Create Development Process On-the-Fly

Pavel Hruby
Navision a/s
Frydenlunds Allé 6
2950 Vedbaek, Denmark
E-mail: ph@navision.com

Acknowledgement
I would like to thank to Jutta Eckstein for shepherding this paper, and for her useful suggestions and comments, and Jens Coldewey for forwarding me the comments from the writers workshop. I do, of course, take full responsibility for any omission or errors.

Introduction and Terminology
This paper represents work in progress towards a pattern language addressing the topics of creating and adapting development processes. The intended user of this pattern language is a methodologist, that is, a member of a development team, whose interest is continuous improvement of software development processes. Depending on the organization, this role is also often fulfilled by quality assurance, managers and developers.

The term *process* is used to mean a comprehensive set of best practices that cover both static and dynamic aspects of software development. The term *development scenario* is used to mean the dynamic aspects of software development, such as a sequence of steps and activities during software projects. The term *artifact* covers static information about a software product, determined or created during software development processes.

General Context
You are using a standard software development process, such as the Rational Unified Process, Extreme Programming, Catalysis or Fusion. Software development processes describe sets of activities that result in certain specifications concerning the software product. These specifications can be very general, such as the vision of the product or a user story, or very concrete, such as the source code. Other examples are functional and unit tests, object collaborations and class descriptions. Software development processes also describe activities resulting in management products, such as project plans, organizational structures and responsibilities of team members.

You realize that the standard software development process can be successfully used as a general guideline, but it often does not match the needs of specific projects in all details.

General Forces

1. In order to improve a development process, an organization will generally either adopt and customize a standard development process, or develop its own process. The process consists of descriptions of activities to be performed and software development and management artifacts to be delivered by each project. However, it is not possible to determine a sequence of activities and a set of software development and management artifacts that fits all the projects, because the projects vary in size and complexity.
2. The organization wants to customize the development process at the beginning of each project. However, unexpected changes during projects prevent any up-front determination of which kind of customization (which software development and management artifacts) will be needed.
3. The team decides to use a standard development process and customize it on the fly during the project. This means disregarding some software development and management artifacts and possibly including new software development and management artifacts not specified by the process. However, the team does not know any simple rule for contracting or extending the process in a consistent manner.
4. The project repository contains a set of software development and management artifacts that reflects the project needs at each specific time, but which might be different from what a standard process requires. The team wants to get an overview of the completeness of the specification and the consistency between software development and management artifacts in the project repository. The standard process contains consistency assessment procedures, such as checklists, but they cannot be used, because the standard process was not the driving force behind the selection of the software development and management artifacts in the repository.

General Problem

Have you ever tried to apply a standard development process, and realized that you do not always perform all the activities in the specified order? Have your software development process sometimes forced you to create development artifacts you did not need? Have you sometimes want to specify something interesting about the system, but the process did not suggest any suitable software development and management artifact, diagram or document, for your specific information?

Have you ever wanted to know a simple rule for extending and contracting the process in a consistent manner?

Pattern Map

This pattern language consists of seven patterns. Fig. 1 shows the context of the pattern language including patterns not described in this paper. This paper focuses on the process analysis branch. It describes the introductory pattern, five process analysis patterns and one process improvement pattern.

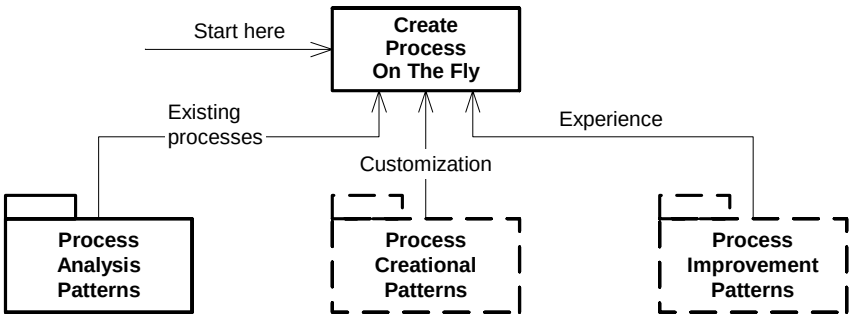

Fig. 1 Context of the pattern language

The pattern map is illustrated in Fig. 2. The rectangles represent patterns of the language and the arrows represent their resulting context.

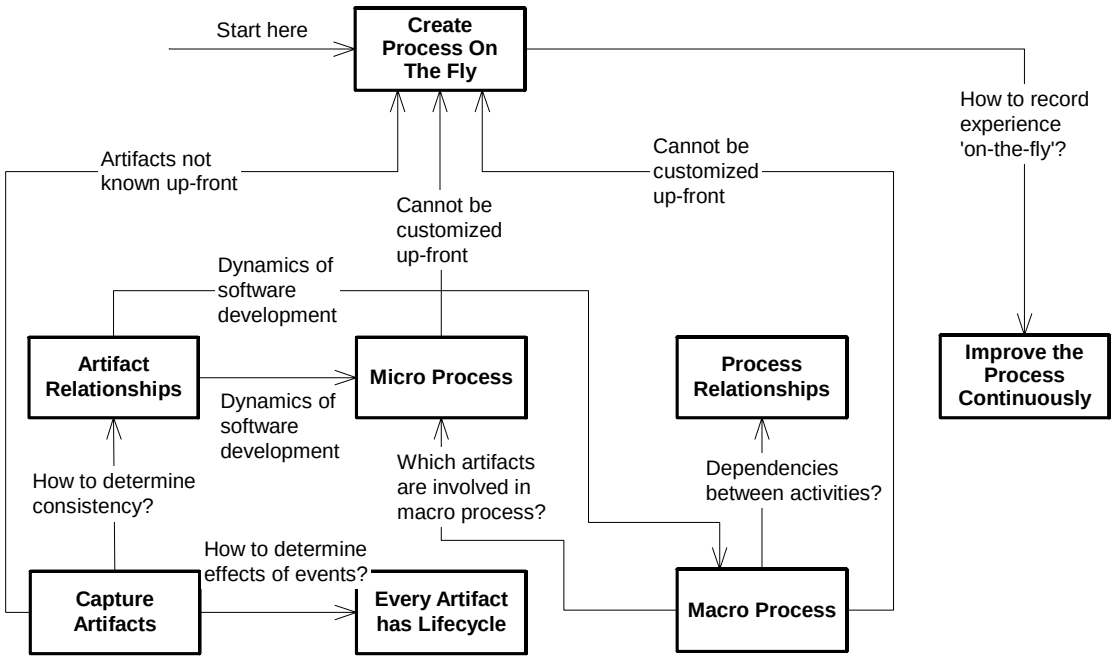

Fig. 2 Pattern map

Introductory Pattern: Create Development Process On-The-Fly

Context

You have identified development and management artifacts; see *Pattern 1, "Capture Artifacts"*. You have specified major process activities, see *Pattern 3, "Macro Process"*, and the scenarios and tasks, see *Pattern 4, "Micro Process"*.

Forces

1. You have specified the software development and management artifacts of your development process, but you realize that various projects and project situations require various subsets of the specified artifacts. Moreover, it is desirable to change form, representation, and other attributes of the artifacts in specific development contexts.
2. You have specified a development scenario, but you realize that no real project follows it on 100%. However, you still want to have a description of the development process, even though the concrete scenarios in each project are different.
3. You want to customize the scenario at the beginning of each project. However, unexpected changes during projects prevent any up-front determination of which kind of customization (which software development and management artifacts in which order) will be needed.

Problem

How to change the development scenario (the macro and micro processes) during the project?

Solution

Instead of specifying and following a firm development process, describe a process framework and customize it to fit each specific development problem. According to *Pattern 2," Artifact Relationships"*, define a process by specifying dependencies between software development and management artifacts, rather than by specifying a concrete scenario. Let project participants pick up the development and management artifacts, as they need it.

Fig. 3 illustrates how to use the framework and "create a development process on-the-fly". During software development, create (instantiate) the appropriate software development and management artifacts on demand, depending on the problem and the concrete situation. Apply *Pattern 1, "Capture Artifacts"*, and use the purpose properties to select those software development and management artifacts that you need in a particular situation in order to move forward. The quality-assurance activities will guarantee consistency between the software development and management artifacts. The constructor activities might have preconditions that require that other software development and management artifacts must exist before other software development and management artifacts are created. Preconditions therefore indirectly imply an order in which the software development and management artifacts may be created. For example, in order to create the *class lifecycle*, the software development and management artifact *class* must be created first. However, the order required by preconditions of constructors and quality-assurance activities is much less restrictive than a specification given by traditional design processes.

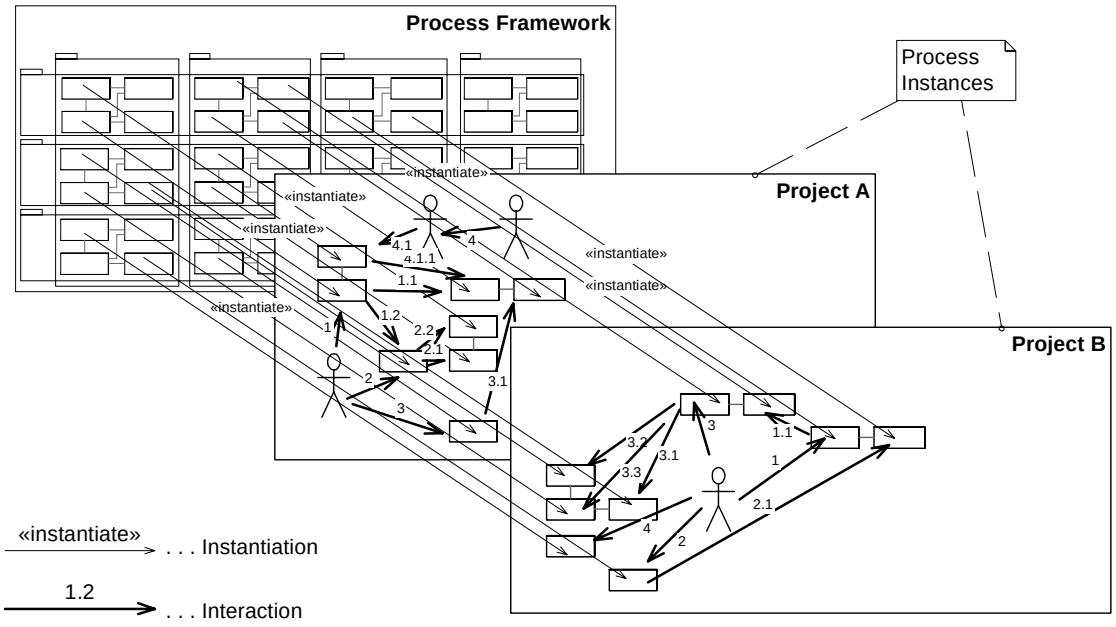

Fig. 3 Creating specific development processes on-the-fly by instantiating the framework

Known Uses

OPEN [5] is an object-oriented framework for software development processes. OPEN provides descriptions of a range of activities, tasks and techniques, which can be tailored specifically to each individual organization or individual project. OPEN has a well defined metamodel, consisting of projects, activities, tasks, deliverables, techniques and sequencing rules.

The OPEN framework can create specific processes by choosing the process components that meet specific demands in specific situations. A concrete process based on OPEN is created by selecting the activities and tasks necessary to perform the project. The execution of the activities is guarded by pre- and postconditions on tasks. The postconditions include testing requirements and deliverables.

Extreme Programming [1] defines three management and four development artifacts. The management artifacts are the release plan, task and iteration plan, and the development artifacts are the user story, unit test and acceptance test and code. Within each iteration, extreme programming does not define any order in which development artifacts are created or updated – it lets developers select them according to the specific situation. It does, however, specify the dependencies between management and development artifacts. The iteration plan and user stories, selected for development, do not change within the iteration.

Resulting Context

We have the problem how to record experience and improve the process ‘on-the-fly’. This problem is resolved in *Pattern 5, Improve Process Continuously.*

Pattern 1: Capture Software Development and Management Artifacts

Context

Various software development processes reflect “best practices” of various aspects of software development. You want to study and use the “best practices” from various processes in the light of your own experience. Various software development processes are described in various styles, which makes it difficult to compare them.

Forces

1. You want to compare various software development processes, but they are described in different and mutually inconsistent ways.
2. You want to select “best practices” that reflect your specific needs in your current situation. It is difficult for you to find and categorize the “best practices”, which are common across various projects.
3. You realize that the order of development steps is often different in different processes, but you want to capture the information, which is common across various processes.

Problem

What to focus on when comparing various software development processes?

Solution

Concentrate on work results rather than chronological order and sequencing of activities. For example, the primary deliverables of (almost) all software projects are code and tests. Many projects create, in addition, a project plan, and requirements specifications of various kinds.

Concentrate on what kind of information you want to specify, rather than how this information is represented. In other words, make a distinction between the software development and management artifact (the information itself) and the representation of the information (the UML diagram, CRC-card, table, text).

For each software development and management artifact identify constructor (a process or activity how to create the artifact), quality assurance (a test, procedure, activity or check-list determining the quality of the artifact, its completeness and consistency of the artifact with other artifacts), information specific to artifact type, such as typical representation, and information specific to artifact instance, such as the concrete representation of the artifact (text, UML diagram, table).

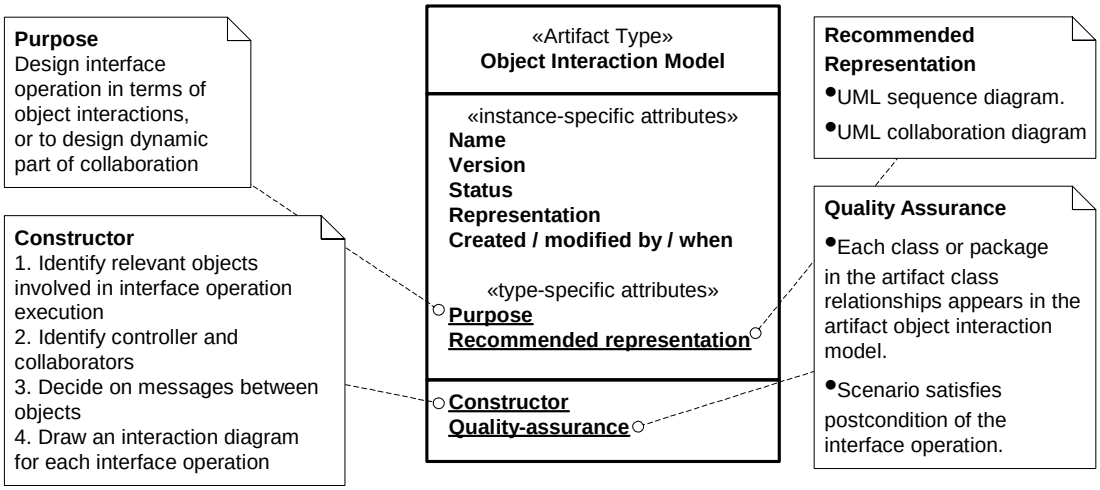

Fig.4. Specification of the artifact type object interaction model

Known Uses

The Rational Unified Process [6], for most of the artifacts, specifies guidelines, purpose, properties, timing, responsibility and for some artifacts also tailoring. The "guidelines", "timing" and "tailoring" cover information about how to create the artifact and the quality assurance. The "purpose", and "responsibility" are artifact type properties, the "properties" are the artifact instance properties.

The Microsoft Solutions Framework [7] provides a "template" and "checklist" for most artifacts, that is, constructor and quality assurance. The Microsoft Solutions Framework specifies the "purpose" and the "owner" of the artifact, which are artifact type properties. The templates contain fields for artifact instance properties.

Extreme programming has seven well-defined development and management artifacts: user story, release plan, acceptance test, iteration plan, task, unit test and code. The constructors and quality assurance methods are expressed as extreme programming practices, such as metaphor, small releases, simple design, pair programming, collective code ownership and coding standards.

Resulting Context

You specified which development and management artifacts to create during software process, how to create them and how to assess their quality. However, development artifacts are often related: they depend on each other, refine each other, or might just refer to each other. Which relationships exist between software development and management artifacts? This context is resolved in *Pattern 2, Artifact Relationships*.

Pattern 2: Artifact Relationships

Context

You identified a number of development and management artifacts, which describe the information about software or management products. This information could be very general, such as the vision of the product, or very concrete, such as the source code and test scripts. Other examples are use cases, object collaborations, class descriptions and test cases. You also specified information about management products, such as projects, project plans, organizational structures and job descriptions. All these artifacts were specified by various people, and it becomes difficult to get an overview and navigate through the repository.

Forces

1. You identified a number of development and management artifacts, but it becomes difficult to overview them, and find a specific artifact in a project repository.
2. You would use a standard development process for structuring the artifacts, but the artifacts that reflect your needs and experience are different from the artifacts of the standard process.
3. You cannot anticipate which kinds of artifacts will be added to the repository in the future, but you'd still like to maintain a succinct structure of the repository.

Problem

Do we have all the artifacts we need?

Solution

Do not concentrate on how the artifacts are represented (such as how they are represented in the UML), but concentrate on what kind of system information they describe. Identify views and viewpoints relevant to your concerns. Examples of the development viewpoints are the logical viewpoint, the use case viewpoint, the component viewpoint, the testing viewpoint and the design patterns viewpoint. Examples of the management viewpoints are the project viewpoint and the team viewpoint.

Each view contains development and management artifacts – work products – of the particular concern at various levels of granularity. The levels of granularity refine each other; see Fig. 5.

Within the scope of a viewpoint and the level of granularity, four development and management artifact types can be identified. The artifact called *work product relationships* specifies static relationships between work products. Examples of this artifact are the class model, use case model and database schema. The artifact called *work product interactions* specifies interactions between work products. Examples are any information represented by UML sequence diagram or collaboration diagram, such as object interactions. Examples of the *work product* are the code module, CRC card, test, use case, project and team. The *work product lifecycle* specifies dynamic properties of the work product. Examples are any information represented by UML statechart and activity diagram, such as the allowable order of class interface operations, or a status of a project.

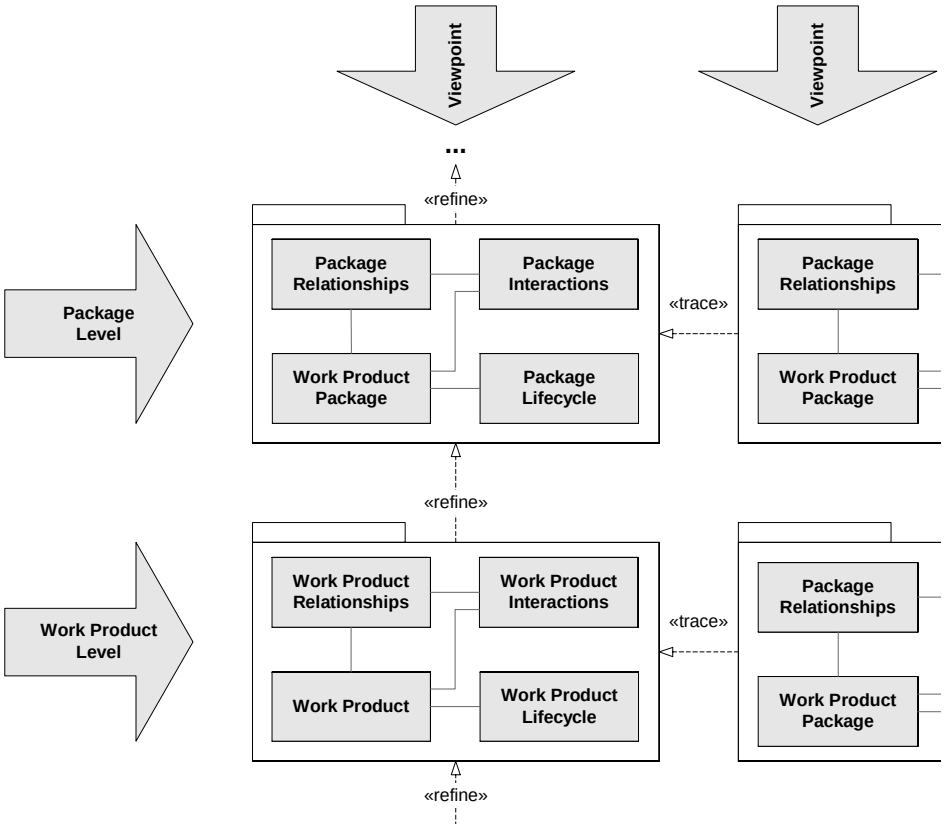

Fig.5. Relationships between artifacts describing work products

Known Uses

In the UML [8], examples of 'work product relationships' are the class diagram and the use case diagram; examples of 'work product interactions' are the sequence diagram and the collaboration diagram; examples of 'work product' are the class and use case; examples of 'work product lifecycle' is the statechart and activity diagram. Example of work product package is a subsystem; example of package relationships is a class diagram containing subsystems.

In software testing, examples of 'work product relationships' are the test dependencies; examples of 'work product interactions' are the test sequences; example of 'work product' is the test case; and example of 'work product lifecycle' is the test algorithm. Example of work product package is a test suite.

In project management, an example of 'work product relationships' is the PERT chart (a chart showing dependencies between tasks or activities); an example of 'work product interactions' is the Gantt chart (showing starts and ends of tasks); examples of 'work product' are the project and task; examples of 'work product lifecycle' are the project states (such as ahead, delayed, canceled).

The Catalysis method [9] structures development artifacts into various levels of granularity that refine each other. The Rational Unified Process [6] structures development artifacts into four views.

Resulting Context

You have described which development and management artifacts and the relationships between them. However, this information depicts static features of a development process. For example, the artifacts do not specify which of them to create at the beginning of the project and what is the final deliverable. You still have a problem of identifying the dynamics of the software development process. This problem is resolved in two patterns, *Pattern 3, The Macro Process* and *Pattern 4, The Micro Process.*

Pattern 3: The Macro Process[1]

Context

You realize that in each project you perform certain activities, such as design, coding, and testing. You perform these activities and create and update software development and management artifacts in various orders. Some of these orders are not ad-hoc, but they often repeat in the same sequence.

Forces

1. Regular releases drive the development process to repeat certain activities in each development cycle. You want to describe this experience and you need a suitable "placeholder" for such information.
2. Some development scenarios have the same purpose, but they vary in concrete steps because of different project situations. Some development scenarios vary in concrete steps because of their different goals. You want to abstract from concrete project situations and identify repeatable information that the development scenarios have in common.

Problem

How to identify the rhythm of the software development process?

Solution

Identify major phases (scenario types) of the software development process. Identify the purpose, (reason, motivation) of each scenario type. In addition, specify the goal, preconditions, postconditions, as well as the list of software development and management artifacts accessed and modified by the scenario type.

Identify activities to be performed by each scenario type. Example of such activities is illustrated in Fig. 6.

[1] The terms macro and micro process are inspired by Object Solutions by Grady Booch, Addison Wesley, 1996.

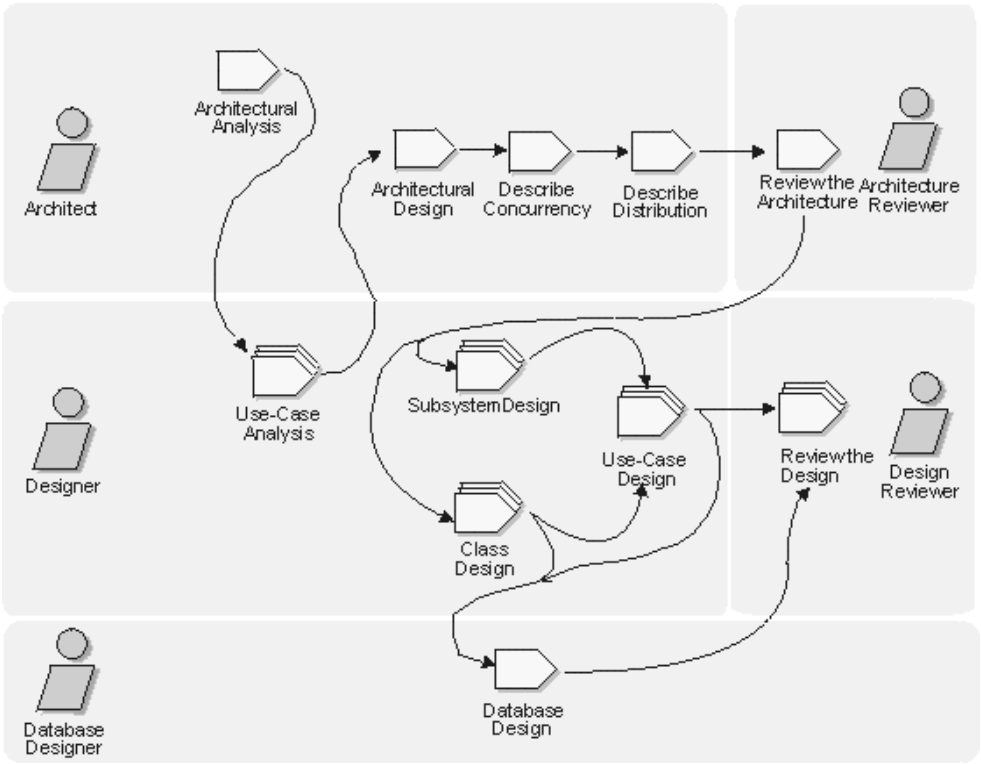

Fig.6. Activities of the Analysis and Design Workflow of the Rational Unified Process

Known Uses

In the Rational Unified Process [6], the scenario types are called "workflows". The Rational Unified Process describes six core and three supporting workflows. The core workflows are the business modeling, requirements, analysis & design, implementation, test, and deployment. The supporting workflows are the configuration & change management, project management, and environment. In addition, the Rational Unified Process specifies four iteration workflows: define the system's vision and scope, outline and clarify the system's functionality, consider the project's feasibility and outline the project plan, and refine the project plan.

The Catalysis [9] process calls the scenario types "process patterns". The Catalysis describes four main process patterns: object development from scratch, reengineering, short-cycle development, and parallel work. In addition, Catalysis describes a number of specialized patterns, such as business process improvement and creating a common business model.

In the Microsoft Solutions Framework [7], the scenario types are called envisioning, planning, developing and stabilizing.

In the OPEN process specification [5], the scenario types are called "activities". OPEN describes 13 activities: project initiation, requirement engineering, analysis and model refinement, project planning, build (with the subactivities evolutionary development, user review and consolidation), evaluation, implementation planning, program planning, resource planning, domain modeling, other projects, use of system, and bug fixing.

Resulting Context

The solution of this pattern depicts the development scenario in term of activities, but it does not show which development artifacts are created, modified or accessed within each activity. *Pattern 4, The Micro Process*, addresses this problem.

Pattern 4: The Micro Process

Context

You have identified the development and management artifacts and the relationships between them.

Forces

1. The development artifacts do not imply any order in which they should be created or updated. However, they are often created or updated in the same or similar sequence.
2. You want to capture dynamic aspects of the software development process, such as a sequence of activities and tasks.
3. The macro process pattern captures the rhythm of the process, however, it does not specify which development and management artifacts are created, modified and accessed within each scenario type.

Problem

What development artifacts are created, modified, or accessed within each activity?

Solution

Consider evolution as interactions between the management and software development artifacts and the users of the development process. Describe scenario instances as interactions between development artifacts and users of the development process. This approach is illustrated in Fig. 7.

As an example, Fig. 7 shows the extreme programming process [1] described in terms of interactions between team members and development and management artifacts. Extreme programming recognizes two major roles in a development team: customers and developers. Customers create and prioritize the user stories and together with developers create the release plan. Developers estimate the user stories; create the iteration plan consisting of estimated tasks. Developers create the unit tests and code. Customers design the acceptance tests. The figure shows one specific instance of the process.

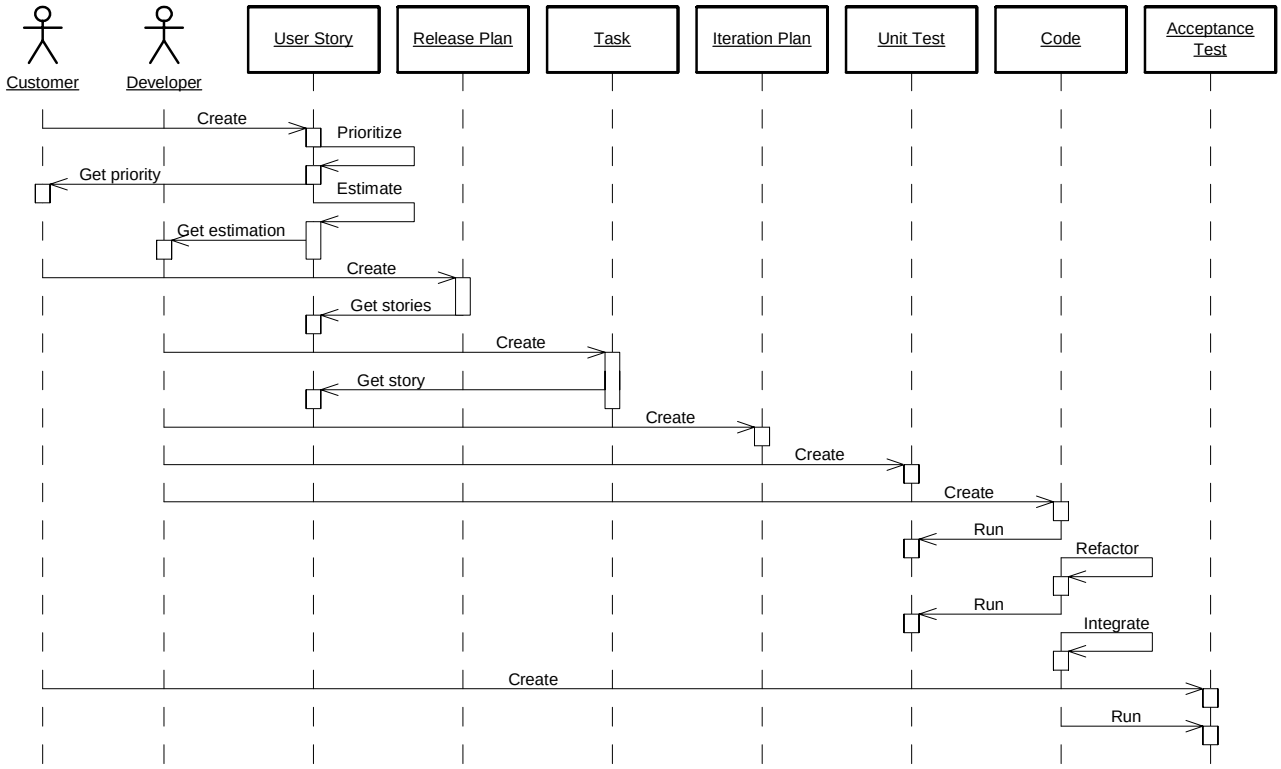

Fig.7. An instance of one iteration of the Extreme Programming Process.

Known Uses

The Fusion process [4] describes the development scenario as a sequence of creating development artifacts in a specific order. Using a requirements document as an input, the Fusion process creates an object model and interface model, which result to object interaction graphs, visibility graphs, inheritance graphs, class descriptions and program.

Microsoft Solutions Framework (MSF) [7] is a set of guidelines for developing client-server systems. MSF is not as well known as Fusion, OPEN and Rational Unified Process, however, it is a highly evolved product that defines a simple and efficient team model, a process model and an application model combined in a common framework. The MSF process defines four milestones, vision and scope approved, project plan approved, scope complete / first use, and the release milestone. The scenario within each milestone is described in terms of development artifacts to be completed. For example, the vision and scope approved milestone delivers the problem statement, vision statement, user profile, solution concept, project structure and risk assessment. The project plan approved milestone delivers the functional specification, master schedule, master plan and the updated risk assessment.

Resulting Context

The solution of this pattern depicts the development scenario in terms of development artifacts created, accessed and modified within each scenario type. Because this sequence is a concrete example of a development scenario, no real projects fully follow it to all details. Therefore, we have

a problem how to customize it. *The Introductory Pattern, Create Development Process On-the-Fly*, resolves this problem.

Pattern 5: Improve the Process Continuously[2]

Context

You have identified software development and management artifacts and relationships between them. You participate in a post-mortem meeting at the end of each project.

Forces

1.You want to record suggestions to process improvements *during* the project. But post-mortem meetings are typically created at the project end, if ever.
2. Post-mortem minutes of meetings have various ad-hoc structures and searching for and finding a specific piece of information is not easy. It takes a long time to find whether the post-mortem minutes of meetings contain any information useful to another team.
3. It is difficult to extract patterns and similar solutions to often occurring problems from post-mortem meeting documentation. It is difficult to evaluate and improve the quality of the development processes.
4. Post mortem will never help to improve the process of the current project. It will perhaps help to improve the process used in the next project, although this would probably be completely different.

Problem

How to capture knowledge gained during software development.

Solution

Record your experience by updating the elements of the framework: the software development and management artifacts and scenario types. Update the properties, constructor and quality assurance activities of existing artifacts. Create new software development and management artifacts as you need them, simply by defining their properties, constructors and quality-assurance activities.

[2] This pattern represents work in progress

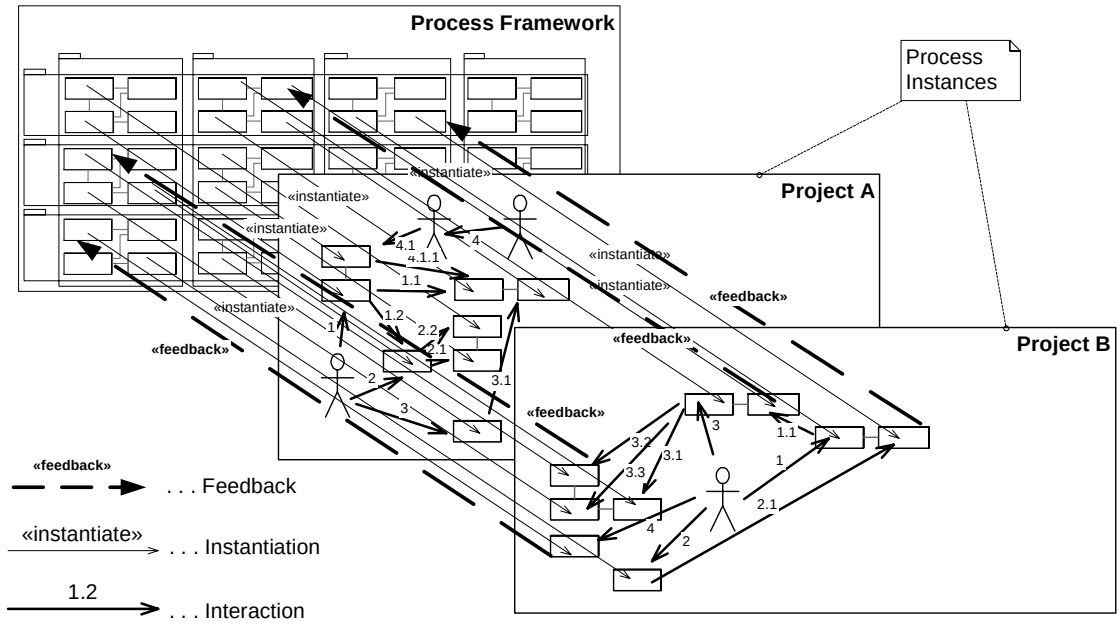

Fig. 8 Continuous improvement

Known Uses

Adaptive Software Development [10] organizes end-of-cycle mini post-mortems to force the organization to learn during the project. The postmortems focus on four basic questions: what is working, what is not working, what do we need to do more of, and what do we need to do less of. Although the end-of-cycle reviews are not “continuous”, they illustrate increased frequency of the feedback on the development processes.

Capability Maturity Model (CMM) [11] defines a set of quality criteria and key practices that are used to classify software processes into five maturity levels, based on organization’s support for key process areas. Level 5 is characterized by continuous process improvement. The CMM level 5 organization maintains plans for process improvement, has a written policy for implementing software process improvements, the training in process improvement is required for both management and technical staff and improvement effort is periodically improved. Unfortunately, author is not familiar with any reference to a software organization that has reached CMM level 5. The Capability Maturity Model at least shows that continuous improvement has been described.

References

[1] Beck, K.: Extreme Programming Explained; Addison-Wesley, 1999.
[2] Booch, G.: Object Solutions, Managing the Object-Oriented Project, Addison-Wesley, 1996.
[3] Cockburn, A.: Using "V-W" Staging to Clarify Spiral Development, available at: http://members.aol.com/acockburn/papers/vwstage.htm
[4] Coleman, D. Arnold, P. Bodoff, S. Dollin, C. Gilchrist, H. Hayes, F., Jeremaes ,P.: Object-Oriented Development: the Fusion method, Prentice Hall, 1994

[5] Graham I., Henderson-Sellers B., Younessi H.: The OPEN Process Specification, Addison-Wesley, New York, 1997
[6] Kruchten, P.: The Rational Unified Process, Addison-Wesley, 1998.
[7] Microsoft Solutions Framework 2.0, Microsoft, 1997.
[8] The Unified Modeling Language, version 1.4, OMG, www.omg.org
[9] D'Souza, D., Wills, A.: Objects, Components and Frameworks with UML: the Catalysis Approach, Addison.Wesley, 1998.
[10] Highsmith, J.: Adaptive Software Development: A Collaborative Approach to Managing Complex Systems, Dorset House, 2000.
[11] Paulk M. C. et al.: Capability Maturity Model for Software version 1.1, CMU/SEI-93-TR-024