Serdica J. Computing **1** (2007), 241–254

Serdica
Journal of Computing

Bulgarian Academy of Sciences
Institute of Mathematics and Informatics

AGILE CASE STUDY EVALUATION IN MIDDLE SIZE PROJECT

Penko Ivanov, Sylvia Ilieva

Abstract. In the last few years Agile methodologies appeared as a reaction to traditional ways of developing software and acknowledge the need for an alternative to documentation driven, heavyweight software development processes. This paper shortly presents a combination between Rational Unified Process and an agile approach for software development of e-business applications. The resulting approach is described stressing on the strong aspects of both combined methodologies. The article provides a case study of the proposed methodology which was developed and executed in a successful e-project in the area of the embedded systems.

1. Introduction. There are no two identical projects in service oriented companies. Nevertheless the IT projects that are executed within these companies might be categorized according to some indicators. Most common these indicators include:

ACM Computing Classification System (1998): K6.1, K6.3.
Key words: agile methodologies, agile management, software process, software metrics.

- The technical aspect of the project – there are different types of projects according to their complexity, technology and their place in the other aspects of the industry and/or technology.
- The size and the scope of the project – how big is the project and how many people participate in the project team is always a project feature that is described in all software development processes and metrics.
- Type of client – some clients know very well from the project beginning what should the final product look like while still there are clients that can only describe the broad-brush picture of the product functionalities and the final system functionalities are refined "on he fly".

Companies involved in the software industry always try to find out a universal methodology for the whole company attempting to adjust all the different classes of projects in one and the same workflow. This leads to some discrepancies in the project execution due to the project specifics listed above.

Traditional and very common known in the agile society as "heavy" approaches are characterized with the following features:

- Well planned from the beginning
- Predictable from the beginning
- Well documented at all project phases
- Suitable for any team and project size

While the agile methodologies stress on the following [4]:

- De-emphasizing a lot of up-front effort in analysis and design and documentation
- Small teams
- Incremental development
- Time-boxed scheduling with dates that are not allowed to slip.

The reasons for these commonalities in the agile methodologies support the image of agile and lightweight development. No heavy-duty process, no reams of useless documentation to weigh down the process [3]. Some of these methodologies are well known in the software industry. The most popular among them are [1]:

- XP (eXtreme Programming)
- Crystal Family
- Adaptive Software Process
- Scrum

The purpose of the paper is not to describe all the agile methodologies and point their advantages while to stress on the case study of a successfully used combination between two approaches in an e-project in ProSyst Software Gmbh Company. These two approaches are namely RUP and SCRUM. Nevertheless, the stress of this article is upon the possible combinations of a company ISO certified process with ideas and practices from the agile software engineering that produce a methodology which tries to balance between the traditional and agile methods of software development according to the specifics of the project.

The article is structured as follows: Section 2 presents the problematic and the types of projects in ProSyst Company. Section 3 presents the challenging project within the company that is reviewed. In section 4 are outlined the methodologies used for the execution of the project. In Section 5 experiment results are given. Section 6 concludes the paper.

2. The Problematic Domain – Fit the Process to the Project. All software projects are different and require different strategies. Experienced project managers plan extensively, they realize that plans are just a beginning; and try to predict the changes. In that aspect the following two management approaches become a main factor on dealing with the changes:

The practices of agile software development – short iterations, continuous testing, self-organizing teams, constant collaboration (daily integration meetings and pair programming for example), and frequent replanning based on current reality – are all geared to the understanding of software development as an empirical process and changes are likely to occur. In this aspect the agile approaches are driven by the changes.

On the other hand, CMMI was invented to help assessing contractors' abilities to deliver correct software on time, and in this regard, it has been a great success. But CMMI is not the answer for every organization. [5], [12]. In particular, CMMI does not tell an organization how to implement improvements in software development—it merely indicates where they're needed. CMMI models are not themselves processes or even process descriptions.

As it is very common to say the truth is somewhere between these poles. In many case studies is described the possible combination between methodologies situated on the different branch of development approaches spectrum. Still the specifics of the certain project may prove or reject the usefulness of the selected combination. That is why fitting the process to the concrete project should be carefully assessed prior the project beginning.

3 The Challenge. ProSyst company is a provider of embedded Java and OSGi compliant software [13], [14]. ProSyst offers client and server side OSGi service platforms as well as generic and custom applications that will allow devices to evolve and adapt their capabilities by installing new software components on demand. The corporate customers are able to remotely start or stop a client's application, upgrade a client device and troubleshoot any software problems, fixing them immediately. ProSyst enables the time-and cost-effective development, deployment and management of services and applications.

ProSyst intends to make its customers move into this space as simple, rapid, and profitable as possible, by spearheading advances in open end-to-end Java/OSGi technology-based architecture and developer productivity.

ProSyst has created a comprehensive software OSGi based framework for extending Internet-enabled transactions to a wide range of network-edge (embedded) devices, which can enable end-to-end solutions across multiple market segments. Embedded and server-side software can help device manufacturers, application developers and platform integrators to streamline the production of small-footprint applications that will generate new revenue streams, and enrich their products with innovative service packages. ProSyst solutions simplify the lifecycle management of these products by providing object-oriented tools based on open standards - built from the ground up for the embedded environment.

The project that has been developed and described is build upon the existing OSGi framework provided by ProSyst. The positioning of the product is in the residential (end users' homes) area. It provides browser based access of the users to their home appliances, access via GPRS and palm and handheld devices to their homes as well as means for management of all remote enabled devices. The project also involved the implementation of the embedded drivers, gateways and networking for accomplishing a full end user product. The technology used is mostly Java, OSGi with supporting implementation of native logic and applications.

4. The Approach.

4.1. Rational Unified Process. Rational Unified Process [10] was previously adopted in the company and the company is ISO 9001 [9] certified with this software development process.

The Rational Unified Process or RUP product is a software engineering process. It provides a disciplined approach to assigning tasks and responsibilities within a development organization. Its goal is to ensure the production of high-quality software that meets the needs of its end users within a predictable schedule and budget.

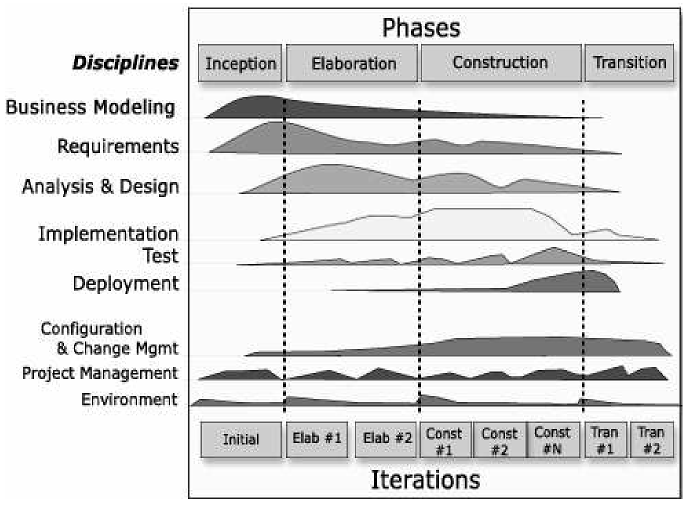

Fig. 1. Rational Unified Process

Figure 1 shows the overall architecture of the RUP. The RUP has two dimensions:

- the horizontal axis represents time and shows the lifecycle aspects of the process as it unfolds
- the vertical axis represents disciplines, which group activities logically by nature.

The first dimension represents the dynamic aspect of the process as it is enacted, and it is expressed in terms of phases, iterations, and milestones.

RUP captures the best practices in software development including controlled iterative development, requirement management, component based development, modeling with Unified Modeling Language (UML), quality assurance and change management.

RUP is not generally considered particularly "agile". The method was originally developed to cater for the entire software production process, and therefore contains extensive guidelines for process phases that are next to irrelevant in an environment that is likely to call for an agile approach. Recent studies examined the potential of RUP for using the method in an agile manner. It appears that the adoption phase is the key to adjusting RUP towards agility. The complete RUP lists over a hundred artifacts to be produced in the various process stages, necessitating rigorous screening, through which only the essential ones are adopted. One of the major drawbacks of RUP is, however, that it fails to provide any clear implementation guidelines

RUP can often be adopted, in whole or in part, "out of the box". In many cases, however, a thorough configuration, i.e. modification of RUP, is suggested before implementing it. The configuration yields a development case, which lists all deviations that have to be made with respect to a complete RUP.

The adoption process itself is an iterative six-step program, which is repeated, until the new process has been completely implemented. Each increment brings in new practices to implement, and adjusts the previously added practices. Based on feedback from the previous cycle, the development case is updated if needed. Piloting the process first in a suitable project is suggested.

Despite the need for extensive tailoring, however, RUP has been implemented in many organizations. The success of RUP may also be, to some extent, due to the fact that Rational Software sells popular tools to support the RUP phases, for example, Rational Rose, a UML modeling tool.

4.2. SCRUM. SCRUM is defined as an agile methodology for software development. The process shortly presented includes three phases: pre-game, development and post-game [15] (Figure 2).

The process begins by creating the project backlog—a list of all deliverables. These can be user stories (as in eXtreme Programming (XP) practice [2]) or requirements. The backlog is never empty as long as the program is maintained. It constantly changes. It is updated by only one person, the designated project owner, who also sets the priority of the items. The backlog is always visible and anyone can influence the project owner.

The project is divided into 2-4 week Sprints. Each team keeps a team backlog, which constantly changes. All stake holders meet to determine what will be done in the next Sprint. A deliverable must be complete—a visible, usable increment (that builds on previous increments as development proceeds). The

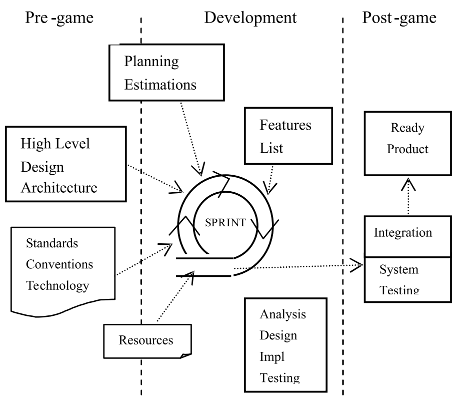

Fig. 2. SCRUM

backlog creation and setting of priorities should be done with the customer (one of the stakeholders).

During a Sprint, no interruptions are allowed from the outside. A short meeting is held daily or every other day where the team lead or ScrumMaster asks three questions of every team member:

1. What have you completed, relative to the team backlog, since the last Scrum meeting?
2. What held you up?
3. What do you plan to do between now and the next Scrum meeting?

At the end of the Sprint the stakeholders meeting is held where the latest achievements are presented and discussed. This practice might be discussed and presented as an alternative to the "Customer onsite" practice in the terms of XP.

- Increments are delivered.
- Surprises are reported.
- Anything can be changed.
- New estimates and team assignments are made for the next Sprint.
- The project can be cancelled.

The Scrum approach has been developed for managing the systems development process. It is an empirical approach applying the ideas of industrial process control theory to systems development resulting in an approach that reintroduces the ideas of flexibility, adaptability and productivity [16]. It does not define any specific software development techniques for the implementation phase, unlike XP. Scrum concentrates on how the team members should function in order to produce the system flexibly in a constantly changing environment.

The main idea of Scrum is that systems development involves several environmental and technical variables (e.g. requirements, time frame, resources, and technology) that are likely to change during the process. This makes the development process unpredictable and complex, requiring flexibility of the systems development process for it to be able to respond to the changes. As a result of the development process, a system is produced which is useful when delivered.

Scrum helps in improving the existing engineering practices (e.g. testing practices) in an organization, for it involves frequent management activities aiming at consistently identifying any deficiencies or impediments in the development process as well as the practices that are used.

Scrum is the agile methodology that is most useful during the project execution phase and can contribute to most of the defined company problems during the implementation stage of the project. It does not cover all the project aspects like configuration management and business modeling. This gives a good background for combing the two methodologies within the company.

4.3. The selected approach. Selecting the correct process for each project is always a difficult job and involves decisions and discussions from the middle and top management of the companies. Middle managers, especially the project managers are always the initiating side in such discussions. Their purpose is to present the new resulting process to the top managers who feel pretty comfortable with the established and proven process. Changes are likely to happen only when the motives are well supported with facts and examples. One of the most important requirements is to ensure that failures are not likely to happen and no chaos will be introduced within the company with the adoption of the new process [6].

For that reason new agile approaches are difficulty accepted within the middle and large size companies. Most often the traditional ways of developing software established in the companies (like RUP in ProSyst but still many companies rely on the Waterfall model) are not changed suddenly at one step [11]. The process suffers several modifications that take some best practices from other approaches and more often from agile approaches. This leads to a small step evolution of the companies towards the dynamics of the software development nowadays.

Both processes – RUP and SCRUM are iterative and have very common features. SRUM is more designed as an agile approach and is designed for small teams. It is usually noted that the team should not exceed more than 10 people.

On the other hand RUP stresses also on iterations but there are no defined sprints with restricted timeframe for iterations. This possibly may introduce some slippage in the project finish especially for projects with a tight schedule and certain end date. RUP also does not narrow the size of the development team which is a feature positive for the selected ProSyst project.

ProSyst has decided to benefit from both processes defining sprints with duration of exactly 4 weeks while still defining time for specification and specification approval of the system framework at the beginning of the project. The specification defined together with the client included the constraints and the general architecture of the system while the modules and the GUI of the modules were not detailed during that phase.

After the specification phase and the approval of the system architecture by the client started the iterative process of incremental development according to the sprints defined in SCRUM. At the end of each sprint is defined a discussion with the client that cannot be slipped or missed. Each of these discussions led to the refined requirements for the end of the next sprint. A feature list is created containing all the requirements that are currently known. The requirements can originate from the customer, sales and marketing division, customer support or ProSyst people participating in the meetings. The requirements are prioritized and the effort needed for their implementation is estimated. The feature list is constantly updated with new and more detailed items, as well as with more accurate estimations and new priority orders. Planning includes the definition of the project team, tools and other resources, risk assessment and controlling issues and verification management approval. At every iteration, the updated product feature list is reviewed by the team so as to gain their commitment for the next sprint. Experience from earlier increments allows better estimates and planning as project progresses. It's always easier to estimate shorter development periods.

The most important factor in the process was the so called "workshops" that are defined after each iteration. These workshops are actually meetings with the client that were held each month at the end of the SCRUM sprint. The meeting could not be postponed and the next workshops is defined and confirmed at the end of the currently held one.

The workshops with the client have two main aspects:

1. Present the achieved results during the passed iteration
2. Define and prioritize the requirements for the next sprint.

For this reason the client workshops last for at least two days. While presenting the achieved results from the passed sprint the client may introduce additional requirements and/or changes to the current implementation. This guaranties that the system developed from ProSyst is adequate and acceptable from the client. On the other hand, on the second day are defined the new requirements and functionality for the system developed. The next workshop is defined and confirmed – at the end of the next sprint.

A working version of the system (the result from the sprint) is installed on the client side and during the time while of the next iteration is carried the system is evaluated from the client representatives and the sponsors of the project. Ideas from all client departments are gathered and are presented as change request or new requirements during the next workshop.

5. Outcomes.

Presenting the achieved results is the most important stem in any experiment. The results should be carefully analyzed and presented to the middle and top management for review and future proposals on the company process evolution.

5.1. Project Frames.

The examined project is for a major client and is in the embedded field of technology. The system developed fits in the company projects portfolio based on the OSGi framework. Existing company products were used as a basis for the client specific needs.

In order to represent the results more adequately it is good to know the size of the project. The represented size is extracted after the project end (a major public event where the client represented the implemented product). The project size is presented in Table 1.

Parameter	**Value**
LOC	190 kLOC
Actual Effort	15700 personhours
Team Size	Up to 20 people

Table 1. Major Project Characteristics

The effort shown is the total effort for the project. I.e. these values include the efforts for project management, Specification, Design, business trips and so on.

The LOC represents the software lines of code implemented for the client. As mentioned at the beginning the system has been developed on the top of the existing company products (mainly the OSGi framework)

The project characteristics are provided in order to put the project size in certain frames for better readers' understanding. The team size varies during the project. The reason for this is the limited number of developers within the company that were also supposed to work on other projects (very common for the agile software development companies).

5.2. Effort Distribution. The described results do not represent a comparison with previous company projects but rather represents the percentage of the effort distribution towards different project activities.

The major positive outcomes were connected with the client attitude towards the ready product. The product release date was fixed according to a major public event and therefore no slippage in the deadline was possible. Having in mind the tight schedule for the project at the beginning it seemed too risky at all to point this event as a deadline.

With the continuous meetings with the client (so called workshops) was almost achieved the client onsite practice (XP). The client stays constantly in touch with the evolution of the product and proposes changes.

Meeting customer on a monthly basis provided the opportunity to meet different members of the client organization and gain impressions from most the stuff members that are going to use the product. This improved the overall acceptance of the product among the client staff members. Of course this led to the following distribution of the project efforts. It is visible that more then 5% of the time is spent in business trips and discussions onsite with the client (Table 2).

Effort Type	**Percentage**
Business Trips	5,15%
Specification	18,45%
Design	5,18%
Implementation	49,61%
Quality Assurance	10,61%
Project Management	6,52%
Other	4,48%
TOTAL:	100,00%

Table 2. Effort Distribution

It is known that applying an agile approach tends towards increasing the percentage of development effort and decreasing the time for specification and design. Even agile methodologies like XP [2] try to adopt these activities in practices like “System Metaphor”, “Simple Design” and “Collective code ownership”. Still the selected approach for the project is not entirely SCRUM, neither XP. Prior starting the implementation of the system a specification phase was defined and carried. During this phase ProSyst company together with the client tried to define as much as possible of the requirements for the system. All the requirements were described in several documents called “Software Requirements Specification” for the different major system modules (like user interface, backend functionalities, third party system integration). The specification of the system architecture consumed more than 18% of the effort. This statistics shows that there are still aspects of improvement. Considered as a pilot project this percentage is acceptable.

Another aspect of the necessity of specification phase was that the project was the first project held with that particular customer. This specification phase tried to guarantee enough confidence in the client at an early project stage towards a successful project completion.

The percentage of the design is relatively small. The design activities were mainly distributed in the monthly sprints during the implementation.

Still a good outcome appears the project management effort that is kept relatively low percentage. This appears a good starting point for future steps for decreasing the management overhead of the company projects.

About 4.5 % of the total effort is not distributed among the usual software development activities. These 4.5 % include efforts spent on system administration, environment setup and research on some of the items and new technologies used in the projects as a whole.

The time spent on quality assurance appears as a totally acceptable with the company. These effort include running some of the automate test cases defined and implemented during the project and periodical execution of specially created checklists for verification of the correct behavior of the developed system.

Summarizing the above it might be concluded that this combination of software development processes deals well with the requirement to deliver the solution in time and within the required budged with the required functionalities.

6. Conclusion.

Agile Methods are the ones that will drive the software processes to evolution. Agile Methods will probably not win over traditional methods but live in symbiosis with them. While many Agile proponents see a gap between Agile and traditional methods, many practitioners believe this narrow gap can be bridged. This case study depicts such combination.

Agile Methods may not out rule traditional methods because diverse processes for software engineering are still needed. Developing complex software for a space shuttle is not the same as developing software for a toaster [7]. Not to mention that the need to maintain software, typically a much bigger concern than development, also differs according to the circumstances. Software maintenance is, however, not an issue discussed in Agile circles yet, probably because it is too early to draw any conclusions on how Agile Methods might impact software maintenance.

One important factor when selecting a development method is the number of people involved, i.e., project size. The more people involved in the project, the more rigorous communication mechanisms need to be. According to Alistair Cockburn, there is one method for each project size, starting with Crystal Clear for small projects and, as the project grows larger, the less Agile the methods become [8].

The other factor for the correct process selection is the project itself. There are no two equal projects and it is difficult to be claimed that there is a universal process that solves all issues for all company projects.

In conclusion, the selection of a method for a specific project must be very careful, taking into consideration many different factors, including those mentioned above. In many cases, being both Agile and stable at the same time will be necessary.

REFERENCES

[1] Abrahamsson P., O. Salo, J. Ronkainen, J. Warsta. Agile Software Development Methods ESPOO 2002, VTT Publications, 2002, 478.

[2] Beck K. Extreme programming explained: Embrace change. Reading, Mass., Addison-Wesley, 2000.

[3] Beck K., M. Beedle, A. van Bennekum, A. Cockburn, W. Cunningham, M. Fowler, J. Grenning, J. Highsmith, A. Hunt, R. Jeffries, J. Kern, B. Marick, R. Martin, S. Mellor, K. Schwaber, J. Sutherland, D. Thomas. Manifesto for Agile Software Development, 2001, `http://AgileManifesto.org`.

[4] Boehm B. Get Ready For The Agile Methods, With Care. *Computer* **35**, No 1 (2002), 64–69.

[5] Chrissis M. B., M. Konrad, S. Shrum. CMMI Guidelines for process integration and product improvement. Addison Wesley, 2003.

[6] Cockburn A. Surviving Object-Oriented Projects: A Manager's Guide. Addison Wesley Longman, 1998.

[7] Cockburn A. Agile Software Development. Boston, Addison-Wesley, 2002.

[8] Cockburn, A. Agile Software Development Joins the "Would-Be" Crowd. *Cutter IT Journal* **15**, No 1 (2002), 6–12.

[9] Ketola J., K. Roberts. ISO 9000:2000 In a Nutshell, Paton Press, 2001.

[10] Kruchten P. The Rational Unified Process: an Introduction. Addison-Wesley, 2000.

[11] Kruchten P. From Waterfall to Iterative Lifecycle – A tough transition for project managers. Rational Software White Paper, 2001, `http://www.rational.com/media/whitepapers/TP173a.pdf`.

[12] Mutafelija B, H. Stromberg. Systematic Process Improvement Using ISO 9001:2000 and CMMI, Artech house, 2003.

[13] OSGi Alliance, OSGi Specification R3, R4 `http://www.osgi.org`.

[14] ProSyst Gmbh, 2006, `http://www.prosyst.com`.

[15] Schwaber K. Scrum Development Process. Proceedings of OOPSLA'95 Workshop on Business Object Design and Implementation, Lecture Notes in Computer Science, Springer-Verlag, 1995.

[16] Schwaber K., M. Beedle. Agile Software Development With Scrum. Upper Saddle River, NJ, Prentice-Hall, 2002.

Penko Ivanov
ProSyst Software Gmbh
e-mail: `p.ivanov@prosyst.com`

Sylvia Ilieva
Sofia University
e-mail: `Sylvia@acad.bg`

Received October 21, 2006
Final Accepted September 13, 2007