Mining MSF for Process Patterns: a SPEM-based Approach

Vladimir Pavlov
Intel*
Turgenyeva str, 30, Nizhny Novgorod,
603950, Russia
+7-(910)-790-3211
VLPavlov@ieee.org , http://www.vlpavlov.com

Dmitry Malenko
Dnepropetrovsk National University
Zaporozhskoe Hwy 82/75
Dnepropetrovsk, 49041, Ukraine
+380-(66)-780-3992
DMalenko@acm.org

ABSTRACT
Process Engineering is a young but important part of Software Engineering. A well-established and efficient software development process is a key factor in delivering high-quality mission critical software solutions. Today there are plenty of different methodologies for organizing software development; process patterns are proving to be an appropriate technique to extract key concepts from different software development processes for further reuse.

In this paper, we discuss process and organizational patterns discovered in Microsoft Solutions Framework (MSF) and describe them with the help of a Software Process Engineering Meta-model (SPEM). Employing a SPEM gives a more formal definition of patterns other than the informal, yet structured, textual description. The patterns discussed in this paper are: Living Document, Clonable Lifecycle, Smart Lifecycle and Stakeholder-Oriented Organization.

Categories and Subject Descriptors
D.2.9 [**Software Engineering**]: Management - s*oftware process models.*

K.6.3 [**Management of Computing and Information Systems**]: Software Management – *software process*.

General Terms
Management

Keywords
Process Engineering, Process Patterns, Organizational Patterns, UML, SPEM, MSF, XP.

1. INTRODUCTION AND BACKGROUND INFORMATION

Today it is difficult to imagine that large mission-critical software is developed using an undocumented ad hoc process, but even with proper control and management throughout the entire development cycle, it is incredibly difficult to deliver high-quality software. The software development process is a major contributor to the success of a product being developed and to the success of a company as a whole. There are many different ways to manage high-tech projects, especially software development projects. In reality, companies seldom adopt methodologies as they are defined in process vendor documentation. Rather, the selected methodology is customized to meet specific needs of a future project. Cases of

* This paper is based on research that was conducted by the author prior to joining Intel.

entirely custom processes being developed are also not uncommon. Process engineers need a way of capturing and reusing best practices from various processes. This is where process patterns [1], [2], [3] come into play. By definition, process patterns describe generic solutions to common software development management problems.

There are many process and organizational patterns documented and available for use. It is worthwhile to mention that descriptions of design patterns [5] traditionally include two main parts: a definition of the pattern model in UML [13] and a textual description that clarifies the use of the pattern. Unlike design patterns, process patterns usually do not define their models. A SPEM [11] – the specification approved by OMG in October 2002 – is specifically designed for software process modeling, and process engineering tools to be based on this specification are beginning to appear (for instance, Iris: http://www.osellus.com/products/iris.html). It is essential that such tools can benefit from formal definitions of process patterns by allowing them to be automatically applied to desired model elements the way like design patterns can be used in some CASE-tools.

Microsoft Solutions Framework (MSF) [8] is the software development process created and used within Microsoft. MSF consists of two models (process model and team model) and three disciplines (project management, risk management and readiness management) that address most of the problems and difficulties project groups face when creating a solution to meet customer expectations within budget and schedule constraints. MSF is a continually evolving product. The current version 3.0 utilizes a strong "agile flavor" that makes it highly customizable and adaptable to a wide range of software development projects. This also makes it difficult to write a formal detailed specification of MSF. MSF employs some interesting principles that once extracted from MSF can be used in other software processes to make them more efficient.

The MSF Team Model defines the roles and responsibilities of a team of peers and includes 6 role clusters that represent key projects stakeholders. The MSF Process Model defines a project life cycle that consists of 5 phases: Envisioning, Planning, Developing, Stabilizing and Deploying, each phase culminating in a milestone. Deeply aligned with ANSI PMI PMBOK [9], MSF Project Management Discipline relates the mentioned models and determines management practices. A six-step risk management process defined by the MSF Risk Management Discipline provides a comprehensive approach to project risk management. The MSF Readiness Management Discipline focuses on the readiness of project teams to successfully accomplish tasks while developing a solution.

2. THIS PAPER

In this paper, we show how SPEM can be used to define process patterns. Such definitions improve understanding of the mechanics of the solution provided by the pattern and can serve as a reuse unit for process engineering.

Our approach is the result of efforts to create a definition of the MSF in SPEM. Many enterprise-level software processes, including Rational Unified Process (RUP) [7], DMR Macroscope, IBM's Global Services Method, the Unisys QuadCycle method, etc. were modeled using SPEM [11]. RUP is supplied with a complete set of models that describes the process and an automated tool for process engineering Rational Process Workbench. This tool, together with the models, can be used to tailor RUP to a specific project. Our goal was to create a model of MSF that can be utilized in automated process engineering for projects that are going to use an instantiation of MSF. The initial modeling was done in Enterprise Architect (http://www.sparxsystems.com.au) equipped with a SPEM Profile. Our first model included more than 30 diagrams that described a project team structure and different disciplines within MSF.

Though creating such a model may seem an obvious task, it really was not. For instance, an MSF project team consists of only 6 roles. However, more than 10 SPEM ProcessRole and more than 20 SPEM ProcessPerformer entities were required to carefully describe the structure of an MSF project team. We used 3D diagrams several times in order to preserve the formal accuracy of the model and enhance clarity and understanding at the same time.

After carefully studying our models, we were surprised to see several diagrams shared a very similar structure. Subsequently, we came to process patterns that truly appeared as general solutions for certain problems that were reused within different contexts within MSF itself. Using the models we created as a foundation, we started one more research project aimed to:

- discover process patterns that are employed by MSF;
- validate our hypothesis that SPEM usage would simplify the pattern mining process as well as make pattern descriptions more formal and easy to understand.

This article presents the results of that project.

We decided to use one more well known process – eXtreme Programming -- to portray the MSF-based patterns that we discovered. Four of the major discovered patterns are discussed in this paper:

- Living Document;
- Smart Lifecycle;
- Clonable Lifecycle;
- Stakeholder-Oriented organization.

The template used to define process patterns is widely adopted [6] and tries to remain as close as possible to the template used for design patterns. We extended this by including a more formal definition of the structure of the pattern in SPEM. In [12] it was proposed to use UML to provide definitions of process patterns, but our experience shows that it is reasonable to use SPEM rather than plain UML. SPEM diagrams are naturally more concise, and thus more clear and understandable -- to the extent that we do not have to define properties of model elements that are specific to process engineering.

3. LIVING DOCUMENT

Intent

The intent of the Living Document is to provide the possibility to document preliminary decisions early in the development process, but also to allow these decisions to be changed as late as possible when more information becomes available to make a more appropriate decision.

Problem

When the process starts, some important decisions must be made, although not all of the required information is available. How can we ensure that these decisions can be refined, changed or justified using new information that becomes available?

Quite often, in order to make a foundation for further process phases and to provide input for subsequent steps, decisions are made and frozen very early, even though all of the information that influences the decision may not be available. Therefore a decision based on incomplete information is used as the basis for further software development process activities. A deficiency arises when a decision is documented, and, even though more information may become known, it is not (explicitly) expected to be changed.

Another extreme situation occurs when no decision is made until as late as possible. In this case, further process development is based only on assumptions. The actual final decision may be very different from what we expected it to be, and all the work that was done under the wrong assumptions will be lost.

Forces

A decision needs to be made and documented as early as possible to provide input for subsequent phases of the process. However, a decision should be finalized as late as possible when more information that influences the decision becomes available.

Further process activities need to be based on artifacts and these process activities cannot (or should not) be delayed. The authors of the artifacts tend to delay the distribution of the artifacts until all details are known because the amount of information available to them changes (increases) over time.

Solution

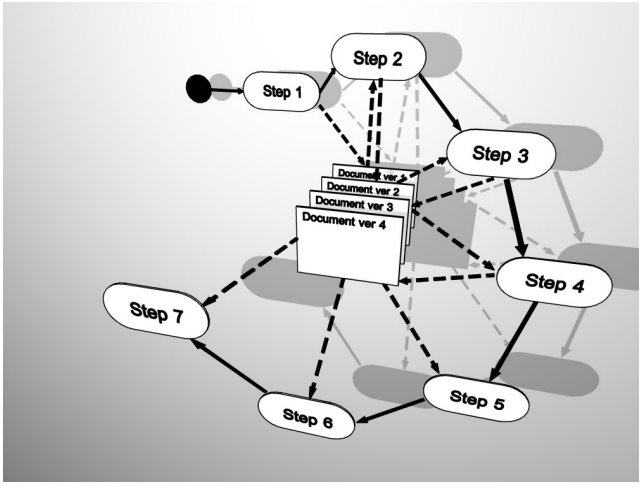

Figure 1. Living document 3D UML diagram

To explicitly define document change management procedures that allow documents to be changed (new version created) when more information become available. The first steps create an initial version of the document, which defines some preliminary decisions. Subsequent phases can use this document as input

but they can also update the document to reflect recent changes in the environment (see Figure 1). At a certain point (when changes can have significant impact on other parts of work) the document is "frozen" and making modifications requires compliance to stricter procedural steps (i.e. discussions in manifold review committees, sign-offs by senior managers, etc).

Stricter procedural steps provide reasonable input to phases that depend on such decisions and at the same time allow incorporating changes in the environment as long as possible.

The issue is that we should always be aware of the hanging nature of the living document and plan explicitly for its possible change. Although a decision may be documented, people who use the artifact should be aware that the documented decision may be refined/revised/changed, and this may require redoing some of the work that was based on the previous decision.

A "document" or "artifact" can be viewed in a very wide sense. A document or artifact can be anything that would serve as input or output of development activities – project documentation, product documentation, source code, test cases, etc. Applying this pattern compresses the project schedule by altering the finish-to-start dependencies between tasks (see Figure 2, a) to pairs of dependencies start-to-start and finish-to-finish (see Figure 2, b)

(a) *(b)*

Figure 2. Compressing schedule by applying the "Living document" pattern.

Consequences

Utilizing the Living Document process, a document is created that contains results of the previous phases of the process and can serve as a basis for our further work. Although the document may not be the final and the most perfect version, it is "good enough" to be used as an input to subsequent activities. We avoid standstills and the project is accelerated because the need to wait until the artifact is completely finalized is eliminated.

Changes in environment can be reflected in the document as late as it is possible. Therefore, we avoid the risk of finalizing important decisions before all the required information is available.

Examples

MSF heavily employs the Living Document pattern based on the principle "Baseline early, freeze late". For instance, the MSF Risk Management Discipline is built around a "living document" Master Risk List (see Figure 3).

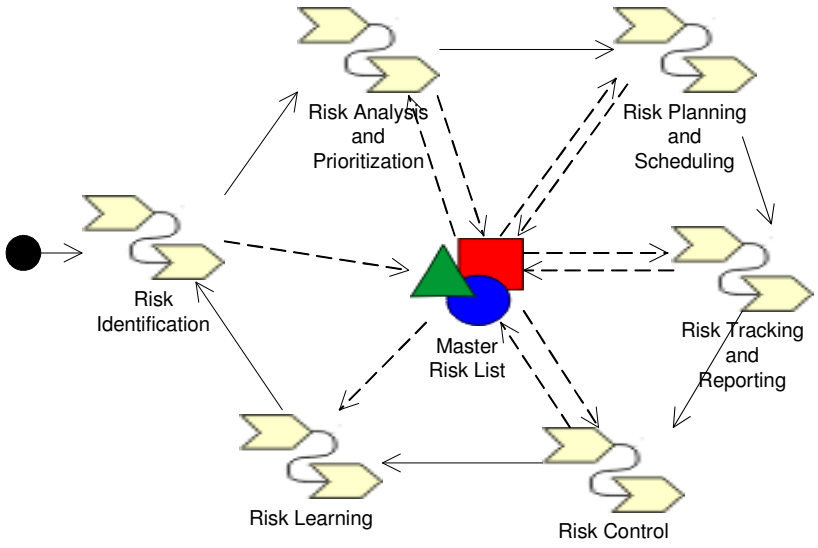

Figure 3. MSF Risk Management Discipline

The diagram shown in Figure 3 is actually a front projection of the 3D diagram that describes an instance of the pattern shown in Figure 1 for the MSF Risk Management Discipline. The diagram demonstrates how the Master Risk List is evolving as the risk management process proceeds.

The Master Project Plan within the MSF project lifecycle is treated almost in the same way (see Figure 4).

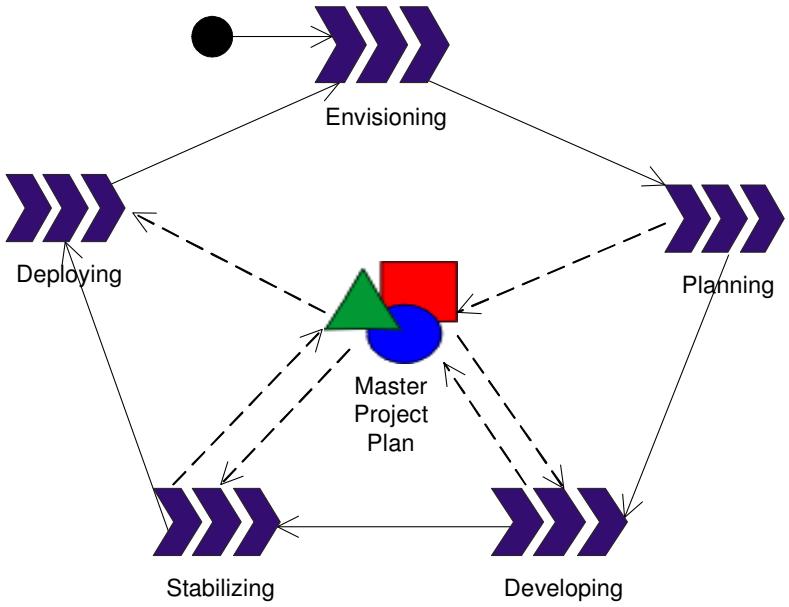

Figure 4. Planning in MSF

In fact, many software processes in one or another way use this pattern. In this pattern, a document undergoes several editions: an initial or baseline version is first created, and then updated during the next process steps.

Considering “document” in a wide sense, we can treat the unit test as a “living document”. We can, therefore, draw the eXtreme Programming daily development cycle organized around the unit test as a “living document” (see Figure 5).

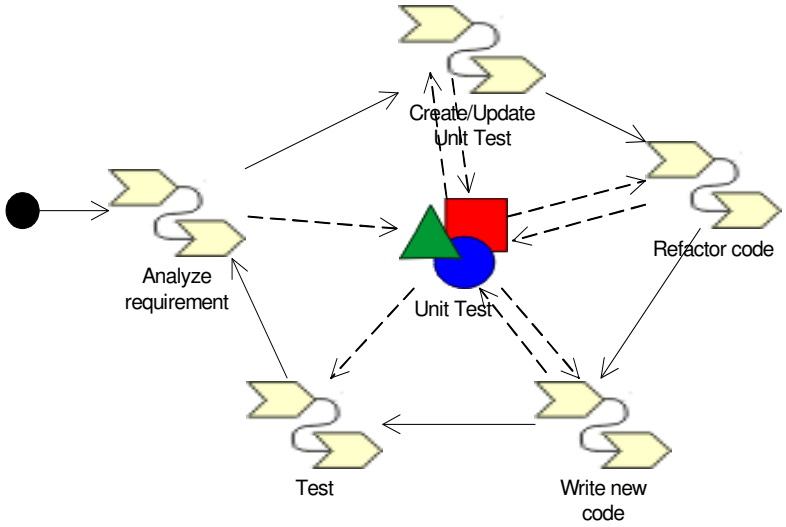

Figure 5. XP daily development cycle

There are many examples of this pattern in life around us. For instance, in almost all of the software development projects where authors have participated, a high level conceptual design was evolving as the development solution progressed. This design was baselined during earlier phases of the project; it was later clarified and refined as the team gained more knowledge about the problem domain.

On the other hand, instance of this pattern are not found only in software development projects. Some time ago, we participated in a courseware development project, where the course curriculum was a perfect example of a living document. Sometimes the lectures needed to be rearranged to provide students a better understanding of the materials; sometimes splitting the lectures was required. In all cases, the course curriculum was changed accordingly.

See also

Define Initial Management Documents [1]; Continuing Documentation [10].

4. SMART LIFECYCLE

Intent

The intent of the Smart Lifecycle is to provide the possibility of adapting the project to a new business situation throughout the entire project lifecycle, to designate the appropriate placement of decisions that influence the future processes of the project, and to make the process agile and adaptable but at the same time maintain its predictability.

Problem

Significant loss of money and time are the result of the inability to adapt the project to new business conditions, to change the future direction of the project, and to make important decisions that influence the entire project can lead to significant loss of money and time. Sometimes people tend to delay critical decisions, continuing development even when it becomes obvious that goals of the project will not be achieved.

Forces

Projects that follow strictly enforced and predetermined processes often fail to react to changes in the business environment and therefore may not produce the expected result.

Absolutely flexible processes that do not utilize even short-term planning are difficult to control and measure. These processes are, therefore, more susceptible to a varying array of project management challenges.

A project is considered successful if it delivers the required functionality on time and within budget. These three components need to be balanced during the project life cycle (this balance is often called "management triangle"). Violation of this balance usually renders the project unsuccessful.

However, when resources are already invested in a project, it is desirable to procure as much as possible from the project.

People usually avoid making negative decisions (like closing project, reducing the feature set etc.) on the project, thinking they can change things during later phases of the project lifecycle.

On the other hand, if a project is closed (or other negative actions are taken) at an inappropriate time, all possible deliverables may be lost. However, future projects may benefit from those results (like risk or domain analysis) – so the results need to be saved.

Solution

At the end of each phase, after a milestone had been achieved, an explicit “point of agility” is introduced to make decisions concerning the future of the project (see Figure 6). Key stakeholders should be involved in making these decisions. For instance, closing a project at the proper time can prevent further loss of resources. The decisions about the future of the project should be properly documented and all appropriate procedures should be performed. Synchronization of these decisions with the milestones is essential.

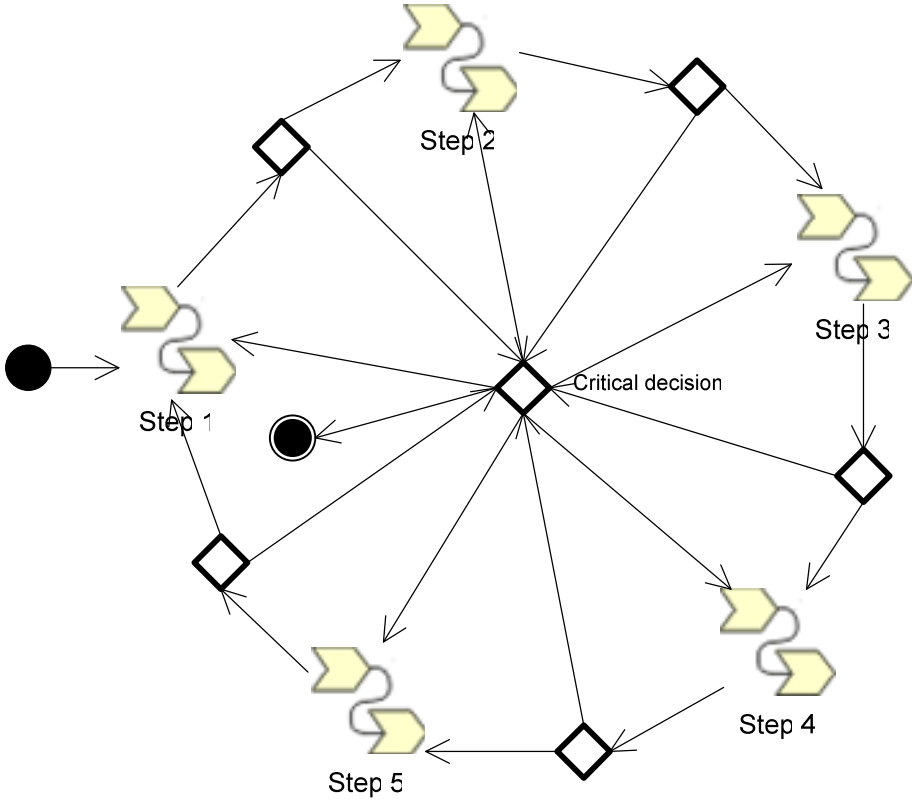

Figure 6. Smart Lifecycle process pattern

Consequences

Explicit decision points throughout the project life cycle ensure that, whenever a business situation changes, the project team can adequately respond to these changes.

Although the process is agile to some extent, it is controllable and measurable by means of short-term planning. After each intermediate decision is made, a process description and plan for the next step is created.

If there is strong evidence that the project is likely to fail, development can be stopped or other critical measures may be taken. The decision to discontinue the project is usually made after the initial phases.

The synchronization of critical decisions with major milestones ensures that everything that was created to that point can be saved by following procedures; this would become valuable input for future projects.

A clear policy about changing projects and planning future project steps is established through this synchronization.

External stakeholders may be confused by the fact that the project can be closed before completing the entire development process.

Examples

Many processes use this pattern in some form. Very often the main goal of the initial phase is to evaluate the feasibility of the project (called “Inception”, “Initiation” and so on). One of the deliverables of this phase is the decision to accept the project. If the decision is not to continue the project, then the analysis

and documents that were created are saved. Similar projects in the future would benefit greatly from those documents.

The MSF Process Model provides a great example of how the Smart Lifecycle pattern can be used (see Figure 7).

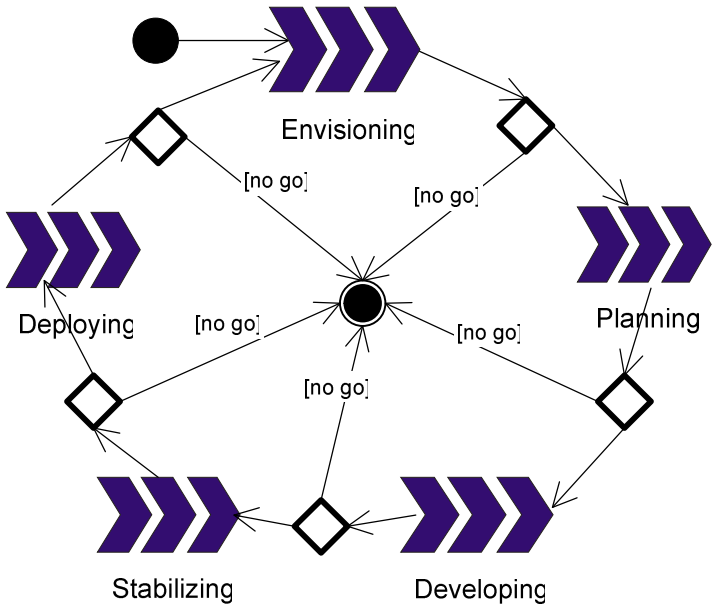

Figure 7. MSF Process Model

Normally a project is closed when the solution has been deployed and functions properly. When a project is not feasible, it is closed after the Envisioning phase. Unexpected events may cause the project to close after any of the other phases.

Many examples of this pattern can be seen in real-life projects. For instance, one of the authors was involved in a project that was initially estimated to be a one year effort in solution development. But after risks for the project were carefully analyzed it appeared that the project was too risky. This project was closed after the envisioning phase, which took approximately one month.

Another real-life example involves the development of a sophisticated HTML Help system for a large enterprise application. When the decision to develop such a system was made, there was no product on the marked with similar functionality. Development was scheduled for about six months, but when the scope of the solution was almost complete, a new version of Microsoft Help was released that provided many of the features the HTML Help system was meant to provide. After careful analysis, it was determined that the cost of developing and supporting the new HTML Help system outweighed the advantages the system would have provided when compared with the option of using the new Microsoft Help. This project was closed after the development phase.

See also

Iterative Development [1], [7], Milestone Based Development [8].

5. CLONABLE LIFECYCLE

Intent

The intent of a Clonable Lifecycle is to provide explicit parallelism for a process, especially when each process clone has its own independent set of artifacts.

Problem

The efficiency of development often depends on the ability to decouple tasks so they can be processed independently and simultaneously. At the same time, sequential processes have closely linked tasks that depend greatly on the results of previous activities.

Sometimes a process is performed on a set of the same or similar artifacts. Some of the artifacts become ready for transition to the next phase earlier than others. Much time can be wasted, however, if the first artifacts are delayed until the entire set of artifacts become ready.

Some (or even most) of the artifacts may be introduced after the process has already started. For example, a new risk may be discovered in the middle of the process of risk management, or a new requirement may be discovered in the middle of the process of requirement management.

Forces

During the process, new information becomes available that influences previous decisions and necessitates the revision of the previous decisions. But this information may only become available after the process has already proceeded to the point where certain types of activities are not permitted (in waterfall-like processes).

Some artifacts will be ready for transition to the next phase earlier than the others.

New artifacts of the same kind may appear later in the process. Such artifacts cannot be ignored and should go through all of the process phases.

Some artifacts may require returning to a certain phase of the process while going through the subsequent stages.

Solution

At any stage of the process, it is possible to “clone” a process and start a new one from the very beginning. Cloning allows for the parallel execution of an arbitrary number of processes that work on different pieces of development. For instance, a new process clone can be dedicated to the next release of the product (see Figure 8). Typically, forking a new process is guarded by some condition and is not considered a normal flow of process.

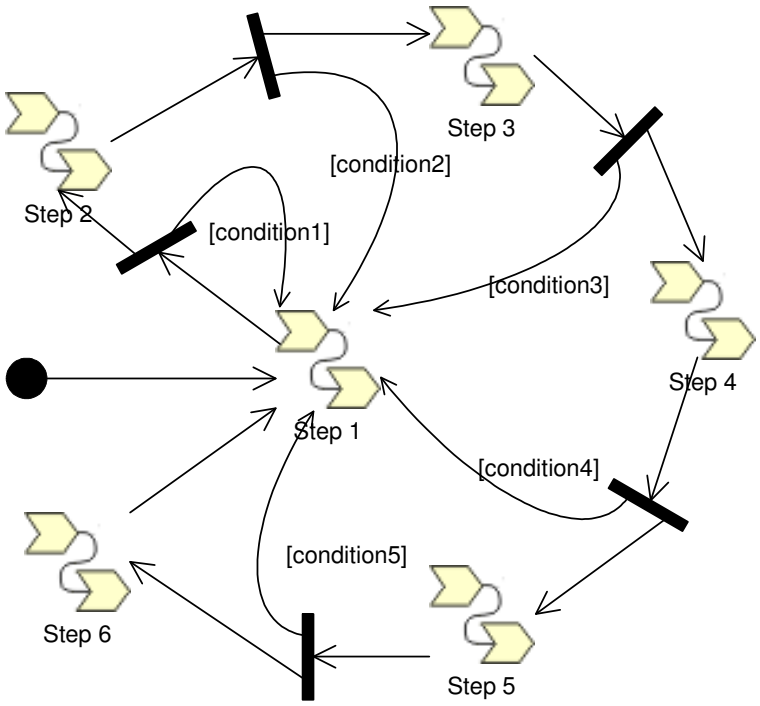

Figure 8. Clonable Lifecycle process pattern

In artifact-centered processes, each instance of a process is associated with a set of artifacts that is normally initialized during the first steps (see Figure 9). Because the need for a new artifact often occurs during the latter steps in the process, the process is cloned to start at the beginning. This is a more concrete and commonly used variation of the general form of the pattern. This variation is often utilized in artifact-centered processes in conjunction with the Living Document process pattern (see Section 3 LIVING DOCUMENT).

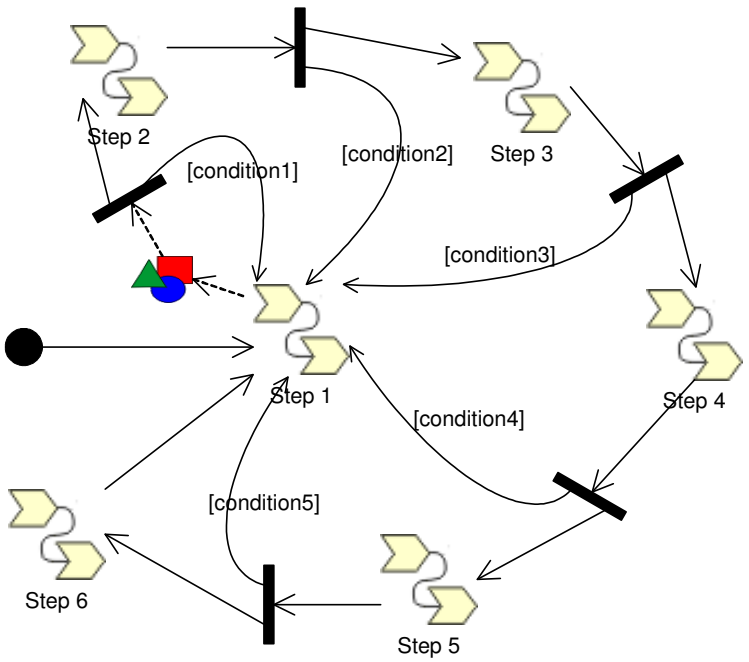

Figure 9. Clonable Lifecycle process pattern variation

As a new artifact appears, additional planning or rescheduling may be required to allocate resources for that artifact. Typically, the initial planning includes the possibility of the appearance of new artifacts. For instance, in MSF it is recommended to plan for future risks that may be uncovered during the project. In such a case, no schedule adjustment would be necessary.

When processing artifacts independently, it is difficult to track the progress of the entire process. Therefore, additional efforts are needed to ensure that all artifacts have gone through all process phases.

When new artifacts are of similar but substantially different natures, there may be no need for them to go through the same process. For new artifacts of lesser importance, a less stringent process may be applied, while critical artifacts must go through all of the prescribed process phases.

Consequences

New information that becomes available later in the process can be effectively incorporated. This incorporation may be done in two ways: an existing artifact may undergo the process again or a new artifact may be created.

The process is made more efficient because similar activities can be done independently and similar artifacts can be processed independently.

New artifacts can be easily incorporated into the process. A new process clone is initiated to deal with those new artifacts.

A new process is started whenever certain artifacts need to transition to the next phase earlier than others or to revisit the beginning phases of the process. The artifact is then processed in the new cloned process.

Examples

The parallel development of several versions of a product may be illustrated by incremental iterative development in MSF--even when the current version is in development we can start envisioning for the

next release. We can transition to the next iteration even though the current iteration was not yet completed (see Figure 10). On a large scale, cloning shortens the release cycle, increasing agility without loosing manageability.

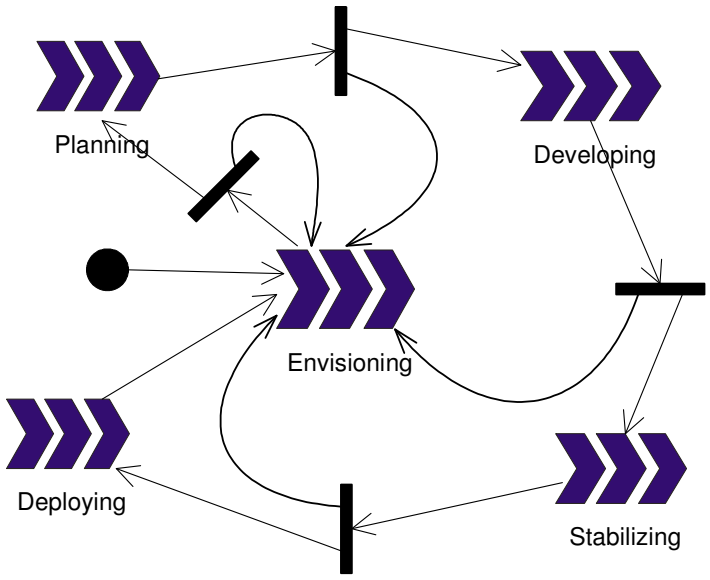

Figure 10. Incremental iterative development in MSF

The MSF Risk Management Discipline provides an example of how artifact-centered variation can be used to effectively manage risks in changing environments. Risks by nature are very dependent on external conditions; therefore the ability to reassess risk or to revise risk plans is very important (see Figure 11).

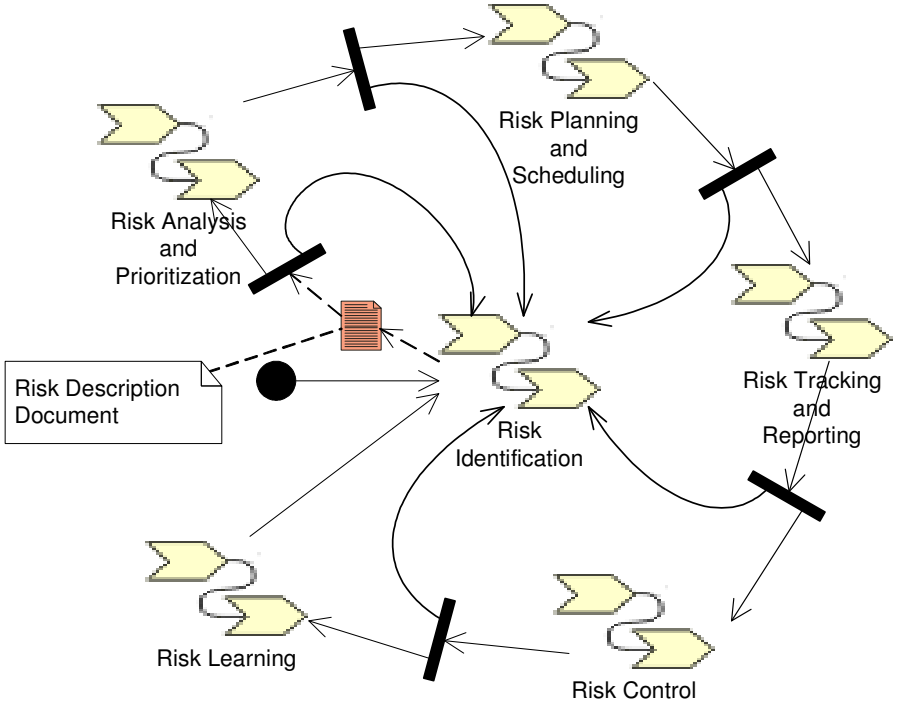

Figure 11. MSF Risk Management Discipline as Clonable Lifecycle

New risks may appear at any moment and must not be ignored. We would have to return to Risk Identification to restart the process for new risks.

The XP daily development cycle is organized in a similar way--to implement a requirement we may return to, for example, the Analysis step if necessary (see Figure 12).

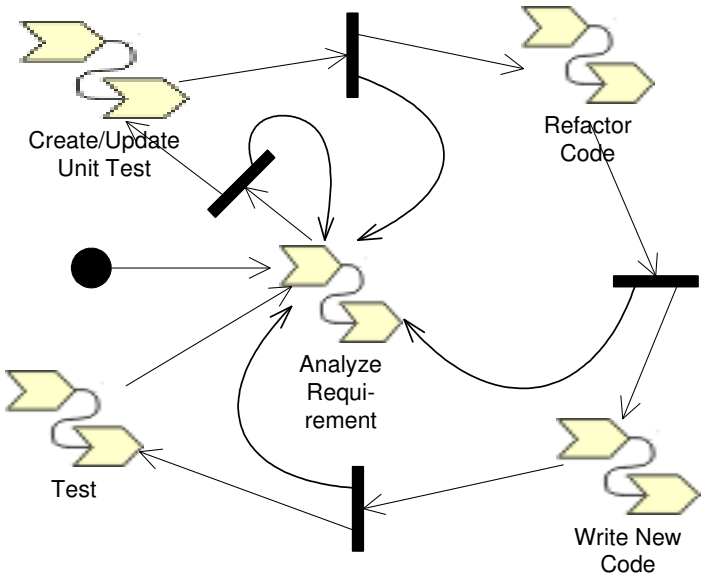

Figure 12. XP daily development cycle as Clonable Lifecycle

One of the authors was involved in a long term software development project that scheduled nine months for the development of one of the basic libraries. Several times during the development of the library, the need for additional modules was realized. All of the added modules underwent all phases of development--from envisioning and design to the actual implementation and testing.

As it was mentioned above, recently authors participated in one year courseware development project. The development process for learning modules was organized into several steps: planning, development, review, testing and integration into the course. During the project, it appeared that some of the learning modules contained too much information. These modules were divided into two or more modules. For all of these new modules, the development process started from the very beginning, while the main development continued to proceed at its own pace.

A defect tracking process is formally established in many organizations. The defect lifecycle includes steps such as evaluation, resolution and testing. Sometimes while resolving a defect, an engineer discovers that there are actually two defects that may be significantly different in nature – but they are hidden under one mixed description. The engineer may decide to split the mixed defect into separate defects that are tracked individually. In this case, the newly created defect undergoes all the steps defined by the process from the beginning.

See also

Spiral Development Cycle [1]; Incremental Development [1]; Iterative Development [7]

6. STAKEHOLDER-ORIENTED ORGANIZATION

Intent

The intent of the Stakeholder-Oriented Organization to ensure the interests of key project stakeholders are taken into account and balanced.

Problem

Naturally the different stakeholders have different and often conflicting interests in the project and its result. Overvaluing some aspects of the solution and omitting interests of other stakeholders may cause the entire project to fail. Stakeholder dissatisfaction may have a negative impact despite any success of the project.

Forces

The project has stakeholders that have different interests in the project and its results.

Different stakeholders have different viewpoints on the project development process and the overall project success.

Stakeholder dissatisfaction may negatively influence current or future projects.

If one person plays different roles, yet serves different interests, it is likely that one of those interests would be valued more than the others.

Solution

Determine all project stakeholders – people or organizations that have interest in the project. The team has role clusters that primarily focus on the interests of a particular stakeholder (or group of stakeholders). These clusters act as mediators between the team and the external stakeholders (i.e. stakeholders' advocates within the team – see Figure 13). All decisions are made with the consensus from all clusters to ensure all interests are taken into account and that there is no overvaluing or omitting of some stakeholders' interests over others.

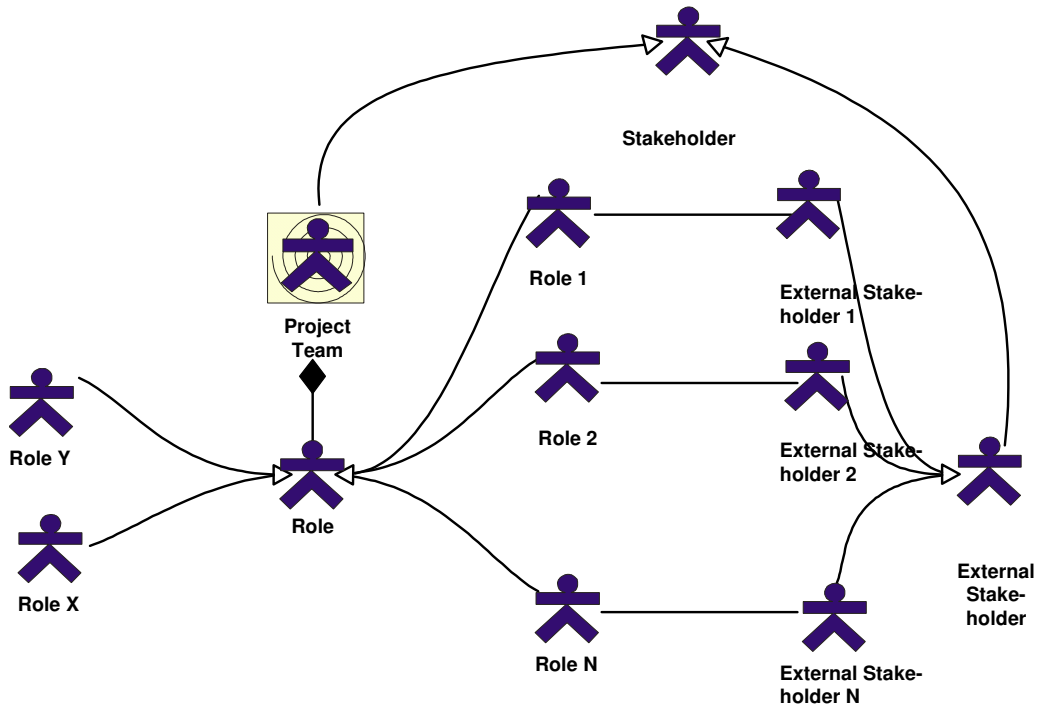

Figure 13. Stakeholder-Oriented Organization

It is not recommended to combine different roles, especially when they have conflicting interests. Participating in several clusters should be carefully considered when combining roles that serve opposite (and therefore conflicting) interests. This leads to the function of one role prevailing over the others. We do not say "may lead to" because practice shows that is always "leads". For instance, to address this issue MSF provides a role cluster compatibility matrix that shows which roles can and cannot be combined.

The team may also have internal clusters that do not correspond to any external stakeholders.

Consequences

The interests of all key stakeholders are taken into account.

Different team members are responsible for satisfying different stakeholders' interests, ensuring that none of them are overvalued. At the same time, the team as a whole strives to satisfy each stakeholder.

Since a consensus is required when making decisions, in some situations additional time may be required to find a solution that will satisfy all stakeholders and be accepted by all clusters.

Examples

The MSF Team Model is built on the Stakeholder-Oriented principle (see Figure 14). It also has two internal role clusters: Development and Testing. There are also four role clusters that correspond to the key external stakeholders of a typical IT-project. A cluster can consist of several employees, and an employee can be a member of several clusters. The primary goal of product management is to satisfy customers. User Experience is responsible for ensuring enhanced user productivity for the solution being built. Program management focuses on delivering the solution within budget and on time (so this cluster is responsible to satisfy the sponsor of the project). Development is responsible for building the solution

according to specification. Testing is concerned with the quality of the solution to be released. Release management ensures that the solution can be deployed and operated smoothly.

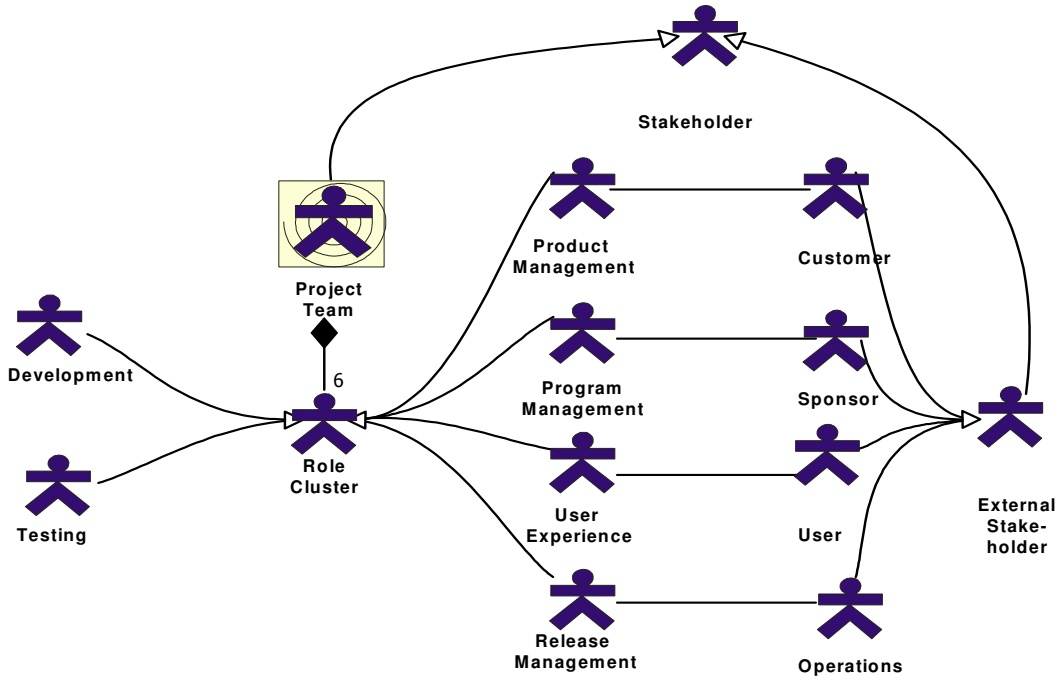

Figure 14. MSF Team Model

We were involved in a courseware development project in which 12 project team members were organized in 7 role clusters (Figure 15). The Coordination role cluster was responsible for managing the plans and the schedule and for the overall coordination of efforts. The Development role cluster primarily prepared the learning units, while Testing focused on ensuring the quality of the prepared materials. The goal of the teacher-care role cluster was to make materials usable and convenient for lecturers. The Student-care role cluster was responsible for ensuring that the materials prepared were acceptable to students. The Business-care role cluster ensured that students were taught and trained to be valuable employees in the future. The Institutionalization role cluster was to ensure that the materials adhered to educational standards and are eligible to be used in the learning process.

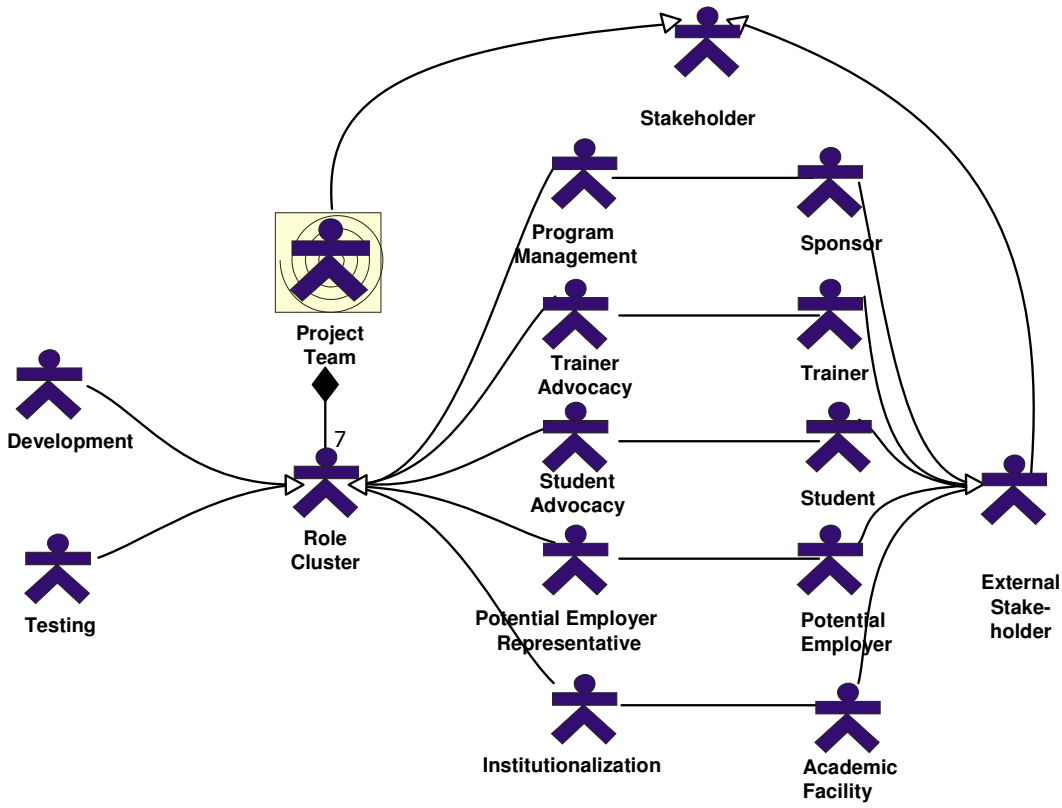

Figure 15. Courseware Development Project Team Model

See also

PMBOK Functional Organization, PMBOK Matrix Organization [9], Multidisciplinary Team [8].

7. CONCLUSIONS

In this paper we have described some process and organizational patterns we discovered while creating a SPEM model of MSF. They are not specific to a certain software development activity or process. They can be applied to any scale of software process: from high-level organization of process phases to arrangements of concrete tasks on the lowest level. The patterns discussed in this paper illustrate high-level concepts of the MSF, but at the same time they are utilized within XP. Often we can apply these patterns to improve the productivity and overall efficiency of the process without significantly altering the process. Therefore, we can easily leverage the experience of MSF in our specific process.

Using SPEM for documenting software development processes, as well as process patterns, helps a formal description of the process structure. This tool provides a better understanding of what occurs during development processes. SPEM-based models are concise and clear, so it is easier to capture similarities between different processes. This allows extracting patterns from processes that do not appear to have anything in common at first.

A formal SPEM-based description allows incorporating patterns in the form of wizards and/or add-ins into existing process engineering tools. Additionally, this makes it easier to understand the “internal mechanics” of patterns and provides groundwork for further research.

8. FURTHER WORK

Different software processes try to address the common problems of software development. Of course, they address problems in different ways. An interesting idea to explore would be to investigate the pattern-level similarities between different software processes; SPEM utilization will definitely facilitate this exploration. Obviously, only processes of the same level and maturity should be compared. It could be very productive, for example, to pattern-level compare two leading enterprise-level object-oriented processes -- Microsoft Solutions Framework [8] and Rational Unified Process [7].

Another concept worth investigating further is a 3D process modeling. Our experience shows that proper use of 3D process modeling can dramatically increase readability of diagrams and provides models that would otherwise be difficult to understand.

9. ACKNOWLEDGEMENTS

We would like to express a special thank you to our shepherd, Allan Kelly (Actix, Great Britain), for his valuable comments on our paper prior to VikingPLoP 2004. Allan's insightful reviews helped us gain a deeper understanding of our own ideas and contributed much to this paper.

We also would like to thank all people who reviewed different versions of this article (either in the form of article or presentation) and whose comments helped us polish our patterns:

- Alex Dubinsky (Dnepropetrovsk National University, Ukraine);
- Alex Zverintsev (Nokia, Poland);
- Andrey Terekhov (Microsoft, Russia);
- Bekah Sondregger (SCC Soft Computer, US);
- Dmitry Bednyak (Digital Road Studio, Ukraine);
- Dmitry Tsymbal (Novgorod State University, Russia);
- Evgeny Zakablukovsky (Intel, Russia);
- Juha Parssinen (VTT, Finland);
- Kevlin Henney (Curbralan, United Kingdom);
- Konstantin Runduev (Dnepropetrovsk National University, Ukraine);
- Kristian Elof Soerensen (The Danish Design Patterns Study Group, Danmark);
- Mariele Hagen (University of Leipzig, Germany);
- Neil Harrison (Avaya Communication & Hillside US, USA);
- Nikita Boyko (Dnepropetrovsk National University, Ukraine);
- Osama Mabrouk Khaled (Vodaphone Egypt & The American University in Cairo, Egypt);
- Pavel Hruby (Microsoft Business Solutions, Denmark);
- Stanislav Busygin (University of Florida, USA);
- Stanislav Petrovsky (ABV Technique, Ukraine);
- Stanislav Vonog (Moscow Institute of Physics and Technology, Russia);
- Valery Antonov (Petrozavodsk State University, Russia);

- Vivienne Suen (Osellus, Inc., Canada);
- Vladimir Rancic (University of Belgarde, Serbia);
- Yevgen Berlyand (Information Systems Development, Ukraine);
- ZEN-Process-Pattern-Team - M. Gnatz, F. Marschall, G. Popp, A. Rausch, M. Rodenberg-Ruiz, W. Schwerin, K. Bergner (University of Munich, Germany);

10. REFERENCES

[1] Ambler, S. W., *Process Patterns: Building Large-Scale Systems Using Object Technology*. New York: SIGS Books/Cambridge University Press, 1998.

[2] Ambler, S. W. *More Process Patterns: Delivering Large-Scale Systems Using Object Technology.* New York: SIGS Books/Cambridge University Press, 1998.

[3] Coplien, J. O. A Generative Development-Process Pattern. In Coplien and Schmidt [4], pages 183–238. Based on the proceedings of PLoP'94. Available also at `www1.bell-labs.com/user/cope/Patterns/Process/index.html`.

[4] Coplien, J. O., and Schmidt D. C., editors. *Pattern Languages of Program Design.* Addison-Wesley, 1995. Based on the proceedings of PLoP'94. Available also at `www1.bell-labs.com/user/cope/Patterns/Process/index.html`.

[5] Gamma, E., Helm, R., Johnson, R., Vlissides, J., *Design Patterns: Elements of Reusable Object-Oriented Software*. Reading: Addison-Wesley, 1995.

[6] Gnatz, M., Marschall, F., Popp, G., Rausch, A., Schwerin, W., Common Template for Software Development Process Patterns, 1st Workshop on Software Development Process Patterns (SDPP'02), `http://www.forsoft.de/zen/sdpp02/authors/template.pdf`.

[7] Jacobson, I., Booch, G., and Rumbaugh, J., *The Unified Software Development Process*. Addison-Wesley, 1999.

[8] Microsoft Solutions Framework, Microsoft, `http://www.microsoft.com/msf`.

[9] PMI Standards Committee. *A guide to the project management body of knowledge.* Project Management Institute, Newton Square, Pennsylvania. 2000.

[10] Rueping, A., *Agile Documentation : A Pattern Guide to Producing Lightweight Documents for Software Projects*. John Wiley & Sons, 2003.

[11] Software Process Engineering Meta-model, Object Management Group, `http://www.omg.org/technology/documents/formal/spem.htm`.

[12] Störrle, H., Describing Process Patterns with UML. In: *Software Process Technology*, LNCS 2077, Springer, 2001, pp. 173-181.

[13] Unified Modelling Language 1.5, Object Management Group, `http://www.omg.org/technology/documents/formal/uml.htm`.