UDC 004.855.5

Vladimir L. Pavlov, Nikita I. Boyko, Alexander V. Babich

FIRST EXPERIENCE OF USING INTSPEI P-MODELING FRAMEWORK IN SOFTWARE DEVELOPMENT PROJECTS

INTSPEI P-Modeling Framework is a set of principles, methods and tools aimed at extending existing SDLCs to make software development teams more efficient and productive, hence to shorten the overall amount of development efforts spent on particular projects. The framework is based on two key methods: Reverse Semantic Traceability and Speechless Modeling, which were created as a result of the authors' experiments. This paper describes the first experience of using these methods in non-experimental real-life business environment.

Introduction: History of INTSPEI P-Modeling Framework

In 2001 Vladimir L. Pavlov developed a training program called "The Babel Experiment". The essence of the method was that a team of students was supposed to design a software system. They had a few hours to complete the task. During this timeframe verbal and written communication was forbidden, and the UML [1, 2] was the only allowed language. This training was a kind of experiment for students – they were to discover whether UML is "a real language" that is suitable and beneficial for a project team. The Babel Experiment was always successful – every time students were able to find the common language and generate the common ideas by UML communication, which led them to successful development of the proposed system model.

This experiment being executed as training helps to achieve the following major goals:

- make students go through communication problems that are typical for large software development projects;
- provide students with the successful experience of applying UML to overcome these problems.

The training was delivered dozens of times in both academic and corporate environments and generated absolutely positive feedback from students and customers [3, 4].

Once (accidentally) during this training there were two independent teams working on the same task. One team was limited to using only the UML language (and pantomime) in their communication. The other was allowed to use speech in addition to the UML. The first team (which was not allowed to use speech) coped with a task more successfully than the other team. Their diagrams were more sound, detailed, readable, elaborated and elegant. After that the first author conducted several experiments aimed at comparing efficiency of traditional and "speechless" modeling sessions. In these experiments speechless teams were always at least as efficient as traditional teams (those who were allowed to speak), in most cases speechless modeling teams outperformed traditional teams. Participants of these experiment concluded that the main reasons for such efficiency of speechless modeling sessions were:

- requirement not to use regular language stimulates creativity of designers as well as makes them stay focused on their task;
- work in speechless mode forces designers to explicitly uncover all assumptions at the very early phases of design;
- designers stop treating UML as a "write-only" language aimed at creating documentation that nobody ever reads – instead they start to care about readability of their models.

After these experiments several software development companies started to incorporate speechless modeling sessions into their SDLCs, and reported that it had positive impact.

Meanwhile Vladimir L. Pavlov continued his experiments aimed at comparing UML and "traditional" human languages.

Let us assume, one has a text written in English, and a team of translators creates a Russian version of this text. The Russian text is then given to another team, and they translate it

back to English. Original and restored English texts may have different number of words (or even sentences), but semantically they will be very close. Will the same happen if a team of designers creates a UML model based on a textual description of some domain, and then another team of designers restores the original textual description from the UML model? To see what are UML capabilities in this area, the first author had conducted several experiments where a team of professional designers ”translated” some texts from English to UML, then another team “translated” it back from UML to English, then original and restored texts were compared. These experiments have shown that UML is quite similar to traditional languages – original and restored technical texts were semantically close.

However, the main result of these experiments was not about UML and modeling – it was about testing and quality assurance. A real-life software development process is a sequence of translations: analysts translate requirements from natural regular human language to a business model, architects translate it to design, developers translate it to source code in some programming language, and finally compiler translates it to executable. On every step of this sequence there are usually some errors/misinterpretations, however, to find these errors people typically test the very final result – executable. But it is easy to test every step in this sequence of translations – just give intermediate deliverables to an independent testing team to translate back, and then compare original and restored versions of artifacts which serve as inputs to the current step. If no information is lost or misinterpreted – then it is OK to proceed to the next step, otherwise it is required to make some corrections to the deliverables of the current step (or, may be, to the inputs of the current step). The author called this simple approach “Reverse Semantic Traceability”.

The most expensive errors are those which are created during the design process. It is important to be able to test models created as an outcome of the design as early as possible. So the architects that participated in Vladimir’s experiments started to implement Reverse Semantic Traceability in their companies. Their feedback is fantastic, because they are able to fix bugs early; the overall duration of software development cycle becomes up to 35% shorter, and finally produced software has better quality.

The two concepts described above - Speechless Modeling and Reverse Semantic Traceability - have underlain the INTSPEI P-Modeling Framework – a set of principles, methods and tools aimed at extending existing SDLCs (such as RUP – [5] or MSF – [6]) to make software development teams more efficient and productive.

To reach synergy, the authors developed a P-Modeling Session - a one-day design event that integrates both techniques.

In this article we will provide a detailed discussion about Speechless Modeling, Reverse Semantic Traceability and P-Modeling Sessions.

Reverse Semantic Traceability

The sequence of major phases in any software development project can be treated as a sequence of “translations” from one language to another. As a result of every translation the translated “text” becomes more formal and more detailed. At the very beginning of the software development lifecycle we deal with very not-concrete, high-level, human-language domain description with many questions not answered (or even not asked). At the end we have a very formal and very detailed text in some programming language, which can easily be “understood” by a computer. The more complex problem we solve, the more “intermediate” languages we need to be able to operate on various abstraction levels.

Of course, every step adds some errors/misinterpretations which accumulate during the product development, and at the end customers get something very different from what they initially wanted/meant.

This problem becomes even more important in today’s international business environment, where multinational companies face additional challenges created by the need to overcome cross-cultural, cross-linguistic, time-difference, geographical-distance and

other communication barriers within globally distributed development teams.

To minimize this effect most software development teams include a special role (or several different roles) aimed at making sure that every “translation” is done properly and, at the end, the project results with what originally was desired by a customer.

This role is usually called tester, quality engineer, QA specialist, etc. Over time

Fig. 1. Each step of product development brings some misinterpretation when one artifact evolves to another. As the result of consequential accumulation of such information distortion, the final product is far away from customer’s expectation.

engineers developed plenty of various methods and techniques for quality control [7, 8]. The most testing is usually done when programmers have already created an executable – on the late phases of software development lifecycle. Unfortunately, the most expensive (the most important) errors are created during the early phases of the development process, when there is no executable code – only diagrams and models [9].

Figure 4). Although it may look like adding extra efforts, experience of early adopters shows that the overall amount of development efforts is reduced because issues are discovered and fixed without delays, so they do not accumulate and do not cascade to subsequent phases of the development cycle.

The key word in the name of this method is “Semantic”, because the original and restored versions of a “text” are to be compared

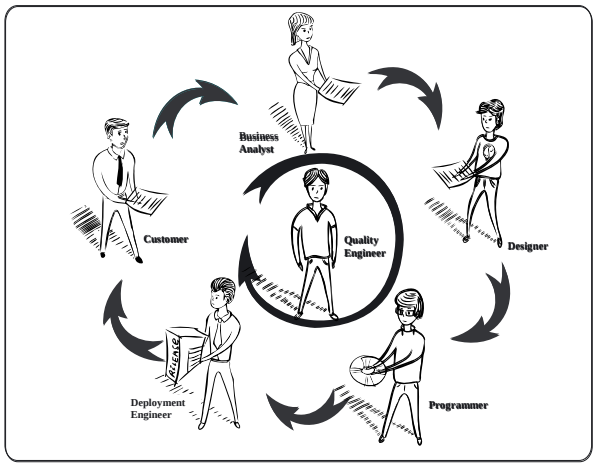

Fig. 2. Quality assurance decreases the information distortion.

Reverse Semantic Traceability is a quality control method that allows testing inputs/outputs of every “translation” step. Before proceeding to the next phase current artifacts are “reverse engineered”, and restored “text” is compared to the original. If there is a difference between these two “texts” – the tested artifacts are corrected to eliminate the problem. As a result every step is confirmed by stepping back and making sure that the development still stays on the correct track (see

semantically, with a focus on the “meaning” of this text, not on particular “words” used in it.

For each project step where we implement Reverse Semantic Traceability, it is important to motivate project participants to properly balance their efforts between replicating information from input artifacts vs. adding design decisions and technical details to output artifacts. INTSPEI P-Modeling Framework provides practical guidelines and recommendations on how to do it.

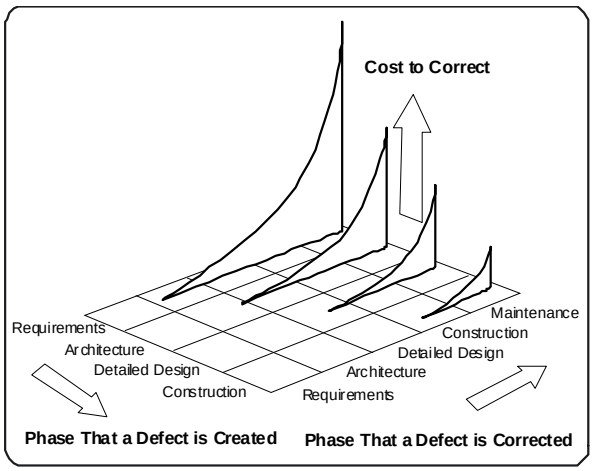

Fig. 3. The earlier an error is created, the later it revealed, the more expensive its correction cost

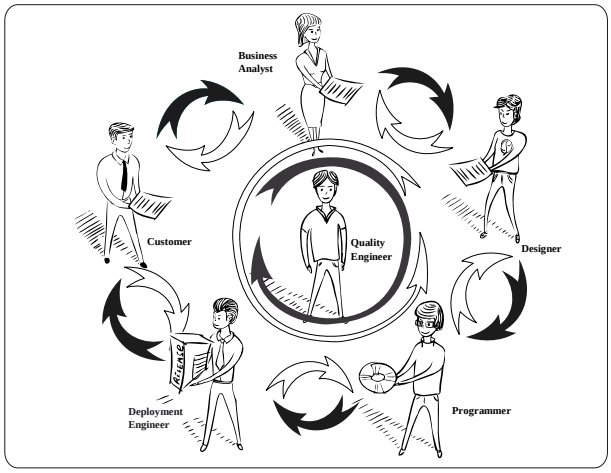

Fig. 4. Reverse Semantic Traceability confirms the correctness of each "translation" from one artifact to another.

During the last two years the authors conducted several experiments aimed at discovering possible usage scenarios for the Reverse Semantic Traceability and to evolve and improve this method. In most cases architects who participated in these experiments started to apply Reverse Semantic Traceability in their everyday jobs. The authors have received many enthusiastic feedbacks from practitioners, e.g., "*I am absolutely positive that Reverse Semantic Traceability approach encourages thinking out of the box, therefore, discovering and correcting faults of the design that would likely be missed otherwise*" or "*It turns out that after participating in an RST session a newcomer integrates into the project as if she was involved into it for months. This way to familiarize newcomers with the project is significantly more effective than forcing them to study the project documentation for days*", etc. Based on these feedbacks, the most popular usage scenarios are:

- Validating UML models: Quality engineers restore a textual description of a domain, original and restored descriptions are compared.
- Validating model changes for a new requirement: Given original and changed versions of a model, quality engineers restore the textual description of the requirement; original and restored descriptions are compared.
- Validating a bug fix: Given an original and a modified source code, quality engineers restore a textual description of the bug that was fixed; original and restored descriptions are compared.
- Integrating a new software engineer into a team: A new team member gets an assignment to do Reverse Semantic Traceability for the key artifacts from the current projects.

Early adopters of INTSPEI P-Modeling Framework employ different SDLCs, have different visions about using formal approaches to quality control, create software for various industries and operate in different business models. Due to the experimental nature of Reverse Semantic Traceability they all adopted it in different ways, and some of the resulting usage scenarios are quite far away from what was initially researched in the authors' experiments. For example, a CMMI-4-level company decided to use Reverse Semantic Traceability to incorporate quality-control procedures into their risk management process: Testers restore original descriptions of risks from the descriptions of contingency and mitigation plans created for these risks, and then original and restored descriptions are compared to make sure that the plans really address the risks.

The experience reports from current users of Reverse Semantic Traceability provide strong evidence of the method's efficiency. The common themes of experience reports are:

- The length of software development projects is cut down for 5 % – 35 %;
- The length of the integration period for new software developers is cut down for 25 % - 60 %.

There are very impressive examples of even better improvements for some specific performance indicators.

Speechless Modeling

As it has been described above, the Speechless Modeling was initially used while teaching students Object-Oriented Analysis and Design with UML. Then it was accidentally discovered that this technique allows increasing productivity of designers, and the authors organized several experiments to research this effect. After these experiments a number of software development companies started to incorporate speechless modeling sessions into their SDLCs.

They all report that it allowed them to shorten time spent on design. Most of them also report that speechless modeling sessions are very tiring for designers, and their efficiency decreases if they are repeated frequently. For example, a project manager from a company which started to use speechless modeling says: "*Using speechless modeling sessions our designers now spend one day on a design task, that before required 5 days to complete. However, it takes so much energy from designers that they are usually not able to work during the very next day after they worked in speechless mode, so we make it a day off for them to recover. Anyway, we spend 2 days on a task which had been taking 5 days before.*"

A few companies have also reported that they successfully used this technique for their team-building purposes: "*This is wonderful means to build the team. It reveals leaders and creates more expressive design. Moreover, speechless mode encourages focusing on ideas but not on the words which express those ideas. The absence of verbal communication promotes more effective information exchange and prevents repeating meaningless Buzzword-Bingo words*"

P-Modeling Sessions

P-Modeling Session is a modeling event that typically takes one or one and a half days. It combines both Speechless Modeling and Reverse Semantic Traceability into one tool for two design teams who work in parallel. The structure of P-Modeling Session is defined on **Ошибка! Источник ссылки не найден.**; it follows the format of the CMMI-P-SPEM experiment, which was organized by the authors during the Software Engineering Conference in Russia in 2005 (Moscow, November 2005) (Ref 10).

The P-Modeling Session provides an opportunity to get synergy from using, together, the two key techniques discussed above. During these sessions two teams work on two different assignments. Initially they do the modeling job in a speechless mode, then they perform Reverse Semantic Traceability for each other, and finally, they correct/improve their models.

This is a typical impression from the session: "*We decided to conduct a P-Modeling Session to elaborate the architecture of a large system. We assembled a team of 15 engineers. In order to kill two birds with one stone, we put a few novice engineers into the team. It is needless to say that the birds were killed. The result was shockingly impressive: the team indeed created an elegant detailed design of supreme quality*". The authors received extremely positive feedback about using P-Modeling Sessions for the following purposes:

- first design session for a new project during its envisioning/inception phase;
- team-building exercise for a newly composed development team

It should be noticed that teams which implemented P-Modeling Sessions decided not to do it more often than 1-2 times per project.

Conclusion

This paper presents INTSPEI P-Modeling Framework – an "add-in" to traditional SDLCs aimed at improving productivity of modeling process and increasing effectiveness of quality control procedures on all phases of SDLC. INTSPEI P-Modeling Framework is based on two key techniques: Reverse Semantic Traceability and Speechless Modeling. Both techniques were created as a result of the authors' research experiments. The paper presents the first feedback about using

Table Approximate schedule of a P-Modeling Session

Team A	Team B	Duration	Speechless
Introduction, ice-breaking activities		1 hour	no
Speechless work on modeling assignment A, creating the first version of model A	Speechless work on modeling assignment B, creating the first version of model B	2-5 hours, includes speechless lunch	yes
Reverse Semantic Traceability for the model B, created on the previous phase	Reverse Semantic Traceability for the model A, created on the previous phase	1 hour	no
Analyzing results of the traceability session, conducted by the team B; creating the second version of model A	Analyzing results of the traceability session, conducted by the team A; creating the second version of model B	1 hour	no
Closing the session		1 hour	

these techniques in real-life business environment. This feedback shows applicability and effectiveness of both of the discussed approaches. Combining the two techniques into a P-Modeling Session allows us to reach synergy and achieve the highest impact.

Acknowledgements

We are very thankful to Stanislav Busygin, Erika Short (University of Florida), Tatyana Taganskaya (International Software & Productivity Engineering Institute) and Dmitry Bednyak (Digital Road Studio) for their valuable contribution into preparation of this report.

1. *Unified* Modelling Language *2.05*, Object Management Group, http://www.omg.org / technology/documents/formal/uml.htm
2. *Grady Booch , James Rumbaugh , Ivar Jacobson.* The Unified Modeling Language user guide. – Redwood City, CA: Addison Wesley Longman Publishing Co., Inc., 1999
3. *Vladimir L. Pavlov, Anton Yatsenko.* Using Pantomime in Teaching OOA&OOD with UML // Proceedings of the 18th IEEE Conference on Software Engineering Education & Training (CSEE&T'05), 2005. – P. 77 – 84.
4. *Vladimir L. Pavlov, Anton Yatsenko.* The Babel experiment: an advanced pantomime-based training in OOA&OOD with UML // February 2005, ACM SIGCSE Bulletin, Proceedings of the 36th SIGCSE technical symposium on Computer science education SIGCSE '05. – 2005. – Vol. 37, Issue 1.
5. *Jacobson, I., Booch, G., and Rumbaugh, J.* The Unified Software Development Process. Addison-Wesley, 1999.
6. Microsoft Solutions Framework. Microsoft, http://www.microsoft.com/msf
7. *Nina S. Godbole.* Software Quality Assurance, Alpha Science Int'l Ltd., 2004.
8. *Guide* to the Software Engineering Body of Knowledge (2004), IEEE Computer Society.
9. *Steve McConnell.* Upstream Decisions, Downstream Costs // Windows Tech Journal, November 1997 (http://www.stevemcconnell.com/articles/art08.htm)
10. *http:*//www.secr.ru

Date received 16.04.2007

About the authors:

Vladimir L. Pavlov[1],
Senior Member of IEEE; member of ACM and PMI,

Nikita I Boyko[2],
Member of ACM

Alexander V Babich[1],
Member of ACM

Author's affiliation:

[1] International Software & Productivity Engineering Institute,
1979 Marcus Avenue, Lake Success, New York, 11042, USA
Phone: +1-877-468-7734
Fax: +1-877-291-9090
Email: vpavlov@intspei.com, ababich@intspei.com
URL: http://www.intspei.com

[2] *University of Florida, Department of Industrial and Systems Engineering,*
303 Weil Hall, P.O. Box 116595, Gainesville, FL 32611 6595, USA
Phone: +1 352 392 3389 ext. 2033
Fax: +1 352 392 3537
Email: nikita@ufl.edu
URL: http://www.ufl.edu/