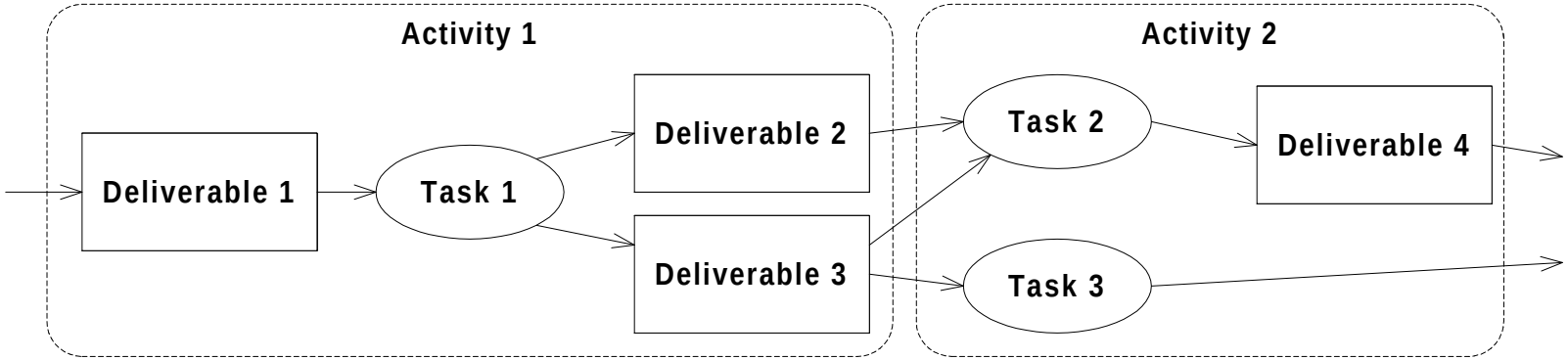
Abstract

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 as 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.

Motivation

The workflow model is defined by 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.



- Activity
- Task
- Deliverable

} Candidate classes?

Solution

Deliverables are considered objects with constructors and quality-assurance methods and a number of specific attributes. Evolution during software development is considered as object interactions.

Methods

- constructors (methods describing how to create deliverables)
- quality-assurance methods (defining the quality of a deliverable)

Attributes (object specific)

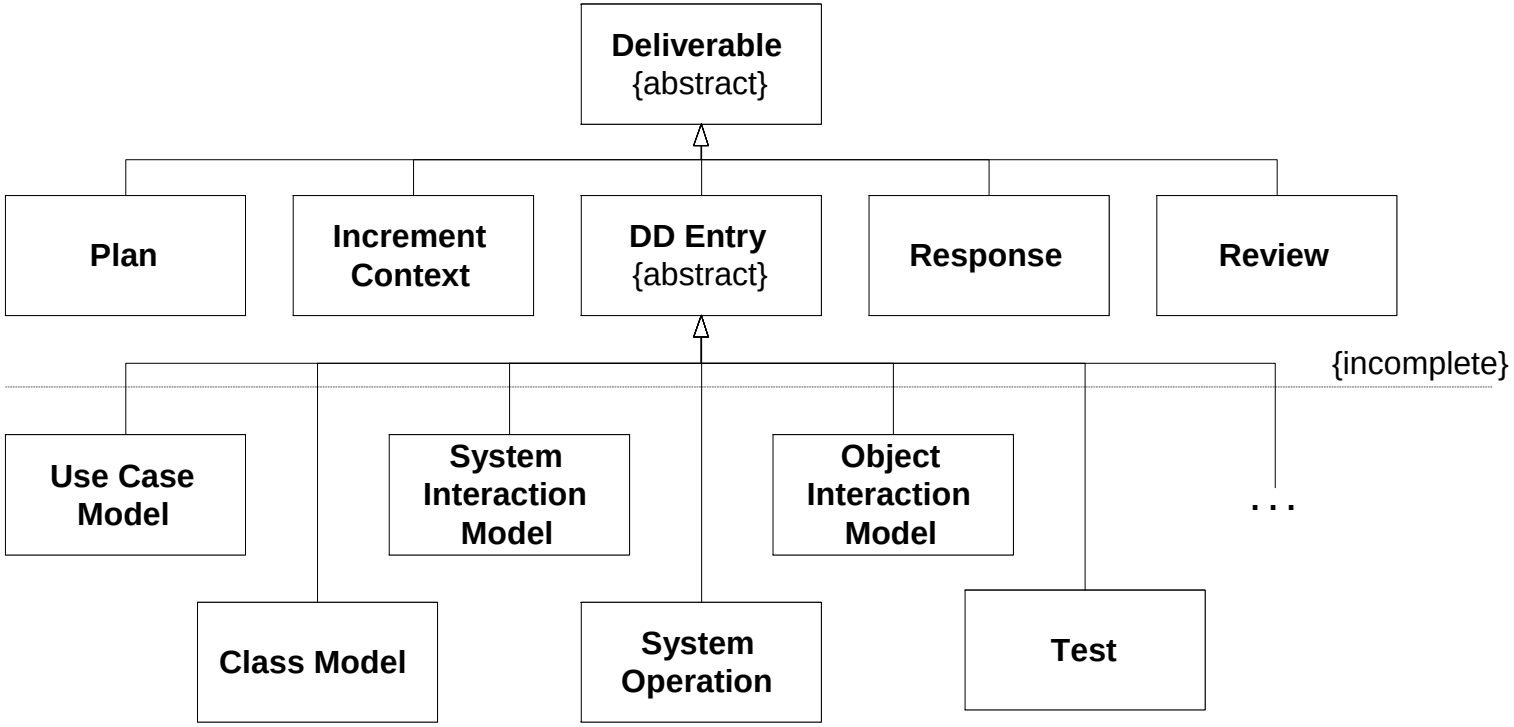
- kind
- name
- representation
- references to other deliverables
- project
- subsystem
- increment identification (task)
- responsible developer
- who created and modified the deliverable and when

Properties (class specific)

- purpose of a deliverable
- owner of a deliverable
- standard

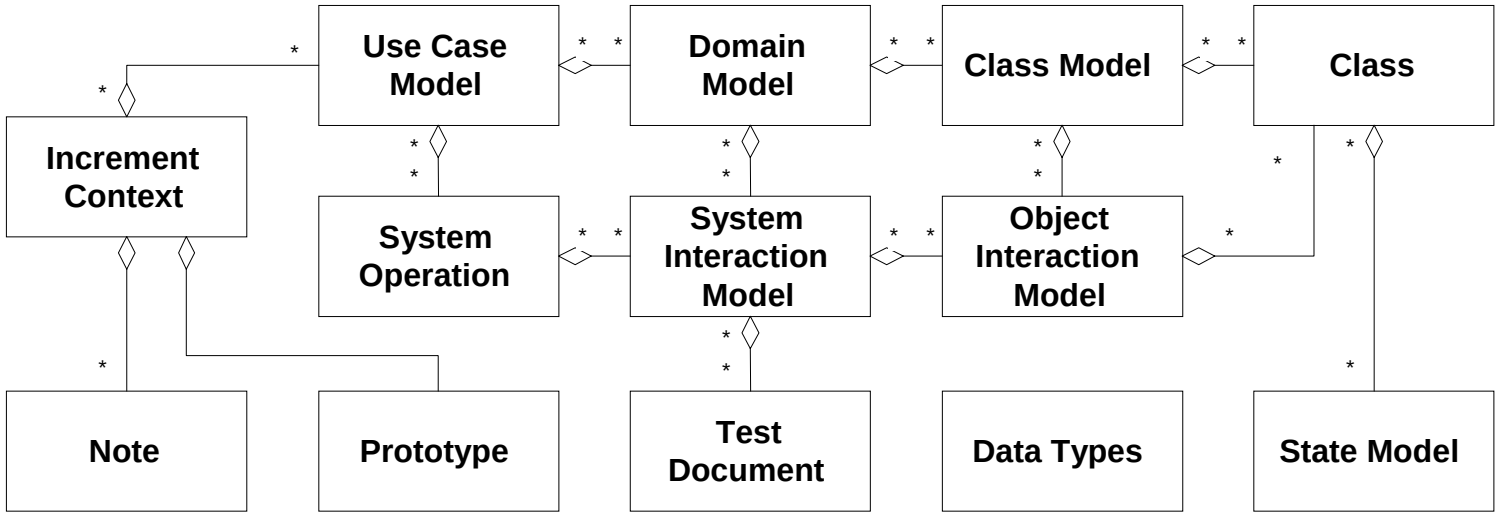
Inheritance Diagram of Deliverable Classes

The incompleteness of the inheritance tree allows for flexibility throughout the process and for the exact matching of project deliverables to different kinds of processes. The class DD Entry has abstract constructor and abstract quality criteria defined in derived classes.



Typical References between Deliverables

In principle, a deliverable can have a bidirectional link to any other deliverable, to enable maximum flexibility. The following diagram shows only typical references, reflecting the state of the data dictionary after six months of use.



Conclusions

- An object-oriented model 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.
- An object-oriented model is robust and easy to use. Small increments typically result in a small subset of deliverable classes. 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 by linking management and development deliverables.
- 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.
- An actual object-oriented process can be directly evaluated using the Capability Maturity Model because key practices and quality criteria can be compared with the quality-assurance methods of the deliverable objects in the object-oriented process.