Object-Oriented Model of Modified Fusion Process

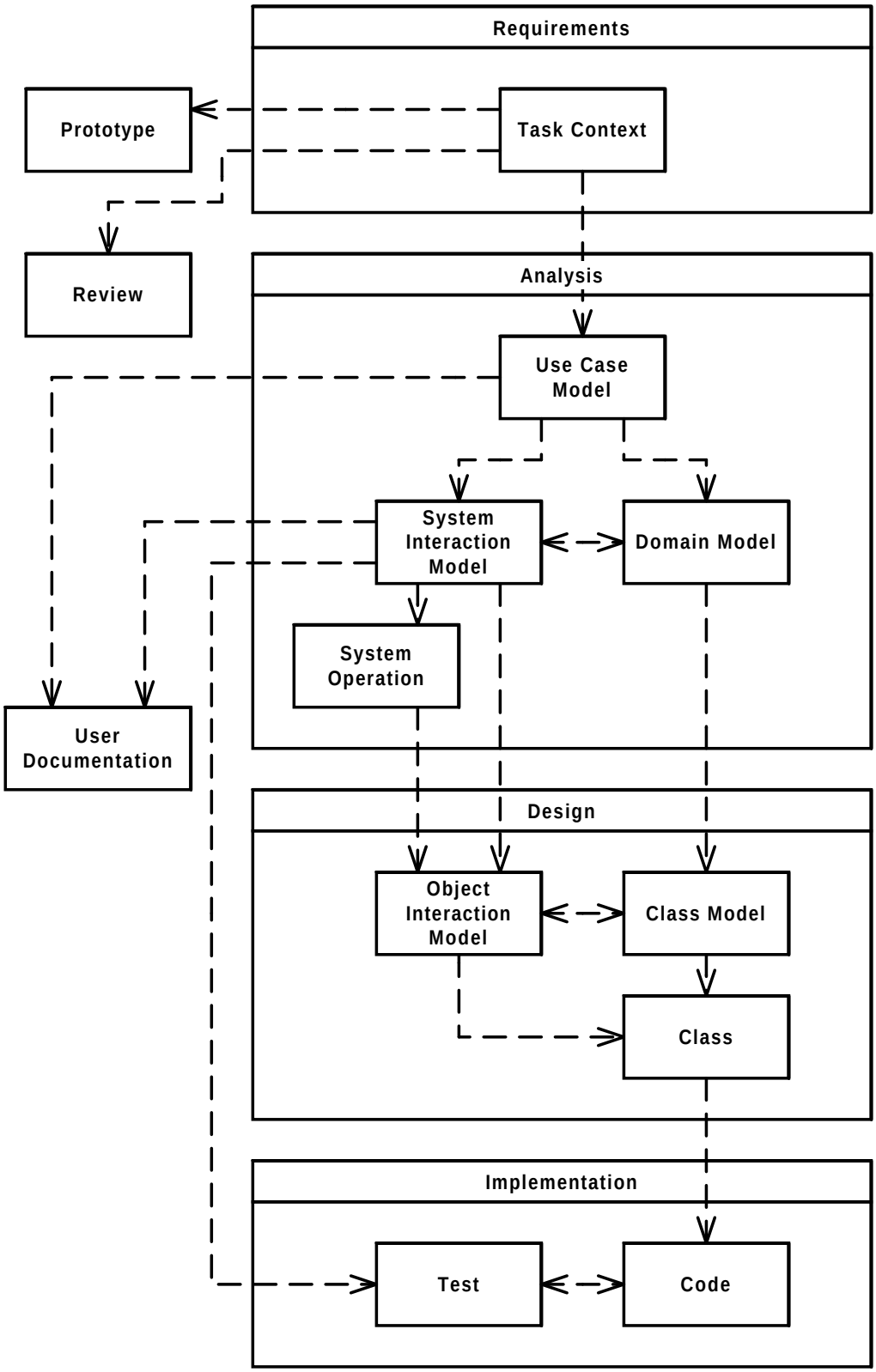
Characteristics of Original Process

The Fusion process (Derek Coleman, Patrick Arnold, Stephanie Bodoff, Chris Dollin, Helena Gilchrist, Fiona Hayes, Paul Jeremaes: Object-Oriented Development: the Fusion method, Prentice Hall, 1994) 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-oriented model for the Fusion method.

Benefits of Object-Oriented Representation

- An object-oriented model makes a distinction between the model and its representation.
- An object-oriented model allows for user-specific extensions and customizations of the method. Object orientation also guarantees that the extensions are consistent with the original method.
- It allows for easy mapping to the team model, specifying which team roles are responsible for which deliverables.

Static Structure Diagram of Modified Fusion Process



Examples of Process Class Specifications

Task Context
Derived from: Requirement suggestion Synopsis Requirements Solution Motivation and benefits Consequences and costs Stakeholders Task breakdown, time estimates and other metrics
«constructor» 1. Brainstorm, or obtain requirements suggestions 2. Identify stakeholders 3. Modify task context document in the light of stakeholder analysis «quality assurance» 1. Task context is complete in the light of stakeholder analysis 2. Be short. Describe only features relevant to management.
Purpose (responsibility): to provide overview of the task. Task context should not contain detailed technological issues; they should be removed to a separate document called Note.

Use Case Model
Derived from: Task Context Representation: Use case diagram or in simple cases, a list of use cases (behavioral patterns). Description of each use case containing: name, goal statement, precondition, postcondition, input, output, reads, changes, transactions
«constructor» 1. Identify actors, system operations and events 2. Describe each use case by use case scheme 3. Consider extends and uses relationships between use cases «quality assurance» All functional requirements are described by use cases
Purpose (responsibility): Specification of user services and behavioral patterns. Other benefits are: - gaining and understanding the problem - capturing an understanding proposed solution - identifying candidate classes - providing tests cases

Domain Model

Derived from:
Use case model, system interaction model, class model

Representation:
Static structure diagram containing packages and components and high-level relationships

«constructor»
1. Group classes to suitable clusters (logical subsystems)
2. Identify interface classes of your subsystem and show them in your domain model

«quality assurance»
1. All clients and suppliers for system operations appear in the domain model.
2. Diagram is consistent with class model and system interaction model.

Purpose (responsibility):
specify high level architecture of the whole system
specify an interface from your subsystem to the rest of the system

Class Model

Derived from:
Task context, use case model, domain model

Representation:
One or more class diagrams. Class diagrams with associations and dependencies specifying references between classes and class diagrams with generalizations.

«constructor»
1. Identify candidate classes, their responsibilities and collaborators.
2. Draw class diagram showing class dependencies.
3. Inspect all diagrams in object interaction model. Each operation needs a visibility reference from client to supplier class.
4. Decide on kind of visibility reference: dynamic or permanent, exclusive or shared, constant or variable.
5. Update class diagram. Record visibility information to tagged values of dependencies and attributes of association ends. In complex cases draw one class diagram for each system operation context. In simple cases draw one class diagram for each subsystem.
6. Consider common class responsibilities and class operations.
7. Consider common visibility relationships.
8. Draw class diagram with generalizations. In simple cases add generalizations to the existing class diagram.

«quality assurance»
1. The class model captures all relevant static information from task context requirements.
2. Boundary of the class model is consistent with domain model and use case model.
3. All classes and packages that appear in object interaction model appear in the class model as well
4. Each class has defined associations and dependencies with visibility information to other classes.
5. Ensure that classes with shared visibility are referenced by more than class. Ensure that classes with exclusive visibility are not referenced by several classes.
6. Each message defined in the object interaction model has an association or dependency in the class model.
7. If class model is represented both by inheritance and visibility diagrams, then each class appears in both kinds of diagrams.

Purpose (responsibility):
To determine relationships between classes.

System Operation

Derived from:
Use case model or system interaction model

Representation:
Signature, agent, description, precondition, postcondition, reads, changes, sends, exceptions

«constructor»
1. Describe different results of the operation as different subclauses in a postcondition
2. Find the events that have to be output in a postcondition and describe them as clauses to a precondition
3. Ensure that preconditions and postconditions are satisfiable
4. Describe Reads, Changes and Sends clauses from the pre- and postconditions.

«quality assurance»
All system operations are described to the data dictionary
Preconditions and postconditions are satisfiable
Preconditions and postconditions satisfy system interaction model.

Purpose (responsibility):
Specification of input or output operation or an operation crossing the subsystem border. The detailed design of an operation can be shown by activity diagram or can refer to the object interaction model.

Design

Derived from:
Analysis deliverables

Representation:
Object interaction model, represented by sequence diagrams or collaboration diagrams
Class model, possibly split to:
- class diagram with dependencies (showing object collaborations)
- class diagram with associations (showing data relationships)
- class diagram with associations, association ends and dependencies with tagged values (showing visibility between classes)
- class diagram with generalizations (showing inheritance)

Class, showing class responsibilities and class design

«constructor»
1. Develop object interaction model for each input/output operation
2. Specify visibility properties of class associations and dependencies
3. Refine inheritance relationships
4. Describe class interfaces (class headers)

«quality assurance»
1. Each class or package mentioned in the object interaction model appears in the class model.
2. If class model is split to visibility, inheritance, association and collaboration diagrams, then each class appears in all respective diagrams.
3. Object interaction model satisfy use case model, system interaction model, or postconditions of the system operations.
4. Each operation defined in an object interaction model (scenario) has an association or dependency in the class model.

Purpose (responsibility): Description of system architecture. Design of input/output operations. Identification of operations with parameters and return values. Defining inheritance and properties of associations and dependencies. Defining class responsibilities, operations and attributes.

Object Interaction Model

Derived from:
System operation, system interaction model

Representation:
One or more sequence diagrams for each system operation.
One or more object collaboration diagrams for each system operation
Creation chart (how objects create each other), if it is relevant.

«constructor»
1. Identify relevant objects involved in system operation execution
2. Identify controller and collaborators
3. Decide on messages between objects
4. Draw object interaction diagram for each input/output operation

«quality assurance»
1. Each class or package in the class or domain model appears in the object interaction model.
2. Functionality described by object interaction model satisfies postconditions of system operations, system interaction model and use case model.

Purpose (responsibility):
Definition of object interactions for each system operation.

System Interaction Model

Derived from:
Use case model, domain model.

Representation:
One or more sequence diagrams for each use case (i.e. primary and secondary scenarios
One or more collaboration diagrams for each use case.
Backus-Naur form for the subsystem.
State diagram or activity diagram of the subsystem.

«constructor»
1. Identify subsystems involved in operation execution
2. Decide on messages between subsystems
3. Draw one or more sequence diagrams for each use case
or, draw one or more system collaboration diagrams for each use case
or, express use case scenario in Backus-Naur form
or, draw state or activity diagram for each use case.

«quality assurance»
1. Each class, package or subsystem that appear in the system interaction model is described in the domain model.
2. Scenarios described by system interaction model satisfy use case model.

Purpose (responsibility):
Showing typical examples of system execution. Identification of input/output operations and events, but not their detailed description.

Class

Derived from:
Object interaction model, class model, system operation

Representation:
Text. Preferably in the form of CRC card showing responsibilities, collaborators, methods and attributes

«constructor»
1. Methods and their parameters are derived from operations in object interaction model, or can be already defined as input/output operations. Additional parameters of operations are derived from dependencies between classes in the class model.
2. All permanent references between classes are implemented by object attributes, specifying kind of visibility reference.
3. Source for data attributes is the class model and data dictionary, and possibly also description of operations.
4. Inheritance information is documented in the inheritance diagram.

«quality assurance»
1. All operations from object interaction model are recorded
2. All data attributes from class model, system operations and data dictionary are recorded
3. All visibility references from class model are recorded
4. All superclasses from class model are recorded

Purpose (responsibility): To define class responsibilities, methods and attributes