The Object-Oriented Model for a Development Process

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

Abstract

Have you tried to describe your development process based on workflow and later found it difficult to meet demands for customizations, quality and usability in general? It might help to consider deliverables objects and evolution as object interactions. We will discuss our experience with such a process definition with an eye towards approaches such as Fusion, Objectory, OPEN Process Specification, Microsoft Solutions Framework and Capability Maturity Model.

Basic Features of the Object-Oriented Process Model

Deliverables produced during software development are considered objects with various methods and attributes. Deliverables have *constructors*, which are methods describing how to create a deliverable, and *quality-assurance methods*, such as completeness and consistency checks. Deliverables have numerous *attributes*: name, kind, description, references to other deliverables, project, subsystem, increment identification, responsible developer and other attributes such as who created and modified the deliverable and when.

The attributes *kind* and *name* taken together are the key that uniquely identifies the deliverable in the data dictionary. The attribute *description* typically contains a UML diagram, a table or a text. The choice of a suitable representation is left up to the judgement of the developer and depends on the specific situation. For example, a use-case model can be represented by a use-case diagram, a list of use-cases (text), use-case schemes, text describing sample scenarios, and so on. A component interaction model can be represented by one or more sequence diagrams, collaboration diagrams, state diagrams, activity diagrams, in Backus-Naur form, and so on.

Almost all other attributes can contain a number of values so that a deliverable can be related to several other deliverables and can be reused within different projects, increments or components.

Concrete *classes* of deliverables define methods of the deliverable and can include other specific attributes. Examples of deliverable classes are task context (see below), use-case model, class model, component interaction model, data types, note, and so on. The name of the deliverable class is specified in the *kind* attribute.

Each increment (a small piece of functionality added to the existing product) is defined by a single deliverable called a *task context* document, which corresponds to a *task* in Microsoft Project. The task context document contains general information about the increment, such as a synopsis, requirements, metrics (time estimates), the developers responsible, and so on. The task context document exists throughout the entire life cycle of the increment (from requirement analysis to implementation and testing). The development phase of the increment is represented as a value of one of the attributes of the task context document.

Experience with the Object-Oriented Process Model

We at Navision Software have developed tools that are helpful for creating and organizing project deliverables. Our solution was based on two Lotus Notes databases: one was used as a data dictionary, and one was used as a project management database. The data dictionary contained both terminology definitions and design deliverables. Lotus Notes documents can have fields that contain OLE objects, for example, diagrams created in Visio or ABC Flow Charter. They can also have links to other documents in the same or other databases. Documents can be organized (viewed) using different sorting and selection criteria, for example by kind of entry, by project, by task, by subsystem, by person responsible, and so on. The main benefit of using these tools was flexibility – in notation, in the kinds of deliverables in the repository and in the possibilities provided for modifying the process according to the size and character of the task.

The benefits of the object-oriented definition of the software development process are:

- It can manage the complexity of the development process in a better way than the workflow model. As a result, the final process description is more transparent and easier to modify or customize.
- It is robust and easy to use. Small increments typically result in a small subset of deliverable classes: task context, plan, source code, user documentation and perhaps several design documents. The quality-assurance methods guarantee consistency between deliverables. With larger increments, the number of kinds of deliverables can be increased and always reflects project state and any specific requirements.
- The process provides good management support for incremental development. Each increment is defined in a single document that exists throughout the increment's life cycle.
- The model is flexible and applicable to processes with a wide range of characteristics. By simply defining or redefining the methods and attributes of the deliverable classes, the model can be easily adapted for use with different kinds of development processes.

Other Models for Development Processes

Most of the currently used models for development processes are based on the workflow model, or on the object-oriented model in which activities are objects, tasks are object operations and deliverables are operation postconditions.

The workflow model is defined with a graph of activities, tasks and deliverables. In general, such a definition cannot cover all the possible combinations of activities and deliverables without becoming overly complex. A model (whether object-oriented, or not) can be deliverable-based and focused on which deliverables are produced or activity-based and focused on which activities or tasks are performed within each development phase. The deliverable-based model has several advantages over the activity-based model: it is usually easier to ensure the quality of a deliverable than the quality of an activity. It is also easy to determine whether a deliverable is finished, while various activities can be performed in parallel making the activity-based model more difficult to manage.

The model presented in this article is object-oriented and deliverable-based. The following paragraphs compare our model with several other models for software development processes.

Fusion

The Fusion process [1] uses a deliverable-based workflow model. For each deliverable and milestone, the constructors and quality checks are well defined. It is therefore pretty straightforward to construct the object process model for the Fusion method. The object orientation of the method makes it easy to define user-specific extensions and customizations of the method. Object orientation also guarantees that extensions are consistent with the original method. You can read more about this topic in reference [3].

Objectory

The Objectory process [6] uses an activity-based workflow model. The process is complex, but it is made more manageable by viewing the process in different ways such as by notation, by workflow, by documentation, by artifacts and by workers. It is also made more manageable by necessary configuration (adaptation) to meet specific needs. The representation of the process in Objectory Online helps significantly in managing the complexity of the process. However, the object-oriented model for process is probably simpler. It would be easier to ensure the quality of each deliverable and it would be probably also be easier to manage the iterative development with the object-oriented model.

OPEN Process Specification

OPEN Process Specification [2] defines activities as objects, tasks as object operations and deliverables as operation postconditions. This approach leads to a complex solution, with time-boxes for managing activities, and probabilistic links between tasks and activities. Compared to the object model presented in this article, this model is more complex, seems less fail-safe (activities and tasks do not contain any quality criteria; their quality is typically ensured by another task) and would be more difficult to customize.

Microsoft Solutions Framework (MSF)

MSF [4] is a set of guidelines for developing client-server systems. MSF establishes a common framework that defines a team model, a process model and an application model. MSF can include user-defined processes (called best practices) that are compatible with the framework. The MSF process model is deliverable-based and can therefore quite naturally incorporate concrete processes defined in the object-oriented process model focused on here.

Capability Maturity Model (CMM)

CMM [5] defines a set of quality criteria and key practices that are used to classify software processes into five maturity levels. An actual object-oriented process can be directly evaluated using CMM because key practices and quality criteria can be compared with the quality-assurance methods of the deliverable objects in the process.

Conclusions

We have described an object-oriented model for a development process. The main artifacts of our model are deliverables, which are modeled as objects with constructors and quality-assurance methods and a number of specific attributes. The process is simpler and it is easier to customize than other development process models. We have described our experience with the model, which we find flexible, robust and, in general, easy to use.

References

[1] Coleman, D. et al.: Object-Oriented Development: the Fusion method, Prentice Hall, 1994

[2] Henderson-Sellers B., Graham I., Younessi H.: The OPEN Process Specification, to be published

[3] Hruby, P.: The Object Model for a Product Based Development Process, ECOOP'97 Workshop on Modeling Software Processes and Artifacts, 1997

[4] Microsoft Solutions Framework version 2.0, Microsoft, 1997

[5] Paulk M. C. et al.: Capability Maturity Model for Software version 1.1, CMU/SEI-93-TR-024

[6] Rational Objectory Process version 4.0, Rational, 1997