An Object Model for Product Based Development Process

Pavel Hruby

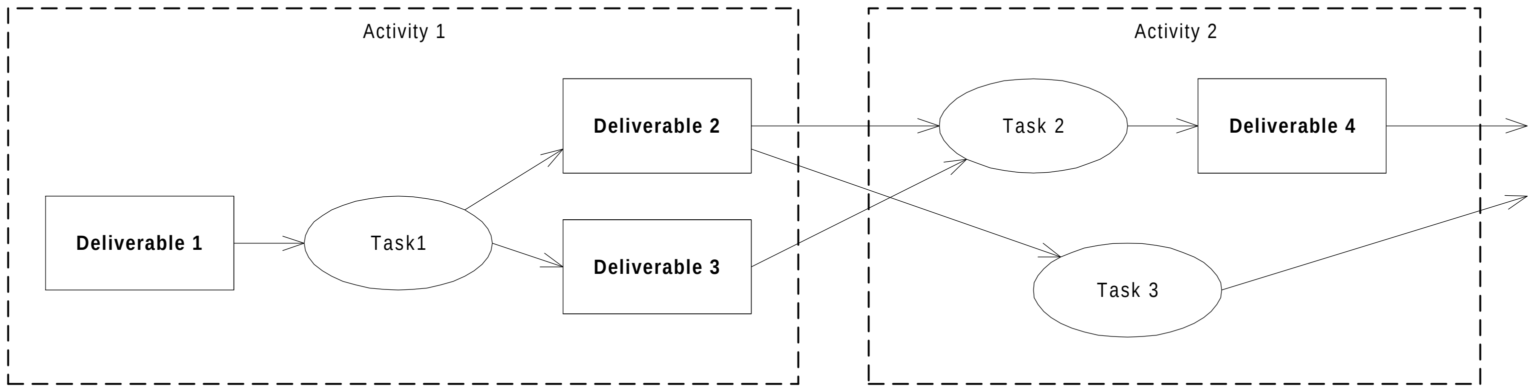
Navision Software a/s

ph@navision.dk

Good Development Process

- Easy to use
- Easy to maintain
- Good support for project management
- Flexible
- Robust and self-consistent

Workflow Model



- Activity
- Task
- Deliverable

Object-Oriented Model 1: Activities are Objects

- Used as OPEN process description
- Tasks are object operations
- Deliverables are operation postconditions
- [Henderson-Sellers, B.: OPEN Process Specification, 1997; http://www.csse.swin.edu.au/cotar/OPEN/PROCESS/index.html]

Activities are Objects: Potential Problems

- Difficult to determine appropriate set of activities for the project
  - depends on the project characteristics (size,…)
  - requires detailed knowledge of the methodology
- Activities are defined in the core method
- Not fail-safe

Object-Oriented Model 2: Deliverables are Objects

- Methods:
  - tasks (constructors)
  - quality methods (consistency, completeness,…)
- Attributes (see next slide for details):
  - content
  - references to other deliverables
- No “activities” in the model

Object Attributes and Methods

Deliverable
{abstract}

<<constructor>>
Procedure how to create the deliverable

<<quality criteria>>
Completeness
Simple consistency
Semantic consistency

Kind
Name
References to other deliverables
Description //UML diagram, text, prototype, etc.
Project
Subsystem
Context //reference to the context deliverable
File&directory //if deliverable is code, test, etc.
Responsible developer
Audit attributes

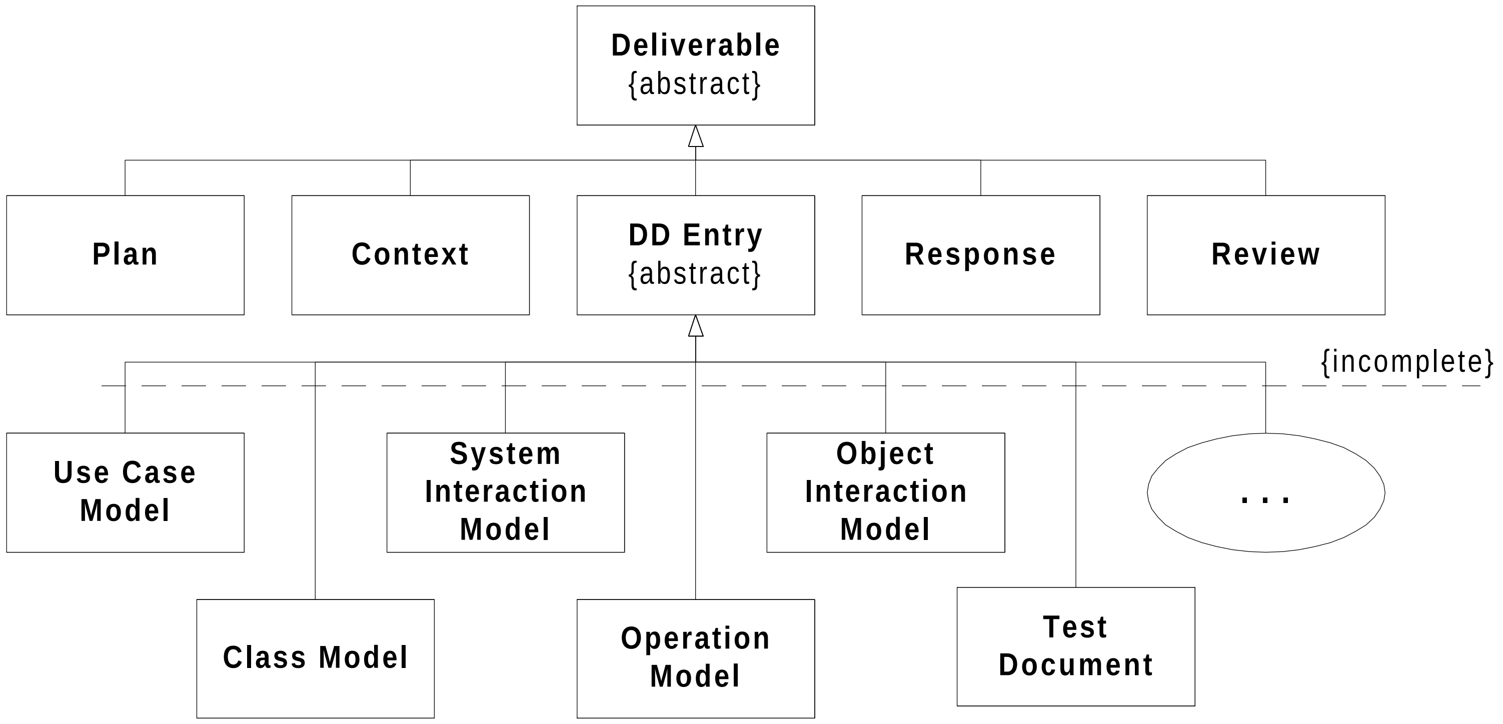
Context
{superclass = Deliverable}

<<constructor>>
1. Brainstorm, or obtain suggestions of requirements
2. Identify stakeholders
3. Modify context document in the light of stakeholder analysis.

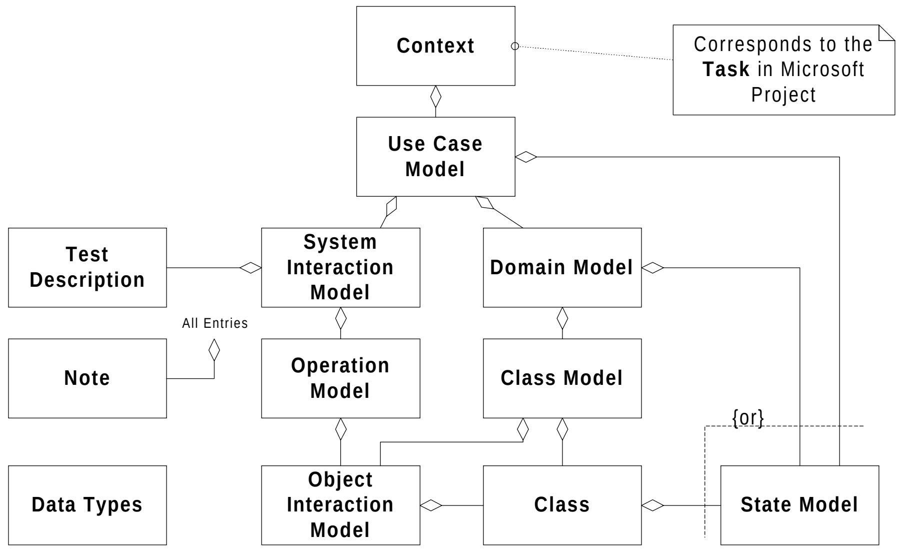
<<quality criteria>>
Document is complete in the light of stakeholder analysis

Synopsis
Requirements
Solution
Not covered issues
Motivation (benefits)
Consequences (costs)
Target group (stakeholders)
Breakdown
Metrics (time estimates)
Comments

Inheritance Diagram



Typical References between Deliverables



Experience with the Process (Fusion with Use Cases)

- 350 documents in the repository
- context 36%, note 15%,
- system interaction model 14%,
- use case model 10%,
- class model 6%, operation model 5%,
- class 5%, domain model 4%,
- object interaction model 3%,

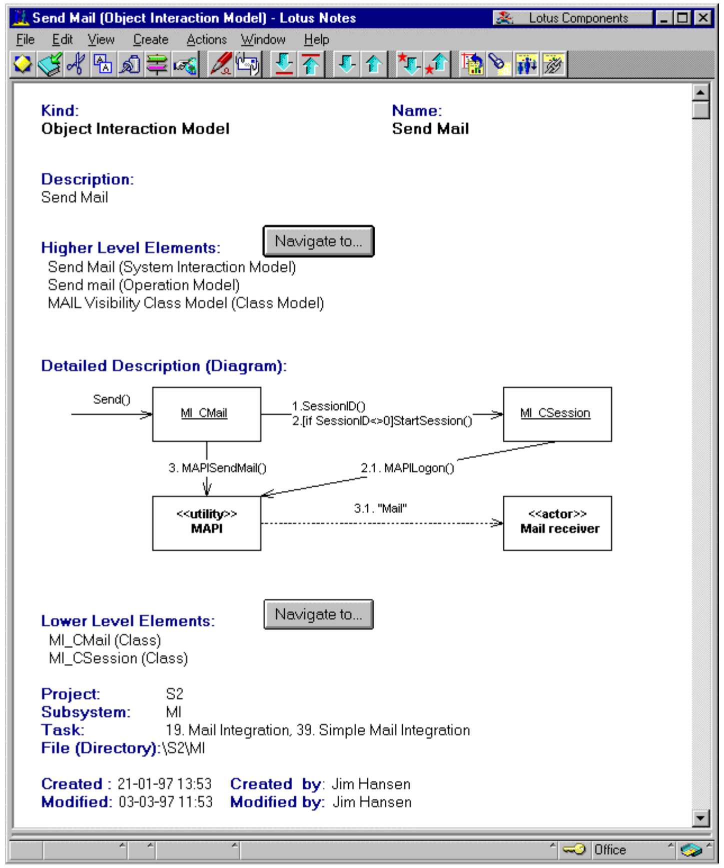
Benefits of the Model based on Deliverables as Objects

- It is easier to define set of deliverables rather than set of activities
- Good support for incremental development (context document is a single instance throughout the life-cycle)
- Different processes can use the same framework (flexible)
- Changes in deliverables do not affect the framework (process is easy to maintain)

Conclusions

- Process model with deliverables as objects:
  - Different processes can use the same framework (flexible)
  - Easy to maintain (changes in deliverables do not affect the framework)
  - Good support for management
  - Robust and self-consistent (constructors and quality methods)

Example of Design Deliverable (Object Interaction Model)


Send Mail (Object Interaction Model) - Lotus Notes
Lotus Components
File Edit View Create Actions Window Help
Navigate to...
Navigate to...
Office

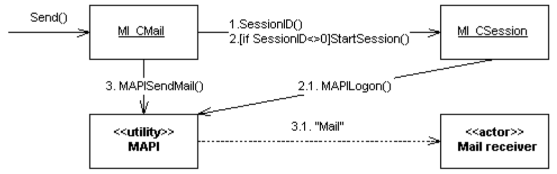
Kind:
Object Interaction Model

Name:
Send Mail

Description:
Send Mail

Higher Level Elements:
- Send Mail (System Interaction Model)
- Send mail (Operation Model)
- MAIL Visibility Class Model (Class Model)

Detailed Description (Diagram):

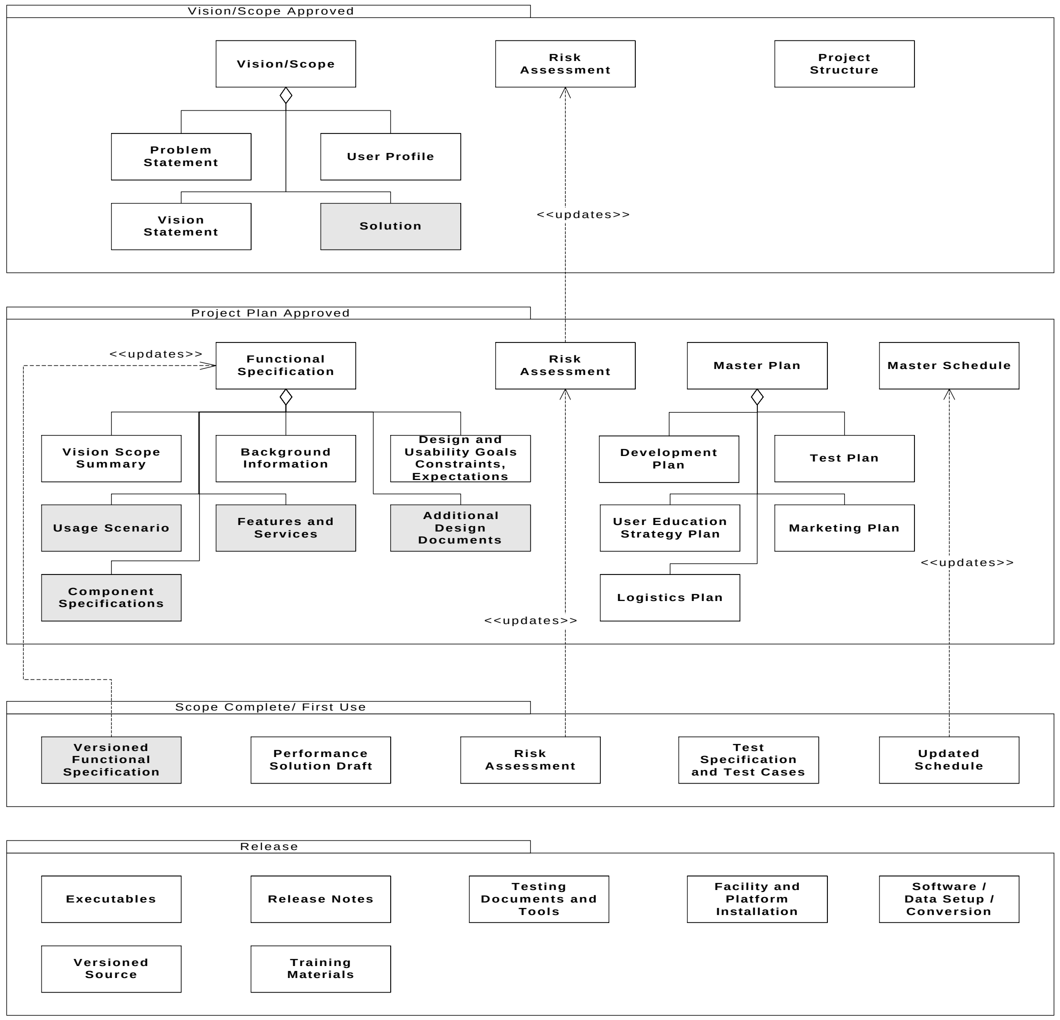


Lower Level Elements:
- MI_CMail (Class)
- MI_CSession (Class)

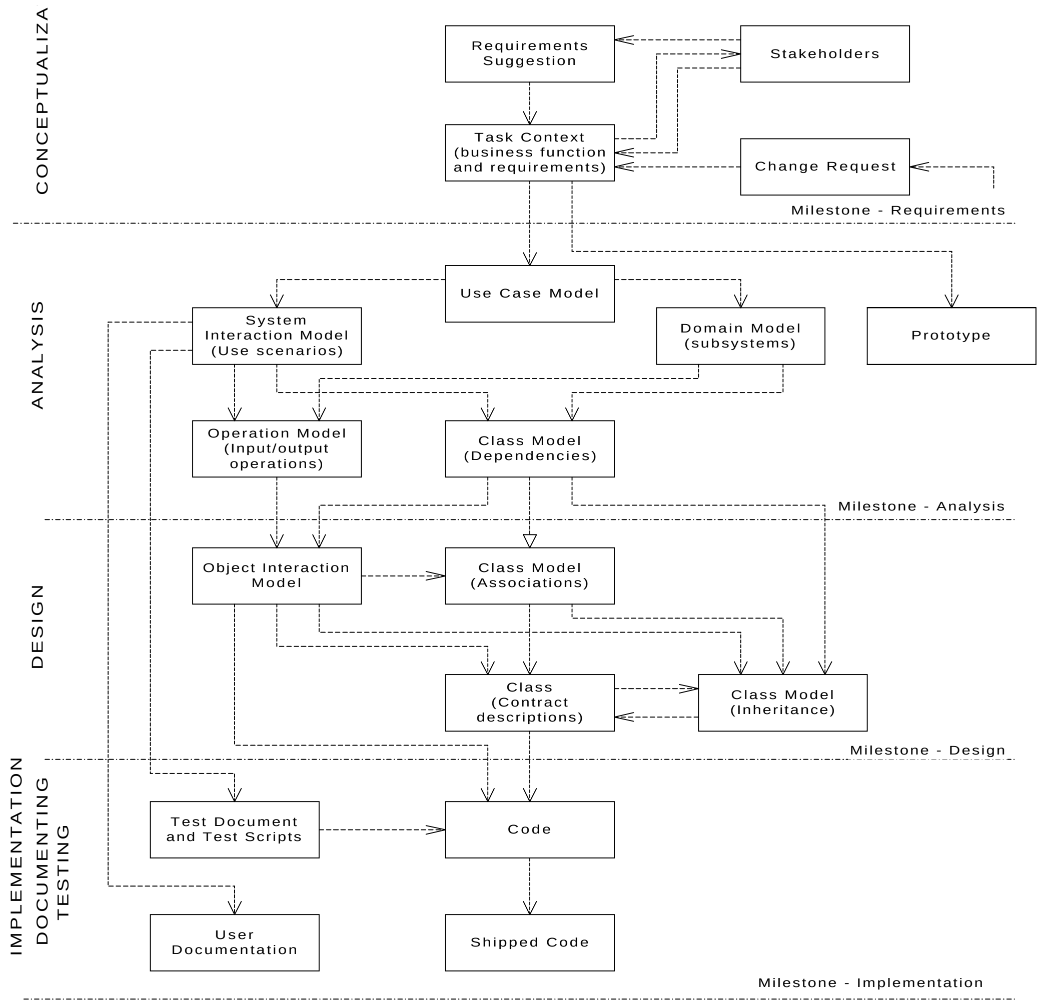
Project:	S2
Subsystem:	MI
Task:	19. Mail Integration, 39. Simple Mail Integration
File (Directory):	\S2\MI

Created : 21-01-97 13:53 **Created by:** Jim Hansen
Modified: 03-03-97 11:53 **Modified by:** Jim Hansen

Example of the Process: Microsoft Solution Framework



Example of the Process: Fusion with Use Cases


CONCEPTUALIZA
Requirements Suggestion
Stakeholders
Task Context (business function and requirements)
Change Request
Milestone - Requirements
ANALYSIS
Use Case Model
System Interaction Model (Use scenarios)
Domain Model (subsystems)
Prototype
Operation Model (Input/output operations)
Class Model (Dependencies)
Milestone - Analysis
DESIGN
Object Interaction Model
Class Model (Associations)
Class (Contract descriptions)
Class Model (Inheritance)
Milestone - Design
IMPLEMENTATION DOCUMENTING TESTING
Test Document and Test Scripts
Code
User Documentation
Shipped Code
Milestone - Implementation