МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

МАТЕРИАЛЫ

VI Международной молодежной научной конференции

«МАТЕМАТИЧЕСКОЕ И ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ИНФОРМАЦИОННЫХ, ТЕХНИЧЕСКИХ И ЭКОНОМИЧЕСКИХ СИСТЕМ»

Томск, 24–26 мая 2018 г.

Под общей редакцией
кандидата технических наук И.С. Шмырина

Томск
Издательский Дом Томского государственного университета
2018

8. *Krenzler R., Daduna H.* Loss systems in a random environment // Probability. – 2013.
9. *Paz N.* Markovian Queues in Random Environment with System Failures // Oper. Res. Lett. – 2007. – 35. – P. 805–812.
10. *Jiang T., Liu L., Li J.* Analysis of the M/G/1 queue in multi-phase random environment with disasters // J. Appl. Prob. – 2015. – 18. – 236–244.
11. *BoxmaO.J.* The M/M/1 queue in a heavy-tailed random environment // Statistica Neerlandica. – 2000. – 54. – P. 221–236.
12. *Sophia1 S., Praba B.* The M/M/1 queue in a heavy-tailed random environment // International Journal of Pure and Applied Mathematics. – 2015. – P. 267-279.
13. *Zaiming Liu 1, Senlin Yu.* The M/M/C queueing system in a random environment // Journal of Mathematical Analysis and Applications. – 2016. – 436. – P. 556–567.
14. *Jianjun Li, Liwei Liu.* Performance analysis of a complexqueueing system with vacations in random environment // Advances in Mechanical Engineering. – 2017. – 9(8). – P. 1–9.

SIMPLE MODEL – A WAY TO INTEGRATE SOFTWARE DEVELOPMENT PROCESSES AND PROJECT MANAGEMENT SOFTWARE

D.O. Zmeev, D. Sokolov, O.A. Zmeev
Tomsk State University
denis.zmeev@accounts.tsu.ru, danila.sokolov@accounts.tsu.ru, ozmeyev@gmail.com

Introduction

In the present condition of the industrial software development project managers have a special role. The manager's role consists in managing a project, which implies monitoring its progress, risk analysis and identifying the key stages of development, building and managing a team, documentation and other activities.

To make the work of the manager easier there are many manuals, reference books, knowledge bases, that describe the project activities in quite different ways. The manager can approach studying those sources more competently to solve problems more efficiently. Many of the difficulties that project managers face are not exclusive to their subject area. For example, a skyscraper architect, a head of a department developing an operating system or even a student studying project management can face similar problems of motivating the team members [6]. But, naturally, there is a great influence of the subject area on project management. In software development this influence resulted in so-called software development process (SDP).

An SDP is effectively a set of instructions that should help organize teamwork in the most efficient way to reach the goals of a project. These instructions can range from a set of abstract principles describing the main work stages and correlation with the tasks to strict rules that specify the order of artefacts that should be added at each step of the working process of the project team. Practical application of the SDP has shown that there is no ideal and universal process. Every one of them has its advantages and disadvantages and is applicable for a certain group of projects. A project manager in software development uses either an existing SDP (it can be adapted for a certain project team) or tries to manage work relying on personal experience (but usually his experience is based on methods from different SDPs).

There are different project management software solutions (PM) created to aid managers. Such systems allow managers and their teams to formalize the work on the projects. At present there are many examples of software solutions the most popular of which are Microsoft Project [4], Team Foundation Server [5], Redmine [2] and Jira [3]. Some of the PM are targeted for specific development processes which means that their features focus on the specific aspects of a certain development process (for example, digital Kanban board for marking the progress of tasks). Nonetheless it is evident that regardless of the system (even if it is created for specific purposes) the observance of the development process in the software development project is the professional responsibility of the manager. That leads to the following problems:

1. Following the rules of an SDP in a PM leads to indirect costs (Which can be expressed as price for a processes-based PM or time to configure the PM according to the SDP rules). The more formalized process is the higher the costs.
2. Accordance of the SDP and the real process used by the team depends on the people involved. Often it is the case that the manager believes that he follows a certain method of an SDP when in fact he does not.

This work consists of the follows parts: first, we describe the main idea of the simple model of the SDP in the Section 1. The example of using the simple model of the SDP for the Feature Driven Development [1] software process and its implementation to Redmine project management software is discussed in the Sections 2 and 3. Finally, in the Conclusin the conclusions and plans for the future work are presented.

1. Simple model of SDP

We develop a simple model (not a meta-model) of an SDP for designing the solution which allow to integrate an SDP and PM. Such model allows to formalize basic interaction between different entities in an SDP. Additional advantage of this simplified approach is the possibility to avoid using specific terms and usual tools for different SDPs. A simple model of SDP is shown in terms of a UML class diagram in the Fig. 1. The suggested model does not represent the full description of the development process because its main purpose is to structure the main working entities which compose the SDP. Using the simple structure allows to transform the SDP into a sequence of tasks implemented in the PM and makes possible to develop automatic tools for such kind of implementation. This feature distinguishes this model from other ways of describing an SDP.

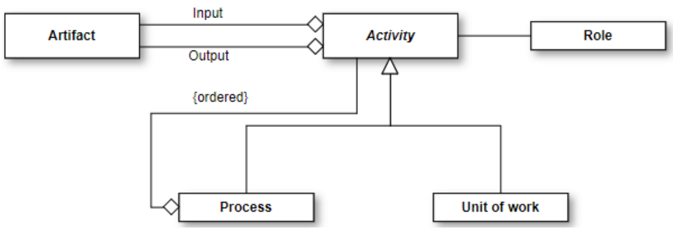


Fig. 1. Simple model of SDP

This model is based on the composite pattern, which allows creating hierarchical structures from elements. These elements are the entities: unit of work, process and activities. In this case the basic element (leaf for composite pattern) is the unit of work that in the implementation of the development process transforms into a task. The aggregating entity (composite pattern) [9] is the process that depending on the difficulty of the SDP structure can be the whole project (if it has no complicated hierarchical structure of actions) or can be an iteration, sprint, phase, etc. (if the SDP implies the using of such entities). Characteristically for the composite pattern while creating the aggregation of entities the order of their addition should be noted (in most developing processes the tasks follow each other in a certain sequence and it is impossible to run test-case before building the system).

Other important entities are artifacts and roles. An artifact denotes an activity in any SDP that is aimed at creation and modifying the output set of artifacts (code, project approval documents, terms of reference, working artifacts for the system design, etc.) from the input set of artifacts (which can be empty, especially at the early stages of the project, for example, collecting the primary requirements for the project). A role denotes a set of responsibilities in a certain activity in the SDP (architect, test engineer, developer, etc.), by assigning a task the project manager assigns a certain role to the team member. In case of the structured development process the roles are defined precisely and in detail, but nonetheless at different points in time any one team member can perform different roles. And if there is no role-based structure there are two default roles in the SDP: the manager and the team member.

Finally, to establish the interaction between the theoretical SDP and the actual implementation of the SDP to the project object-oriented programming dichotomy can be used. If this approach is applied to the development process the abstract activities, iterations, phases described in the development process (classes) implement in real projects into specific tasks, categories and labor costs that are characteristic for projects (objects). Due to this notion it is sufficient to describe the development process with only fundamentally different entities (the concept of a sprint in general), which in practice will be used to describe specific elements of the PM within which the project is implemented.

2. FDD expression using simple model of SDP

To implement the idea of integrating a simple model of an SDP and a PM firstly you need to design the structure of the selected software development process. In this article we use the Feature Driven Development (FDD). The FDD process is based on the main entity – feature and organizes all the work of a team to form, analyze, design, implement and promote features to the software. It is one of the Agile-based development processes, so it is also has the sprint entity to formalize the time-management part of the software project management. The structure of the FDD in terms of the simple model of SDP is as follows (Fig. 2 shows a UML class diagram describing the main entities related to the feature used at FDD for task management).

There are two entities that are used for task management at FDD: a top-level activity and a specific task. As for the roles which are not presented at Fig. 2, the FDD does not formalize the strict model of roles, one of the main ideas is to separate the whole team into small mixed groups (team members with different competencies) and distribute features between them. As for the artifacts we do not show them for clarity (they make the diagram overloaded) and despite each task has a certain type of artifacts in the FDD, we do not use them for implementation into the PM because each team has their own template for different artifacts. The main part of the FDD is to divide the whole sprint into 5 main activities and determine the default task for executing for each activity. Usually, this task transforms to task related to a specific feature which is determined in the first (Develop Overall Model) activity.

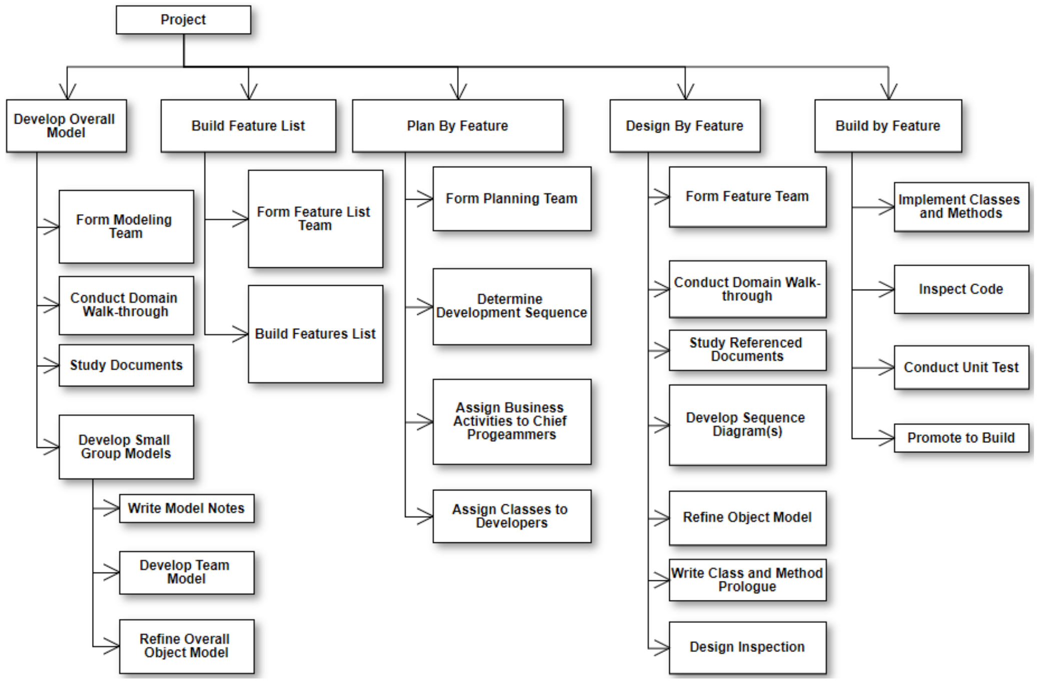

Project
Develop Overall Model
Build Feature List
Plan By Feature
Design By Feature
Build by Feature
Form Modeling Team
Conduct Domain Walk-through
Study Documents
Develop Small Group Models
Write Model Notes
Develop Team Model
Refine Overall Object Model
Form Feature List Team
Build Features List
Form Planning Team
Determine Development Sequence
Assign Business Activities to Chief Progeammers
Assign Classes to Developers
Form Feature Team
Conduct Domain Walk-through
Study Referenced Documents
Develop Sequence Diagram(s)
Refine Object Model
Write Class and Method Prologue
Design Inspection
Implement Classes and Methods
Inspect Code
Conduct Unit Test
Promote to Build

Fig. 2. Simple development process model of FDD

3. Integration between Simple model of FDD and Redmine

At the next step the software development process is integrated with the project management software. To do this, it is necessary to create correspondence between the elements of the development process expressed in the simple model hierarchy and the specific elements of the real project management system (Redmine in this work). The result is shown in Fig. 3.

Let us analyze this correspondence: **Project** is the same entity both in the SDP and the PM. **Sprint** is the entity which can be determined as the best time management practice for the Agile-based development processes correspond to versions mechanism in Redmine which allows the manager to determine time, amount of work, and tasks for each version. **Activities** is the general description for different stages of the sprint, there are two possible solution for it in Redmine, one of them uses specific trackers for each activity and the other is the parent issue feature. Given that the specific trackers make Redmine more limited to a determined implementation of the FDD we choose parent issue feature to express this hierarchy which allows to use other modifications of the FDD and other practices of different SDPs. And finally, the leaf element is **task** for the simple model of the SDP and **issue** for the Redmine system.

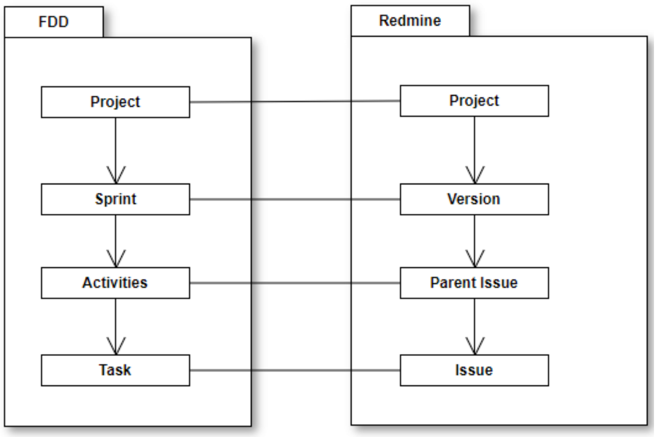
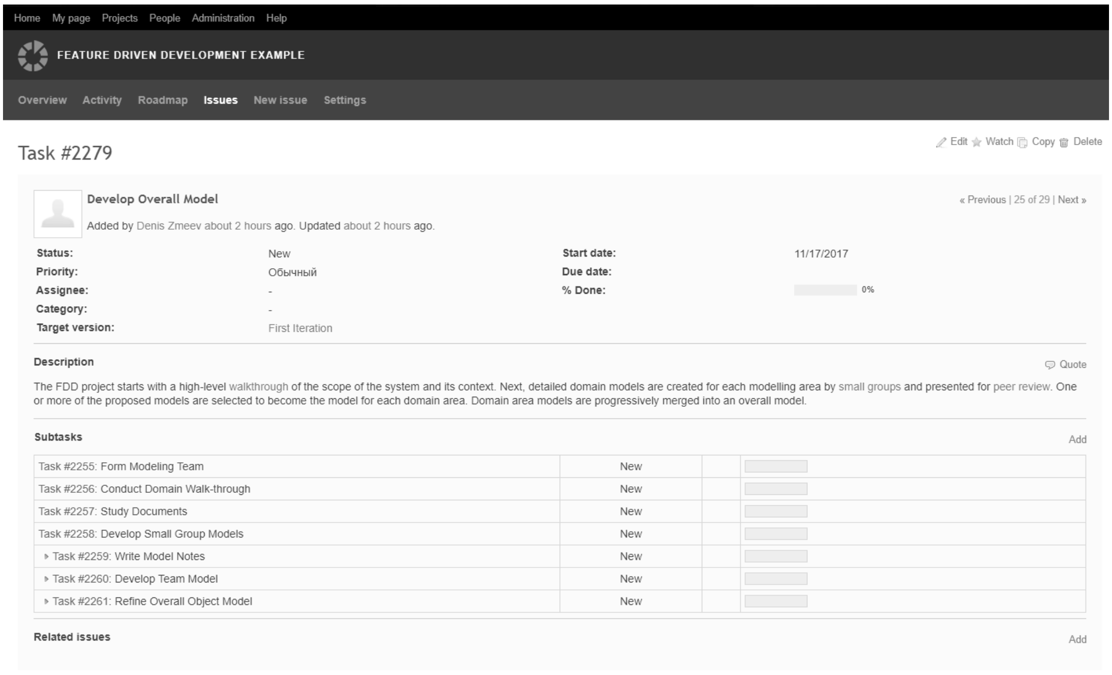


Fig. 3. Implementing the model into Redmine

At the last stage, the Redmine is filled with an empty FDD-prototype project, which allows to create new project based on the FDD development process by copying the prototype instead of repeated filling of projects by the same settings, default initial tasks and usual configuration. The example of the prototype filled “Develop Overall Model” activity as the parent issue is shown in Fig. 4


Home My page Projects People Administration Help
FEATURE DRIVEN DEVELOPMENT EXAMPLE
Overview Activity Roadmap Issues New issue Settings
Edit Watch Copy Delete
« Previous | 25 of 29 | Next »
Quote
Add
Add

Task #2279

Develop Overall Model

Added by Denis Zmeev about 2 hours ago. Updated about 2 hours ago.

Status:	New	**Start date:**	11/17/2017
Priority:	Обычный	**Due date:**	
Assignee:	-	**% Done:**	0%
Category:	-		
Target version:	First Iteration		

Description

The FDD project starts with a high-level walkthrough of the scope of the system and its context. Next, detailed domain models are created for each modelling area by small groups and presented for peer review. One or more of the proposed models are selected to become the model for each domain area. Domain area models are progressively merged into an overall model.

Subtasks

Task #2255: Form Modeling Team	New		
Task #2256: Conduct Domain Walk-through	New		
Task #2257: Study Documents	New		
Task #2258: Develop Small Group Models	New		
Task #2259: Write Model Notes	New		
Task #2260: Develop Team Model	New		
Task #2261: Refine Overall Object Model	New		

Related issues

Fig. 4. Example of "Develop Overall Model" activity

It should be noted that the fact that to integrate software development processes and the project management software were used only the native features of Redmine. It makes possible to migrate created prototype development process projects between different teams, companies, Redmine, which allows to apply reusability [8] to the software project management practices.

Conclusion

The simple model allows to express different development processes in understandable way which can be used as hierarchical structure for implementing to different project management software. At the current state of research this model allows to create pre-configured prototype project that allows different teams to use the same implementation of the SDP without spending additional time or financial resources. Future plans include integration of the simple model with the SEMAT [7] initiative which will allow to configure different software development processes as the cooperation of used practices in software engineering.

REFERENCES

1 En.wikipedia.org. (2017). Feature-driven development. [online] Available at: https://en.wikipedia.org/wiki/Feature-driven_development [Accessed 29 Nov. 2017].

2 Redmine.org. (2017). Overview – Redmine. [online] Available at: http://www.redmine.org [Accessed 29 Nov. 2017].

3 Atlassian. (2017). Jira | Issue & Project Tracking Software | Atlassian. [online] Available at: https://www.atlassian.com/software/jira [Accessed 29 Nov. 2017].

4 Products.office.com. (2017). Project Management Software | Microsoft Project. [online] Available at: https://products.office.com/en/project/project-and-portfolio-management-software [Accessed 29 Nov. 2017].

5 Webster, L. (2017). Agile, Git, CI with TFS | Team Foundation Server. [online] Visual Studio. Available at: http://www.visualstudio.com/ru/tfs/ [Accessed 29 Nov. 2017].

6 Pmi.org. (2017). PMBOK Guide and Standards | Project Management Institute. [online] Available at: https://www.pmi.org/pmbok-guide-standards [Accessed 29 Nov. 2017].

7 Semat.org. (2017). Welcome - SEMAT. [online] Available at: http://semat.org/home [Accessed 29 Nov. 2017].

8 En.wikipedia.org. (2017). Reusability. [online] Available at: https://en.wikipedia.org/wiki/Reusability [Accessed 29 Nov. 2017].

9 En.wikipedia.org. (2017). Composite pattern. [online] Available at: https://en.wikipedia.org/wiki/Composite_pattern [Accessed 29 Nov. 2017].