HOW JAZZ ROCKS TEACHING ITERATIVE SOFTWARE DEVELOPMENT

Utilizing IBM Rational Team Concert at the University of Ostrava

Jaroslav Prochazka
University of Ostrava / TietoEnator Corp., 30. dubna 22, Ostrava, Czech Republic

Keywords: Teaching software engineering, Rational Team Concert.

Abstract: This paper summarizes our experience with Software Engineering lectures. These lectures are focused on practical experience with iterative and agile way of developing software. Paper describes the way from classical tool like SVN, text editors and wiki to IBM Rational Team Concert. We stress the aspect how this tool supports the way of working and understanding of theoretical principles.

1 INTRODUCTION

Our team of mentors in TietoEnator Corporation has piloted beta versions of IBM Rational Team Concert in 2007 and 2008. We see that this tool helps us to incorporate software development best practices in our daily life and helps us also to explain teams what we mean by agile planning, traceability, iterative framework and other "buzzwords from the books".

At the same time we were conducting lectures and tutorials at the University of Ostrava and learnt of IBM University program. So we wondered how to leverage Rational Team Concert also during lectures and tutorials to increase student's understanding of theoretical principles and practices.

2 TUTORIALS AND TOOLS

To explain the context and starting point, we briefly introduce content of two semestral practical tutorial called Grade project and also introduce IBM tool called Rational Team Concert (RTC).

2.1 Grade Project Tutorial

The Grade project is 2 semestral practical tutorial focused on teaching basic software engineering practices. Students attending this tutorial have already knowledge of object oriented programming (Java and C#), database knowledge as well as basics of software engineering. Students group into the teams, choose one predefined theme and iteratively develop final product.

Basic features of the Grade project are the following:

- Teamwork as the most important objective;
- Iterative development following OpenUP (Kroll and MacIsaac, 2006) based process prepared by lecturers;
- Agile and distributed way of working with higher ceremony (see Fig. 1) and very short iterations – tutorial is held once a week;
- Lecturers behave as mentors/coaches as well as technological architects (Java and .Net).

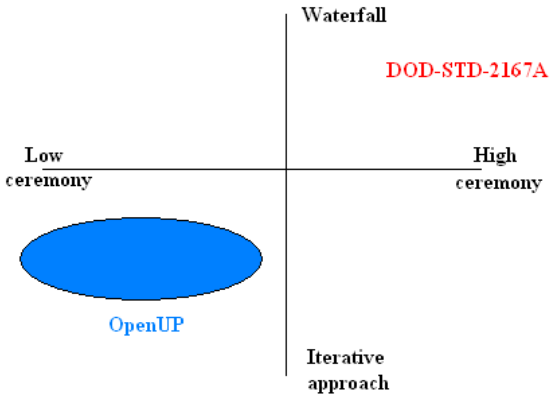

Figure 1: OpenUP framework and its ceremony.

Higher ceremony means the level of control of distinct artefacts and components, not the amount of

Prochazka J. (2009).
HOW JAZZ ROCKS TEACHING ITERATIVE SOFTWARE DEVELOPMENT - Utilizing IBM Rational Team Concert at the University of Ostrava.
In *Proceedings of the First International Conference on Computer Supported Education*, pages 225-229
DOI: 10.5220/0001967202250229

created or generated not used documentation. Higher ceremony in our case means performing architectural and code reviews by lecturers as well as formal iteration planning and assessment meeting led by lecturers (mentors).

The first semester of the Grade project is focused on Inception and Elaboration phase. The second one is focused on Construction phase and adding agile practices.

Technologies used to implement the solution are Java and .Net. These technologies have been chosen as modern ones with a bunch of supporting tools (Java) and subtle IDEs (both). Another reason is RTC support. RTC supports Java development because it is built upon Eclipse platform and offers also connector for Microsoft Visual Studio IDE (in Beta 1 version).

Grade project 1 (winter semester) objectives:

- Form the teams;
- Agree project's vision and project plan;
- Establish the process and tools;
- Mitigate the main risks;
- Demonstrate executable architecture.

Grade project 2 (summer semester) objectives:

- Develop features in iterations;
- Iteratively add agile practices;
- Demonstrate and deliver beta release.

For more details about phases and their objectives, see (Kroll and MacIsaac, 2006).

These are limitations and constraints for the form of tutorial and student team's way of working.

2.2 IBM Rational Team Concert

Rational Team Concert (RTC) is a collaborative software delivery environment that empowers project teams to simplify, automate and govern software delivery. Automated data collection and reporting reduces administrative overhead and provides the real-time insight required to effectively govern software projects. Dynamic project provisioning enables day one productivity, while real-time collaboration helps significantly reduce scrap and rework. Rational Team Concert extends the capabilities of the team with integrated (Jazz net, 2008):

- work item management;
- build management;
- software configuration management (SCM);
- and the collaborative infrastructure of the Jazz Team Server.

Jazz platform is a platform and RTC product is built upon this platform. RTC is based on Eclipse and behind Jazz platform and RTC product development stands Eclipse team as well as reputable IT guru Erich Gamma.

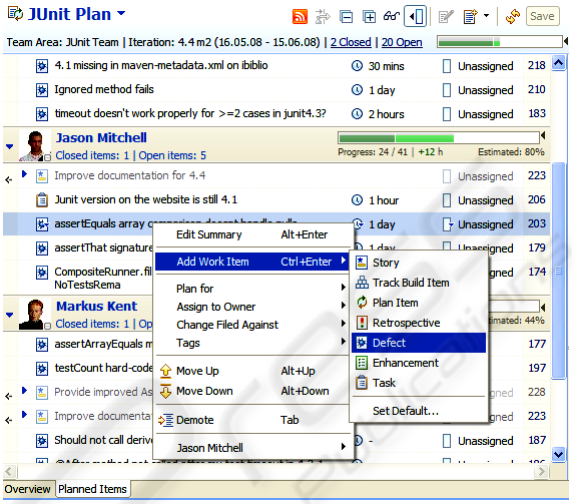

Figure 2: Iteration planned items for the team and easy of adding the new work item (Jazz net, 2008).

The greatest benefit of RTC from the teaching perspective is that RTC allows us to incorporate process rules into its behaviour. More about this feature is explained in the chapter 3.

RTC is available for universities for free via IBM Academic Initiative.

2.3 Other Tools

The Grade Project uses also other tools, mainly for business modelling. These tools are information system QI that incorporates results of Research Centre DAR (Prochazka and Klimes, 2007) and IDS Scheer's tool ARIS.

2.4 Brief History of the Grade Project

The Grade Project was taught for two years (4 semesters) with common tools like SVN as a code and document repository; wiki for sharing team plans, risks and notes; and university e-mail boxes as a communication mechanism. There wasn't any issue and task tracking tool in the place at that time to make number of different tools as low as possible.

At the end of every semester lecturers ask students for feedback to improve the tutorials. So also teaching itself is iterative and incremental.

The main problem for students was several different tools with different user interface, some

with command line only and none of them connected. IDE could be connected at least to the build mechanism and/or to SVN repository, but the overall integration, mainly code and task level was impossible. Apart from risk management and connection of risk mitigation actions with tasks. This was also reason, why we have excluded Jira and used only wiki for managing iteration plans, assessments and hi level tasks. We wondered how to improve and connect tools without increasing number of tools or coding in-house integrations.

Another input factor for change was student's misunderstanding of the principles. There was gap between the behaviour and understanding thus students didn't followed agreed and explained iterative approach. Students started development with database design, without understanding problem domain (no business modelling).

Finally, we have decided to leverage IBM Rational Team Concert.

3 WAY OF TEACHING

The initiation of the Grade Project is assigned to brief explanation of problem and to business modelling to gain the problem domain knowledge. IS QI and ARIS tool are used for this purpose. Business modelling is optional part and depends on complexity of the problem domain, because learning of another tool can even more decrease low productivity of student teams in the beginning.

It is obvious that bringing practices into the team is easier with the tool. If any problems, obstacles after the change, people tend to switch to the old mode, to the old way of working. Tool supports to make the change permanent by incorporating practices and bringing them into daily life. The RTC tool supports understand iterative framework; team agile planning; traceability and integration of tasks with delivered components.

What is more important is support of distributed way of working and incorporation of development process rules. The first mentioned fact is crucial, because students have only one tutorial per week that lasts 1.5 hour. During this time we can only conduct iteration assessment and planning for the next iteration and discuss problems and impediments. There is no time to develop the code during tutorial. Thus students need to analyse, design, test and develop, share their codes and extensively communicate during the week.

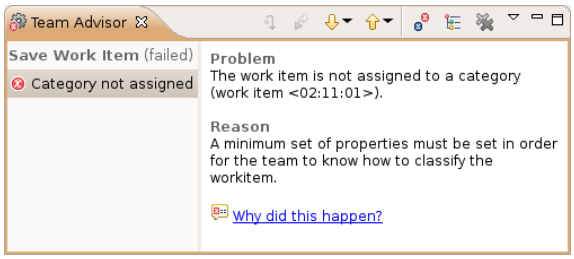

Figure 3: Process rules control result displayed in Team Advisor window – not assigned work item to any category.

The second mentioned fact, development process rules, can be said by lecturers, but students do not need to follow them. The great benefit of RTC from the teaching perspective is that RTC allows us to incorporate process rules into its behaviour. That means direct support of our software development process. If we have rule saying no delivery (commit in SVN speech) to repository without code review or jUnit test and we try to deliver without it, RTC will announce process rule violation and will not deliver to the stream (see Fig. 3, 4 and 5).

Software development process rule definition can be done on different levels. The basic behaviour definition is process definition. We follow OpenUP/RUP phases with one week iterations within each phase in Inception and Elaboration. The length can be different in later Elaboration and Construction phases. This is the main line and calendar driving team planning, demonstration and assessment (see following figure).

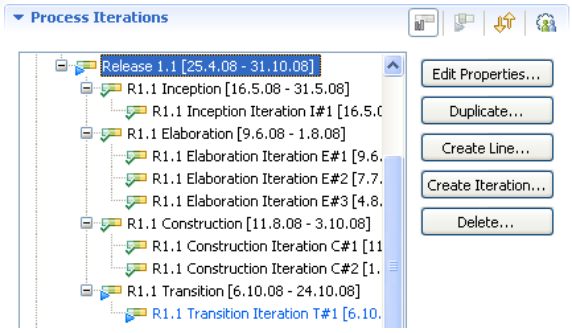

Figure 4: Our development process based on OpenUP.

Another specific behaviour that students have to follow can be defined on activity level for each system activity (e.g. creating work items, delivering to the stream, creating baselines and report and many others). Following example shows student's attempt to deliver to the stream without the rights.

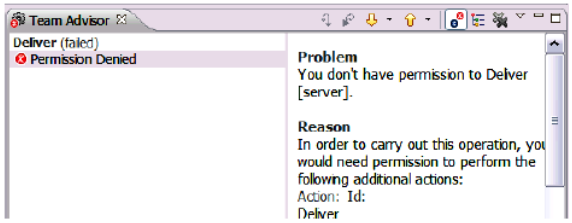

Figure 5: Process rules control result displayed in Team Advisor window – not allowed delivery.

According to this specific rule related to the delivery action only team lead is allowed to deliver. Contributor role has not checked this action.

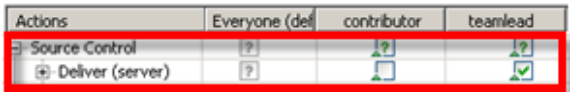

Figure 6: Process rules control result displayed in Team Advisor window – not allowed changes.

Another artefact supporting team work is so called Planned Items view giving us the overall view over team members work items (see Figure 2 above). This view helps team members keep synchronized. We can see who is working on what and helps us to avoid conflicts and/or missed tasks. The progress bar shows how many hours remain to the end of iteration so students can see if they are able to manage it or if they need to ask for help (it exerts the pressure on students).

Support of iteration planning and assessment is great in two ways. The basic artefact is wiki page for defining objectives and evaluation criteria. Another page is list of planned work items (see Figure 2 above) for each team member. Work items can be easily derived from the wiki pages and/or linked back. This mutual connections boost up direct generation of tasks, no duplication is needed.

Component ready for delivery can be easily linked with task, bug and requirement. This gives us information for every delivery. Why did this delivery happen? What was this bug about and how to reproduce it? What has changed in this component? If we talk about linking work items with delivery to the stream, we have to mention traceability as well. RTC directly creates bindings among different RTC artefacts (code, builds, work items, components). This information is very beneficial. It is important to know if some requirement is implemented or not. A lot of defects related to one component reveal us the needed for refactoring and many others.

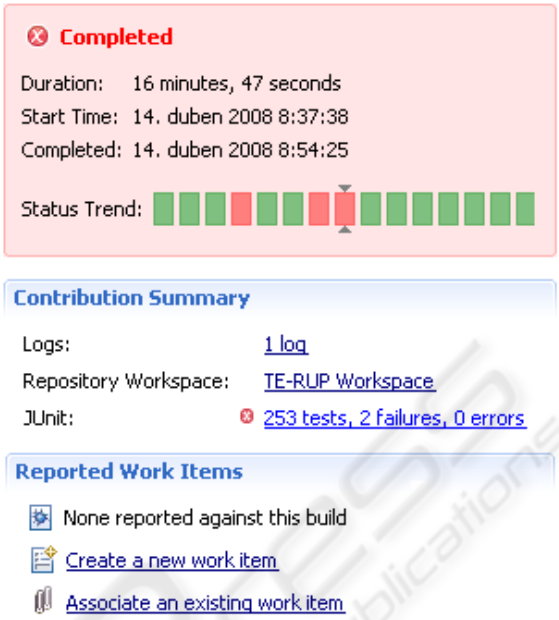

Figure 7: Build result with direct link to failed unit tests.

Nowadays has each and single iteration its overview page in wiki style listing objectives and evaluation criteria; risks and mitigation actions that are directly linked with work items. The team generates detailed tasks based on objectives and see in the Team view the progress and can propose actions if problems occur. All the tasks should be closed at the end of the iteration (after the week). Evaluation criteria are used to assess if work is done or not. Iteration assessment and another iteration planning are conducted after demonstration to the customer (lecturer in this case).

To boost up team communication in university environment, students are evaluated as the team at the end of the semester. The whole team succeeds or fails as in real projects.

To summarize at the end what is the benefit of Rational Team Concert in practical tutorials in comparison to common tools like Jira, wiki, SVN and Ant or Cruise Control, I would name the following:

- RTC supports to follow basic incorporated practices (e.g. iterations, performing team planning and assessment) – students do not need to think about them but simply follow the rules.
- Process behaviour can be defined as well as process itself (different levels).
- Support of distributed development – the must for university projects; plus instant messaging integration.

- Integration and Traceability – all parts are connected together; requirements can be mapped to the code; bugs can be traced to the requirements; all deliveries are assigned to tasks.
- Possibility to have all necessary information integrated in development IDE (Eclipse and MS Visual Studio).
- Students get familiar with modern, leading edge tools.

4 CONCLUSIONS

This paper summarizes the way we use modern tool called Rational Team Concert at the University of Ostrava to support teaching software engineering practices.

We explained objectives of the taught practical tutorial called the Grade Project and compared RTC with common tools like Jira, SVN and Ant from the teaching perspective.

The most visible advantages from the teaching perspective are following: tool supports new way of working (understanding of the principles practically – learning by doing); students have to follow incorporated rules and process and tool supports distributed development.

REFERENCES

Kroll, P., MacIsaac, B., 2006. *Agility and Discipline Made Easy*, Addison-Wesley Professional. 1st edition.

Jazz net, 2008. *Jazz platform homepage*. Available at jazz.net.

Prochazka, J., 2008. Experience from Agile Adoption in Distributed Environment. *In Proceedings of 34th International Conference on Software Development TSW2008*, VSB-TU Ostrava. pp. 156-163.

Prochazka, J., 2007. *Grade Project 1, 2*. Tutorial course book. University of Ostrava. Ostrava, 1st edition.

Procházka, J., Klimes, C., 2007. Methodical and Application Framework for Process Modeling. *In Proceedings of International Conference on Enterprise Information Systems and Web Technologies 2007*. ISRST, Orlando, Florida, USA. pp. 204-210.