Object Oriented Methodologies

THE NEXT GENERATION

THE UNIVERSITY of York

Chris Kimble
February 2008

Overview

- A review of the theory
  - Why UML is not a methodology
- Three types of Object Oriented method
  - Object Modelling Technique (OMT)
  - Object Process Methodology (OPM)
  - Rational Unified Process (RUP)
- Strengths and weaknesses
- What happens in practice

THE UNIVERSITY of York

A review of the theory

- What are the key features of Object Oriented methods?
  - Have strong links to object oriented languages
  - Assumes the software description is closed but not complete
  - The software description is formed from observation of reality
  - Uses (closed) reality as a baseline to free the designer from concerns about the problems of dealing with inaccurate representations of the physical system
  - Rather than working from the description to reality, these methods work from reality back to the description = realist ontology and empiricist epistemology

Empiricism and Realism

- *Empiricist* arguments deal principally with epistemology claiming that all knowledge derives from observation. Thus, everything that can be known, can only be known through experience.
- *Realist* arguments deal principally with ontology claiming that there is such a thing as truth and that all beliefs can be tested against a reality that is knowable.

Objects

- An object is a something (a thing or a concept) that has a well-defined role in the application domain
- It has an identity, state and behaviour and is something to which actions are directed
- Its state encompasses its properties (attributes and relationships) and their values
- Its behaviour is how it acts and reacts

UML

- Unified Modelling Language (UML) is the accepted “industry standard” language for modelling the development of object oriented software.
- Grady Booch, James Rumbaugh and Ivar Jacobson (*The Three Amigos*) are credited with creating UML.
- The Object Management Group (OMG) are credited with creating a “standardised” language suitable for for dealing with object oriented analysis and design in “real world” settings

UML

STOP!

- is UML a Methodology?

THE UNIVERSITY *of York*

Chris Kimble
February 2008

UML

- UML is a modelling language
- Object Oriented Analysis (OOA) and Object-Oriented Design (OOD) are processes
- UML has rules for syntax and usage but it does not have procedures (i.e. processes)
- Although a common language is needed in a methodology, a language alone does not make a methodology

THE UNIVERSITY *of York*

Types of OO method

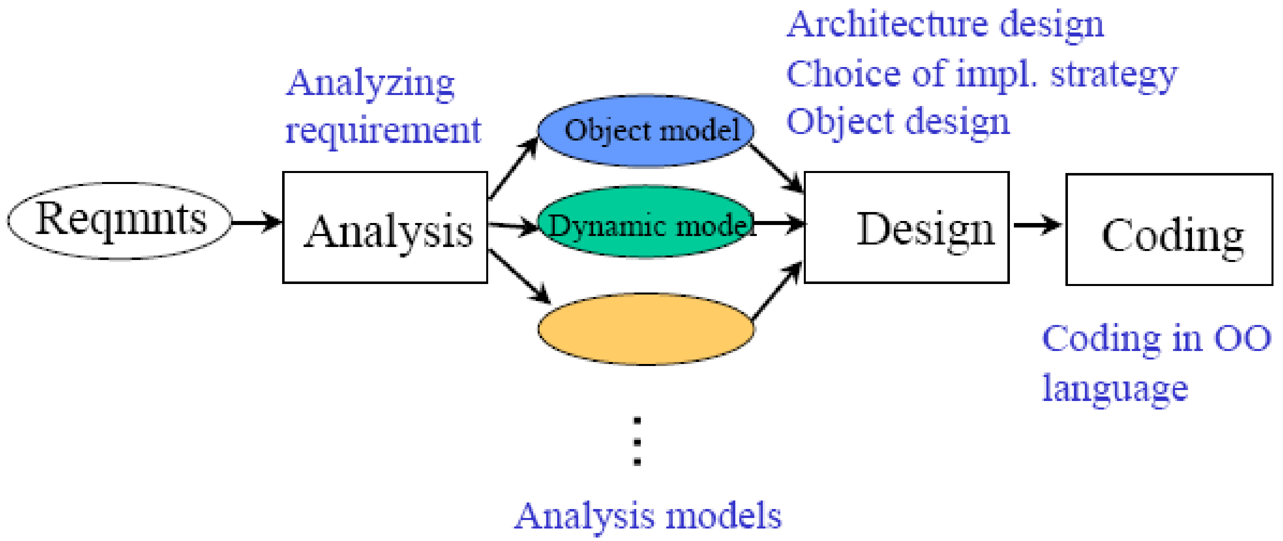
- There are many Object Oriented (OO) tools, techniques and languages but only a very few OO methodologies
- Most of the OO methodologies originated in the late 1990s following the rapid expansion of OO Languages in the 1980s
- We will consider three of the most influential:
  - Object Modelling Technique (OMT)
  - Object Process Methodology (OPM)
  - Rational Unified Process (RUP)

OMT

- OMT (Object Modelling Technique) was one of the first OO methodologies and was introduced by Rumbaugh in 1991

- It uses three different models that are combined in a way that is analogous to the older structured methodologies (e.g. Yourdon DeMarco)

OMT – An Overview



OMT - Analysis

- The goal of the analysis is to build a model of the world. The requirements of the users, developers and managers provide the information needed to develop the initial problem statement. Once the initial problem is defined, the following tasks are carried out:
  - Build the Object Model, including a Class Diagram and a Data Dictionary
  - Develop the Dynamic Model, including State Transition Diagrams and global Event-Trace Diagrams
  - Constructing the Functional Model including Data Flow Diagrams and constraints
  - Verify and refine the three models

OMT – The Models

- *The Object Model (OM)*:
  – depicts the object classes and their relationships (together with their associated attributes and operations) as a *Class Diagram,* which represents the static structure of the system
- *The Dynamic Model (DM)***:**
  – captures the behaviour of the system over time and the flow of control and events in *Event-Trace Diagrams* and *State Transition Diagrams (State Charts)*
- *The Functional Model (FM)***:**
  – a hierarchical set of *Data Flow Diagrams (DFD)* that describe internal processes independently from how these processes are performed

OMT - Object Design

- Object design specifies all of the details needed to describe how the system will be implemented
- All of the classes, associations, attributes and operations are fully defined, together with the operations and data structures and any internal objects needed for implementation

THE UNIVERSITY *of York*

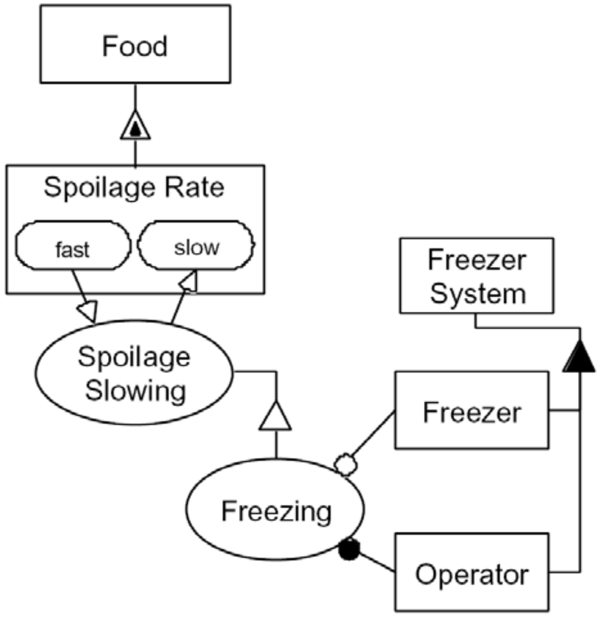
OMT - System Design

- The system design phase deals with the high-level structure of the system, e.g:
  - Identify global resources
  - Organize the system into subsystems
  - Identify boundary conditions
  - Identify concurrency
  - Allocate subsystems and tasks
  - Establish trade-off priorities
  - Choose a strategy for implementing data stores
  - Choose a strategy for implementing software controls

Object Process Methodology (OPM)

- OPM is a so-called second generation methodology and was first introduced in 1995
- OPM has only one diagram the Object Process Diagram (OPD) for modelling the structure, function and behaviour of the system
- Every OPD can be described in text form using the Object Process Language (OPL) a constrained natural language

An OPD and its equivalent in OPL



Food exhibits **Spoilage Rate,** which can be **fast** or **Slow.**

Spoilage Slowing changes **Spoilage Rate** from **fast** to **Slow.**

Freezing is **Spoilage Slowing.**

Freezing System consists of **Freezer** and **Operator**.

Freezing requires **Freezer**.

Operator handles **Freezing**.

Object Process Methodology (OPM)

- OPM has a strong emphasis on modelling but has a weaker emphasis on process and consists of only three main processes:
- Initiating
  - determining the high-level requirements, the scope of the system and the resources that will be required
- Developing
  - the detailed analysis, design and implementation of the system
- Deploying
  - introduction of the system to the user and subsequent maintenance of the system

THE UNIVERSITY *of York*

OPM - Initiating

Consists of three sub-process:

- *Identifying*: the needs and/or opportunities to justify the development of the system

- *Conceiving*: the system is “conceived” through determining its scope and ensuring that the necessary resources are available

- *Initializing*: determining the high-level requirements of the system

THE UNIVERSITY *of York*

OPM - Developing

Consists of three sub-process:

- *Analyzing*: eliciting requirements, modelling the problem domain in OPDs/OPLs and selecting an architecture
- *Designing*: adding implementation specific details, refining the architecture and detailing the process logic (to be implemented as the program)
- *Implementing*: constructing the components of the system and linking them together

THE UNIVERSITY *of York*

OPM - Deploying

Consists of four sub-process:

- *Assimilating*: introducing the system into the user environment (training, documentation and system conversion)
- *Maintaining*: maintenance tasks necessary to keep the system in working order
- *Evaluating*: checking that the current system has the functionality needed to satisfy current requirements
- *Terminating*: declaring the current system as dead, applying post-mortem procedures and the generation of a new system

Rational Unified Process (RUP)

- RUP was developed at Rational Corporation in 1998 by the same people that developed UML
- Although it is said to have a 'life cycle' RUP has an evolutionary / iterative approach rather than the linear / waterfall approach of earlier methodologies

RUP – Phases and Iterations

- The RUP development cycle consists of four *phases* which can be further broken down into *iterations*

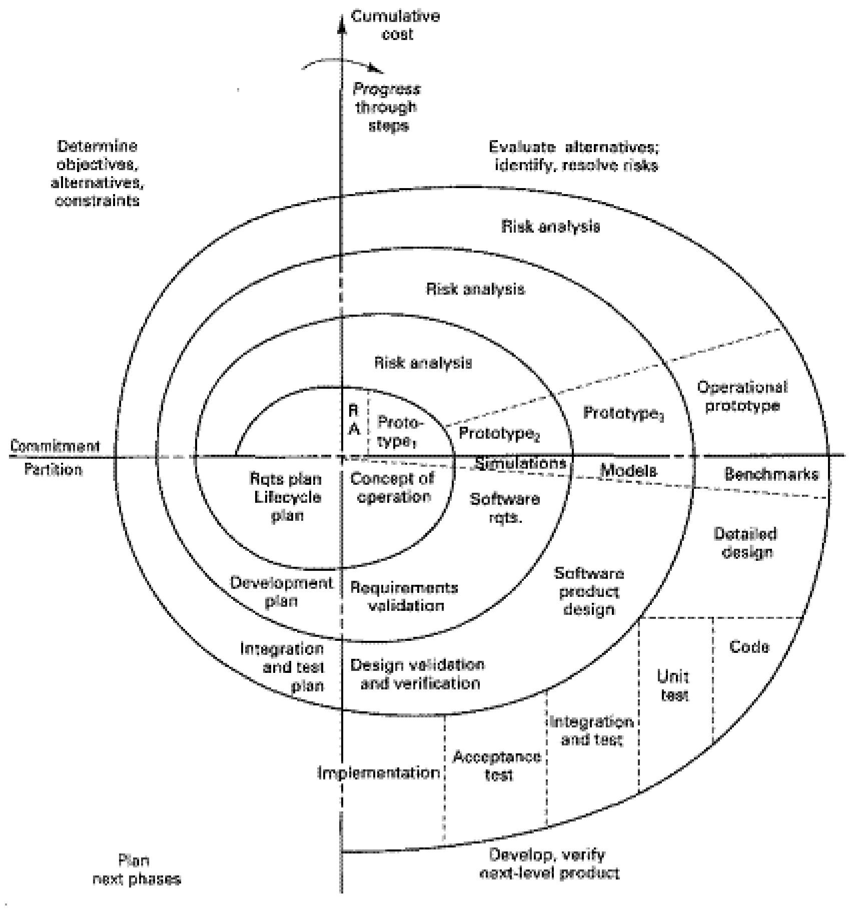
- Each *iteration* consists of nine work areas called *disciplines*

RUP - Disciplines

- The effort expended on a discipline depends on the phase in which the iteration is taking place. For example, Business modelling and requirements are more important in the earlier phases, whereas during later phases, most of the effort is put into deployment and testing

- For each discipline, RUP defines a set of *artefacts* (work products), *activities* (work undertaken on the artefacts), and *roles* (the responsibilities of the members of the development team)

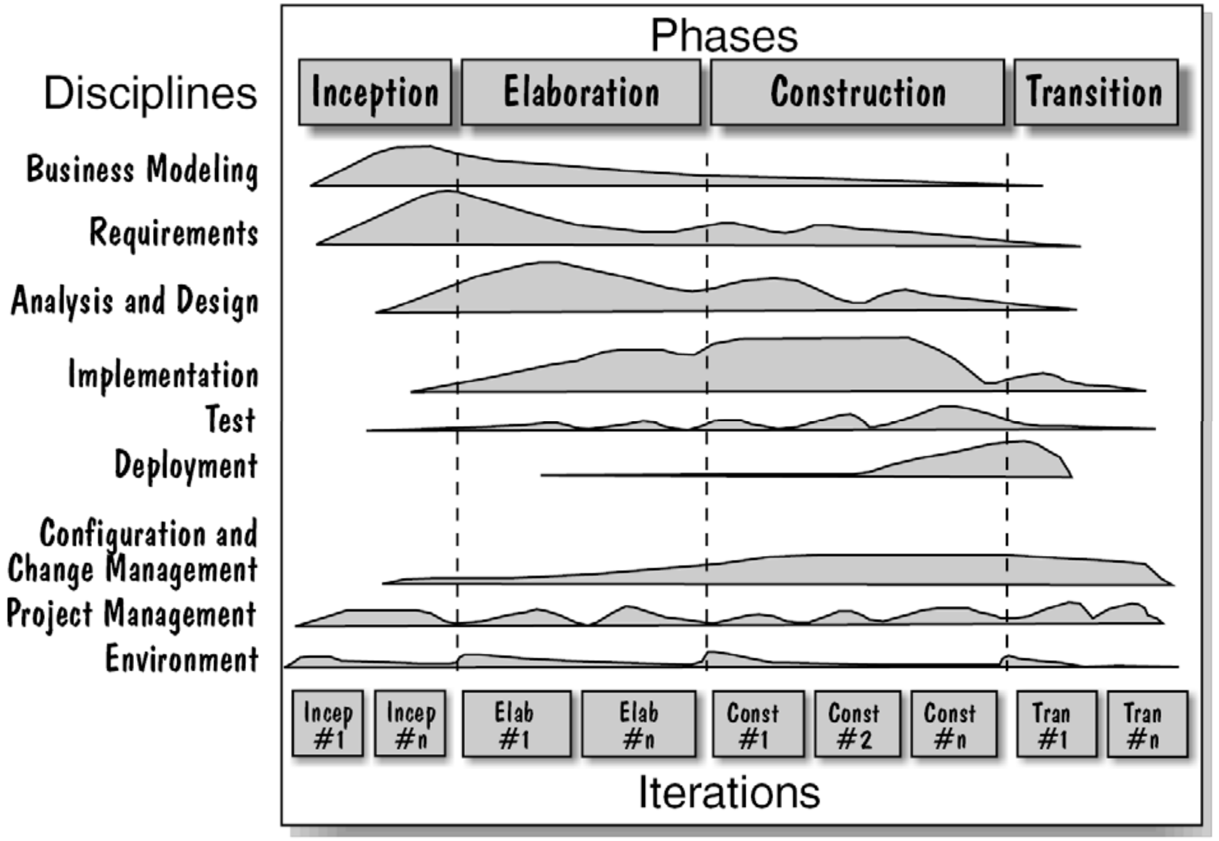
The ‘Life Cycle’



THE UNIVERSITY *of York*

Chris Kimble
February 2008

RUP – The 'Life Cycle'



The RUP life Cycle

THE UNIVERSITY *of York*

Chris Kimble
February 2008

Strengths of OO

- In comparison to structured methodologies it is generally claimed:
  - To encourage greater re-use
  - To produce a more detailed specification of system constraints
  - To have fewer problems with validation (are we building the right product?)

THE UNIVERSITY of York

Strengths of OO

- Generally claimed to allow:
  - A “seamless” development process
  - A shift of development effort from implementation to analysis: conceptual models, not computer models
- Also claimed:
  - To have a clearer division and increased consistency between analysis, design, and implementation
  - To produce a more “natural” model of the problem
  - To produce a “cleaner” design

Weaknesses of OO

- In comparison to structured methodologies it is generally claimed:
  - To be poor at dealing with data
  - To lack rigor and suitable project metrics
  - To have problems with verification (are we building the product right?) and demonstrating completeness
- Object oriented methods are also claimed to have a weak notion of:
  - Hierarchy (e.g. for designing software architecture)
  - Inheritance (e.g. for designing a reusable class library)

THE UNIVERSITY of York

Weaknesses of OO

- The language used to model / describe objects is limited and tends to describe the "what" rather than "how". Consequently it becomes difficult to represent processes and the flow of control

- OO does not have a simple way to deal with data and is not easy to use with legacy / database systems built using structured / data driven approaches

A Practical Example

- Wybolt, N. (1990). Experiences with C++ and Object Oriented software development. *SIGSOFT Softw. Eng. Notes,* **15**(2), 31-39.

THE UNIVERSITY *of York*