperability and simplifies the integration of services in distributed environments. The current implementation of the Export Server (see Figure 2) is based on Mulan/Capa [8]: a Petri net-based multi-agent framework. Thus, the functionality of both services is provided by an agent (called Export agent), which is able to offer its services as Web services. This is possible with the help of the Mulan WebGateway agent, which is acting between the export agent and the invoking Web client, in our case the Redmine Renew Plugin. As depicted in Figure 3 the WebGateway agent has two communication interfaces and is able to communicate on the right hand side with agents over FIPA-ACL[7] messages. On the left hand side the interface is implemented by a Jetty[8] Web server and is able to handle HTTP and WebSocket[9] connections. Besides the capability to register and manage agent Web services, the major purpose of the WebGateway is to translate incoming Web client messages to FIPA-ACL messages and vice versa. During agent service registration at the WebGateway, each service can submit a service representation in terms of ordinary JSP[10] Web sites. These sites serve as web-based service invocation applications and are available by opening the associated service URL in a Web browser.

To fulfill the desired behavior of the export agent services (Image Net Export Service and Image Net Diff Service), the implementation makes use of the Renew Export and ImageNetDiff plugins (mentioned in Section 3.2), which are available due to the fact that the Mulan multi-agent framework uses Renew as a runtime engine.

3.4 Redmine Renew Plugin Implementation

The current prototype of the Redmine Renew Plugin implements two different features, export and diff, which both work with source code files from the connected repository, as described in Section 3.1.

The visualization of a single net file is a relatively simple task and thus fairly simple to implement. The plugin has to fetch the net file from the repository and post the content of this file to the Web service to get a visual representation

[7] FIPA ACL: http://fipa.org/specs/fipa00061/SC00061G.html
[8] Jetty Web server: http://www.eclipse.org/jetty/
[9] W3C WebSocket API: http://dev.w3.org/html5/websockets/
[10] Java Server Page (JSP): http://www.oracle.com/technetwork/java/javaee/jsp/

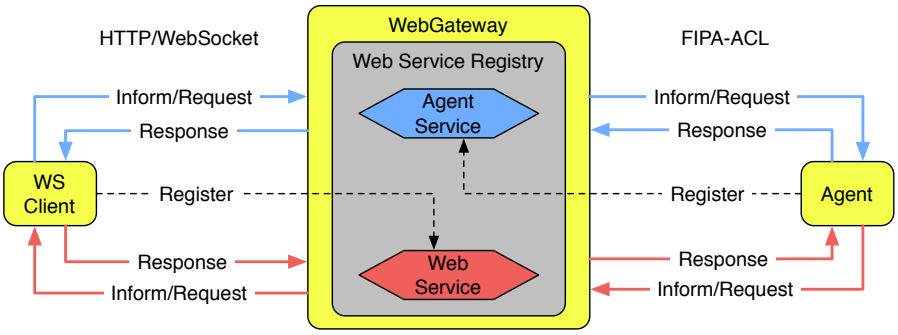

Figure 3. WebGateway Architecture.

of the net as result. This result, the received PNG image, will be shown to the user, embedded inside the repository view of Redmine (compare with Figure 4, in which a diff image is shown, which is explained in Section 3.5). The comparison of two revisions of the same net file from the repository does not make a big effort, too. In this case, the plugin has to fetch the two different revisions of the net file and post its contents to the Web service to receive the image with the highlighted differences.

However, the case of comparing two complete changesets[11] is a complex task with regard to the technical aspect. This is firstly, because a changeset might contain graphical and textual source code changes, therefore the plugin has to decide for every file separately, if it will use the image net diff algorithm or the build-in textual one[12]. Secondly, the type of change can be one of the following for every single file in the changeset: addition, modification, deletion, duplication, moving or renaming. As a consequence, there are a lot of possible cases to take into account e.g. if we add a net file in one changeset, remove this file in the next one and after that we add a file with the same name but different contents again. Then, the complete history of changes made to this file name has to receive attention in order to decide which revisions should be compared. Another complex sequence of operations arises, when a file becomes renamed or moved. In this case the plugin has to fetch the previously named file and the current one in order to compare the changes, what results in a supplementary request to the repository.

[11] A changeset contains all changes, which were made in an atomic commit and contains multiple files in the majority of cases.

[12] The repository itself (i.e. Subversion) does not make any difference between file types and returns a single unified diff file for all changes made in a changeset. Therefore the Redmine Renew Plugin has to split this unified diff into separate parts, whereby every part contains the changes made to a single file.

3.5 Integration into the Existing Tool Chain

In the Paose approach we use project management tools that are already well-established in agile development approaches. By their usage, we directly apply the advantages of these tools to the agent-oriented development and benefit from the tight linking of source code with developer tasks and documentation, as requested within Section 2. Beyond that, we extend these tools with plugins according to the needs of the Petri net-based agent-oriented software development, which result from the model-driven nature of the Petri net-based development, as described in Section 3.1.

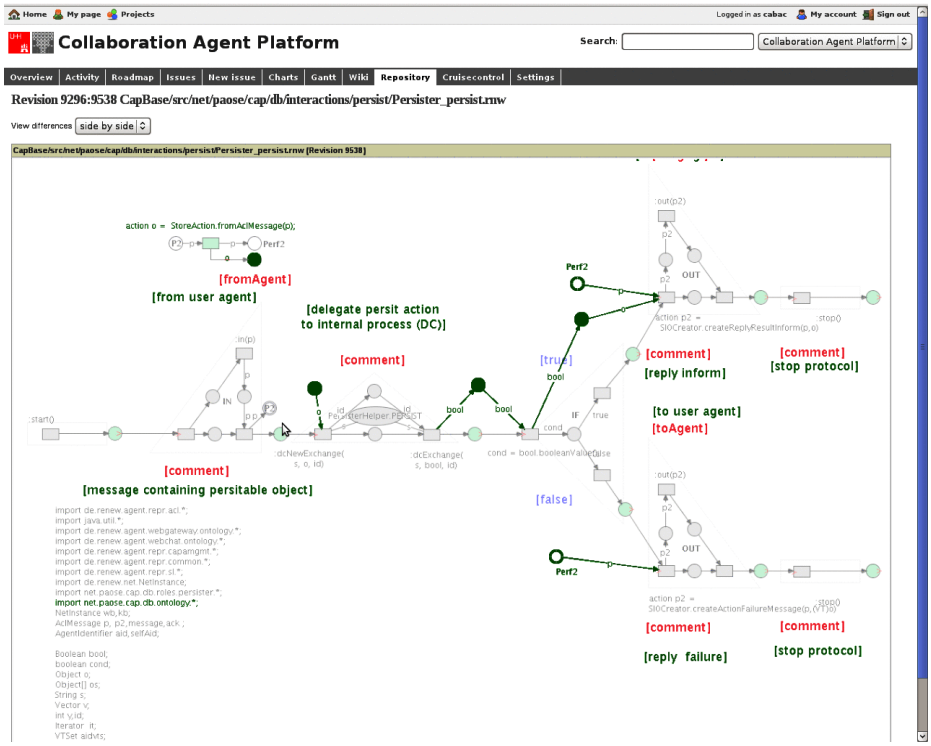

Figure 4. Redmine ImageNetDiff plugin in use.

Figure 4 shows a screenshot of the web-based IPME Redmine. The image shows the repository view extended by the Redmine Renew Plugin in diff mode. Inside the content area in the center, it shows the differences of two revisions (i.e. revisions 9296 and 9538) of a Petri net from the connected Subversion repository. The colored areas describe changed parts of the Petri net, whereby

green[13] signifies added parts and red deleted parts, while the light-gray areas signify unchanged parts. The plugin integrates into the IMPE in a transparent way, without the need of changing anything in the general workflow. Additionally, the presentation of both, textual and graphical source code, is embedded into our IMPE homogeneously. The just described screenshot shows only one possible use of the plugin. There is also the ability to compare other graphical resources, such as use case diagrams or agent interaction protocols, which both were introduced to the Paose approach [5].

A demonstration of the new tool integrated into the IMPE Redmine is accessible under `https://paose.informatik.uni-hamburg.de/redmine/`. The connected Subversion repository contains some examples, which were developed in the context of the P*AOSE student project at the University of Hamburg. The presented IPME allows public users to access the source code, to visualize and to compare different versions.[14]

In Section 2 we described the developer tasks and discussed the need for tool support for the comparison of graphical source code (i.e. in our case Petri nets) and other diagrams in a visual way. The possibilities offered by our presented extensions especially support the developers in communication aspects and the exchange of information. Furthermore, reviewing and refactoring of code, for instance in order to discover and eliminate some bugs, are simplified, because the simple access to a visualization of the graphical source code is integrated into the applied IPME. Besides that, project leaders and other participants can inform themselves straightforwardly about the progress of development without the need of an IDE and thus also derive benefits from the integrated tool set.

A typical use case for the net diff inside the IPME is a session of finding the differences during a debugging session. However, one very useful simple use case is examining, whether changes were made in a model. For instance marginal changes in the layout (e.g. changing the z-order of elements) can result in repository check-ins. Also if changes from for example version A to version B are reverted in version C the control of absence of differences between version A and C and the easy access to this information is of great advantage.

4 Conclusion

In this work we investigated the development process in Petri net-based agent-oriented software development concerning the improvements of tool support. After outlining developer tasks and gaps in tool support, we introduced an extension to the tool set to accommodate the source code management of graphically represented source code, embedded into the widely used open source project management tool Redmine.

[13] A black and white print shows only light-gray parts and black net elements and inscriptions. With the exception of the comments (in square brackets) only additions have been made in the presented diff.

[14] Additionally the Web service-based functionality of diagram export and diagram image diff can be accessed manually via an included Web interface.

With the introduction of our extended tool chain in the context of the P*AOSE projects at the University of Hamburg, which follow the Paose approach, we improved various aspects of the developers' tasks. First of all, of course we supported the possibility to observe graphical source code and compare it concerning its different versions. Supplementary, we improved the speed of development through this feature, in respect of the possibility to visually compare model-based source code (i.e. in our case Petri nets) and other diagrams directly and effortlessly in an integrated environment for the project management and planning (IPME), which is closely linked to the SCM. Furthermore the presented tool integration supports the communication of the developers. Information exchange through source code changes between the developers gets easier than before and performs thereby the requirements delineated in Section 2. Finally, we come to the conclusion that our presented extended tool chain is one substantial step to improve the development of Petri net-based agent-oriented software. The gap between text-based and model-based source code – as well as other design models – is thus further closed in regard to the handling of these documents during development, reviewing, documentation and refactoring. The presented approach could easily be adapted to support other development approaches that make use of graphical representations (e.g. UML diagrams) during development.

References

1. Helmut Balzert. *Lehrbuch der Softwaretechnik: Basiskonzepte und Requirements Engineering*. Spektrum Akademischer Verlag, 3. aufl. edition, 2009.
2. Kent Beck, Mike Beedle, Arie van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, Jon Kern, Brian Marick, Robert C. Martin, Steve Mellor, Ken Schwaber, Jeff Sutherland, and Dave Thomas. The agile manifesto. http://agilemanifesto.org.
3. W.G. Bleek and H. Wolf. *Agile Softwareentwicklung: Werte, Konzepte und Methoden*. dpunkt, Heidelberg, 2008.
4. Lawrence Cabac. Multi-agent system: A guiding metaphor for the organization of software development projects. In Paolo Petta, editor, *Proceedings of the Fifth German Conference on Multiagent System Technologies*, volume 4687 of *Lecture Notes in Computer Science*, pages 1–12, Leipzig, Germany, 2007. Springer-Verlag.
5. Lawrence Cabac. *Modeling Petri Net-Based Multi-Agent Applications*, volume 5 of *Agent Technology – Theory and Applications*. Logos Verlag, Berlin, 2010.
6. Lawrence Cabac, Till Dörges, Michael Duvigneau, Christine Reese, and Matthias Wester-Ebbinghaus. Application development with Mulan. In Daniel Moldt, Fabrice Kordon, Kees van Hee, José-Manuel Colom, and Rémi Bastide, editors, *Proceedings of the International Workshop on Petri Nets and Software Engineering (PNSE'07)*, pages 145–159, Siedlce, Poland, June 2007. Akademia Podlaska.
7. Lawrence Cabac and Jan Schlüter. ImageNetDiff: A visual aid to support the discovery of differences in Petri nets. In *15. Workshop Algorithmen und Werkzeuge für Petrinetze, AWPN'08*, volume 380 of *CEUR Workshop Proceedings*, pages 93–98. Universität Rostock, September 2008.
8. Michael Duvigneau, Daniel Moldt, and Heiko Rölke. Concurrent architecture for a multi-agent platform. In Fausto Giunchiglia, James Odell, and Gerhard

Weiß, editors, *Agent-Oriented Software Engineering. 3rd International Workshop, AOSE 2002, Bologna. Proceedings*, pages 147–159. ACM Press, July 2002.
9. Akhil Mehra, John Grundy, and John Hosking. A generic approach to supporting diagram differencing and merging for collaborative design. In *Proceedings of the 20th IEEE/ACM international Conference on Automated software engineering*, ASE '05, pages 204–213, New York, NY, USA, 2005. ACM.
10. Cesare Pautasso, Olaf Zimmermann, and Frank Leymann. Restful web services vs. "big" web services: making the right architectural decision. In *Proceeding of the 17th international conference on World Wide Web*, WWW '08, pages 805–814, New York, NY, USA, 2008. ACM.
11. C. Petrie, S. Goldmann, and A. Raquet. Agent-based project management. *Artificial intelligence today*, pages 339–363, 1999.
12. Arne Schipper, Hauke Fuhrmann, and Reinhard von Hanxleden. Visual comparison of graphical models. *Engineering of Complex Computer Systems, IEEE International Conference on*, 0:335–340, 2009.
13. Trung Hung VO. Software development process. Available at: `http://cnx.org/content/m14619/1.2/`, July 2007.