Software Development Methodologies

Lecturer: **Raman Ramsin**

Lecture 4

Integrated Object-Oriented Methodologies: OPM and RUP

Object Process Methodology (OPM)

- Introduced by Dori in 1995.
- Primarily intended as a novel approach to analysis modeling, combining the classic process-oriented modeling approach with object-oriented modeling techniques.
- Later evolved into a full-lifecycle methodology (2002).
- Only one type of diagram is used for modeling the structure, function and behaviour of the system.
- Single-model approach avoids the problems of model multiplicity, but the model produced can be complex and hard to grasp.
- OPM process is little more than an abstract framework, and resembles the generic software development process.

OPM: Process

- Consists of three high-level subprocesses:
  - *Initiating:* preliminary analysis of the system, determining the scope of the system, the required resources, and the high-level requirements
  - *Developing:* with the focus on detailed analysis, design and implementation of the system
  - *Deploying:* Introduction of the system into the user environment, and subsequent maintenance activities performed during the operational life of the system

OPM: Initiating

- *Identifying:* the needs and/or opportunities justifying the development of the system are determined.
- *Conceiving:* the system is "conceived" through determining its scope and ensuring that the resources necessary for the development effort are available.
- *Initializing:* the high-level requirements of the system are determined.

OPM: Developing

- *Analyzing:* typically involves:
  - eliciting the requirements
  - modeling the problem domain and the system in Object Process Diagrams (OPD) and their Object Process Language (OPL) equivalents
  - selecting a skeletal architecture for the system
- *Designing:* typically involves:
  - adding implementation-specific details to the models
  - refining the architecture of the system by determining its hardware, middleware and software components
  - designing the software components by detailing the process logic, the database organization, and the user interface
- *Implementing:* constructing the components of the system and linking them together; typically involves:
  - coding and testing the software components
  - setting up the hardware architecture
  - installing the software platform (including the middleware)

OPM: Deploying

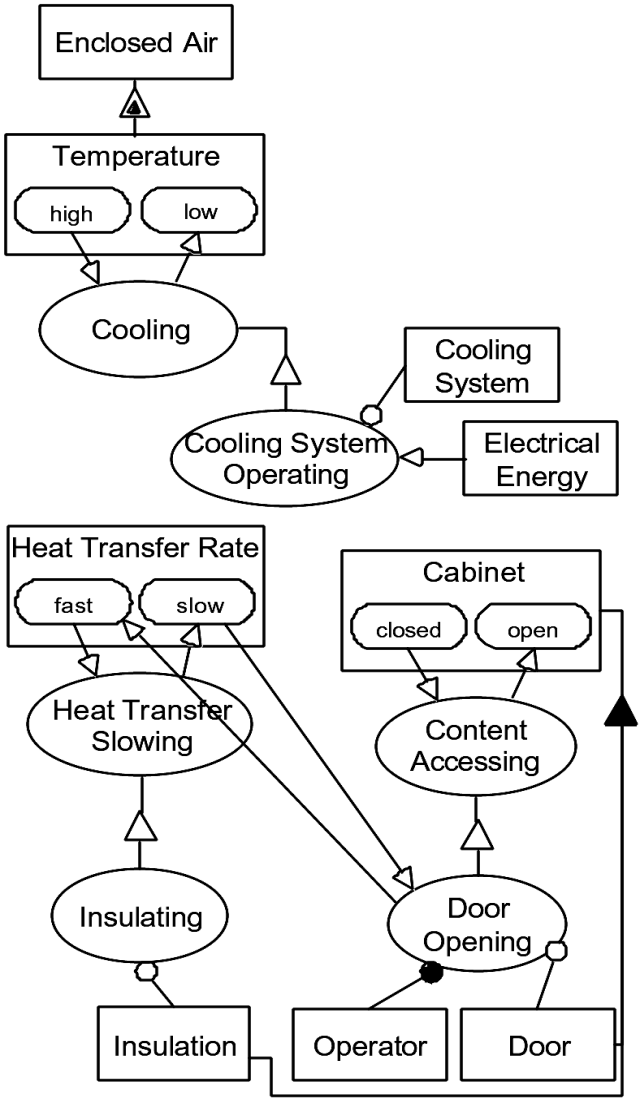
- *Assimilating:* introducing the implemented system into the user environment, mainly involving:
  - Training
  - generation of appropriate documents
  - data and system conversion
  - acceptance testing.
- *Using and Maintaining*
- *Evaluating Functionality:* [typically performed during the Using-and-Maintaining activity] checking that the current system possesses the functionality needed to satisfy the requirements
- *Terminating:*
  - declaring the current system as dead
  - applying the usual post-mortem procedures
  - prompting the generation of a new system

Object Process Diagram (OPD)

- Uses elements of types *object* and *process* to model the structural, functional and behavioural aspects
- Notation was later expanded to also include elements of type *state*, which were particularly useful in modeling real-time systems
- Every OPD can also be expressed in textual form, using a constrained natural language called the OPL (Object-Process Language)
- A set of OPDs is built for the system being developed, typically forming a hierarchy
- Multi-dimensional nature makes it difficult to focus on a particular aspect of the system without being distracted by other aspects.
- Some important behavioural aspects (such as object interactions, especially with regard to message sequencing) cannot be adequately captured

Object Process Diagram

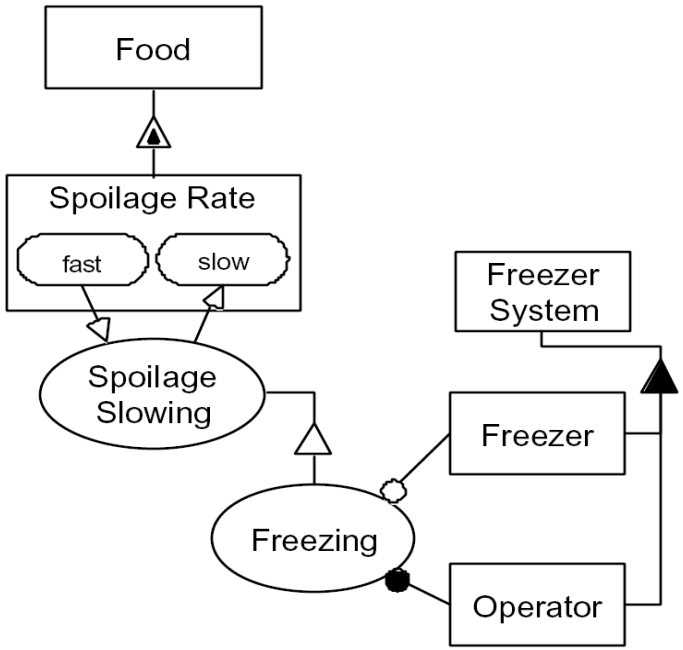
OPD



[Dori 2002]

Object Process Language

OPD and 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**.

[Dori 2002]

OPM: Strengths and Weaknesses

- ***Strengths***
  - Simplicity of process
  - Some degree of seamless development and traceability to requirements due to the singularity of the model type used (disrupted, though, because of OPD's limited modeling capacity)
  - Innovative structural and functional modeling in a single type of diagram (OPD)
  - Strong structural modeling at the inter-object level

OPM: Strengths and Weaknesses

- ***Weaknesses***
  - Process is defined at a shallow level, with ambiguities and inadequate attention to detail
  - Seamlessness and traceability are disrupted due to lack of behavioural models (especially at the inter-object and intra-object levels, directly affecting the identification and design of class operations)
  - No basis in system-level behaviour and usage scenarios
  - Poor behavioural modeling
  - No formalism
  - Poor intra-object structural modeling
  - Models are prone to over-complexity
  - No modeling of physical configuration

Rational Unified Process (RUP)

- Initial version officially released in 1998
- Revised versions introduced in 2000 and 2003
- Developed at Rational Corporation by the three principal developers of the OMT, Booch and OOSE (Objectory) methodologies: Rumbaugh, Booch and Jacobson
- A non-proprietary, somewhat less complex variant called USDP (Unified Software Development Process) was introduced in 1999

Rational Unified Process (RUP)

- UML-based
- Use case driven, a feature inherited from OOSE
- Iterative-incremental, with the overall process resembling the Micro-in-Macro process of the Booch methodology
- Covering the full generic lifecycle

RUP: Process – Phases

- Overall development *cycle* consists of four *phases*:
  - *Inception:* defining the scope and objectives of the project, as well as the business case
  - *Elaboration:* capturing the crucial requirements, developing and validating the architecture of the software system, and planning the remaining phases of the project
  - *Construction:* implementing the system in an iterative and incremental fashion based on the architecture developed in the previous phase
  - *Transition:* testing and releasing the system

RUP: Process – Iterations and Disciplines

- Each phase can be further broken down into *iterations*
- An iteration is a complete development loop resulting in a release of an executable increment to the system
- Each iteration consists of nine work areas (*disciplines*) performed during the iteration
- For each discipline, RUP defines sets of:
  - *artefacts* (work products)
  - *activities* (units of work on the artefacts)
  - *roles* (responsibilities taken on by development team members)

RUP: Process – Disciplines

1. *Business Modeling:* concerned with describing business processes and the internal structure of a business. A *Business Use Case Model* and a *Business Object Model* are developed.

2. *Requirements Management:* concerned with eliciting, organizing, and documenting requirements. The *Use Case Model* is produced.

3. *Analysis and Design:* concerned with creating the architecture and the design of the software system; results in a *Design Model* and optionally an *Analysis Model*.

4. *Implementation:* concerned with writing and debugging source code, unit testing, and build management. Source code files, executables, and supportive files are produced.

5. *Test:* concerned with integration-, system- and acceptance testing.

RUP: Process – Disciplines (Contd.)

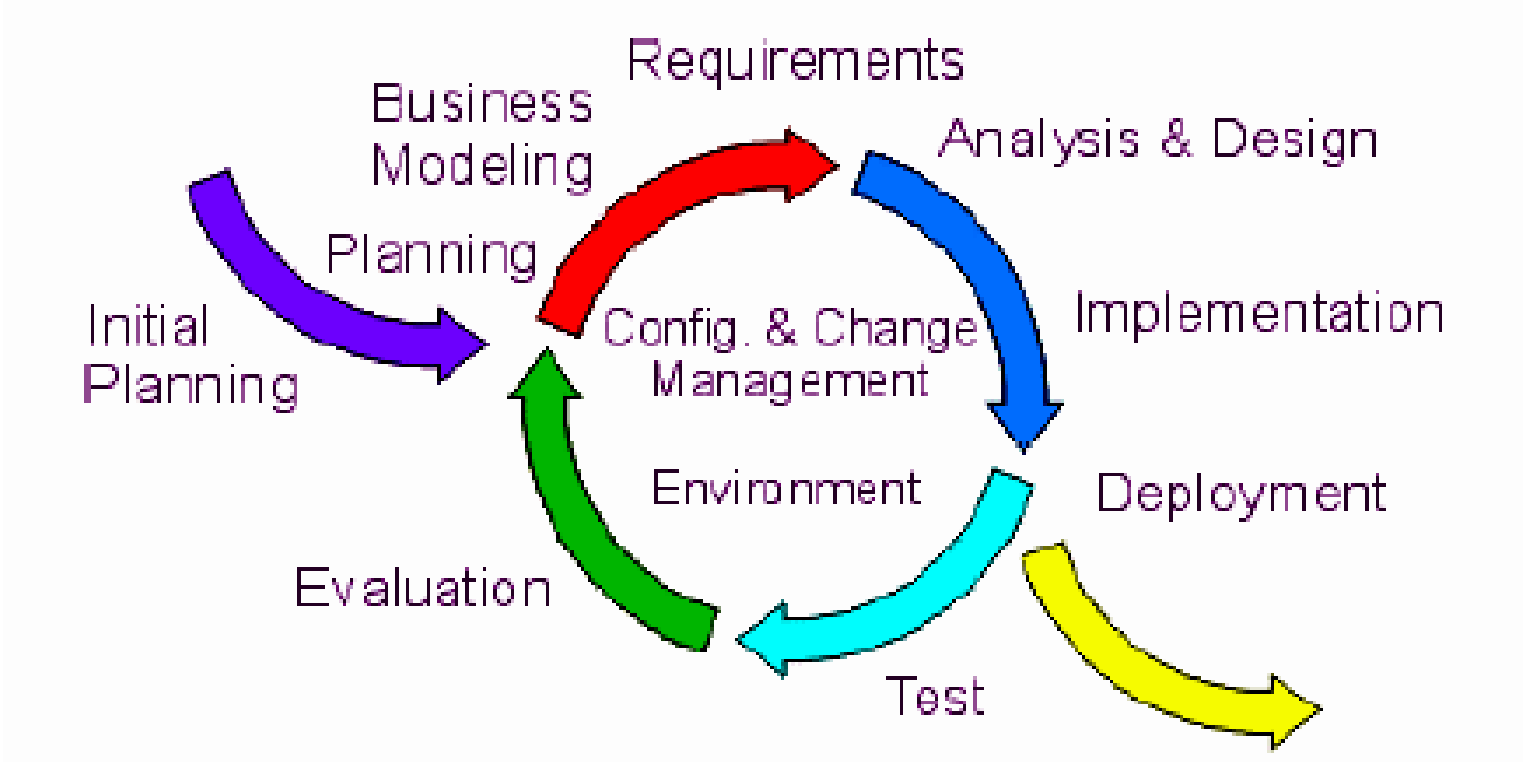
6. *Deployment:* concerned with packaging the software, creating installation scripts, writing end-user documentation and other tasks needed to make the software available to its end-users.

7. *Project Management:* concerned with project planning, scheduling and control.

8. *Configuration and Change Management:* concerned with version- and release management and change-request management.

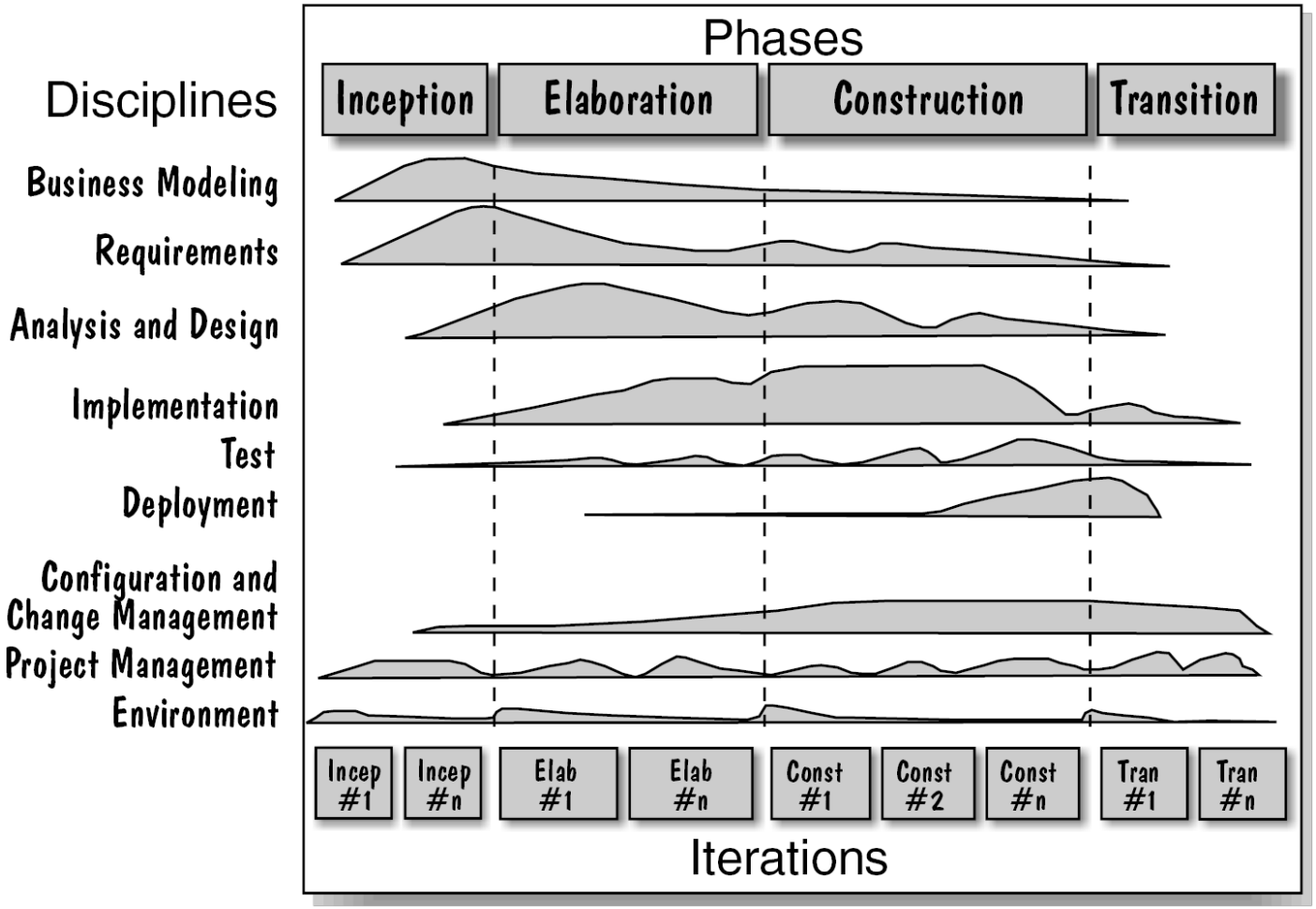
9. *Environment:* concerned with adapting the process to the needs of a project or an organization, and selecting, introducing and supporting development tools.

RUP: Process – Disciplines in Iterations



[Kruchten 2003]

RUP: Process – Disciplines in Iterations and Phases



[Kruchten 2003]

RUP: Process – *Inception* Phase

- Tasks include:

1. Describe the initial requirements.
2. Develop and justify the business case for the system.
3. Determine the scope of the system.
4. Identify the people, organizations, and external systems that will interact with the system.
5. Develop initial risk assessment, schedule, and estimates.
6. Configure the initial system architecture.

RUP: Process – *Elaboration* Phase

- Tasks include:

1. Produce an architectural baseline for the system.
2. Evolve the requirements model to 80% completion.
3. Draft a coarse-grained project plan for the construction phase.
4. Ensure that critical tools, processes, standards, and guidelines have been put in place for the construction phase.
5. Understand and eliminate high-priority risks of the project.

RUP: Process – *Construction* Phase

- Tasks include:

1. Describe the remaining requirements.
2. Develop the design of the system.
3. Ensure that the system meets the needs of its users and fits into the organization's overall system configuration.
4. Complete component development and testing, including both the software product and its documentation.
5. Minimize development costs by optimizing resources.
6. Achieve adequate quality.
7. Develop useful versions of the system.

RUP: Process – *Transition* Phase

- Tasks include:
  1. Test and validate the complete system.
  2. Integrate the system with existing systems.
  3. Convert legacy databases and systems to support the new release.
  4. Train the users of the new system.
  5. Deploy the new system into production.

RUP: Strengths and Weaknesses

- ***Strengths***
  - Iterative-incremental process
  - Well-documented process
  - Based on functional, behavioural, and structural modeling of the problem domain and the system
  - Traceability supported through use cases
  - Seamlessness (though with hiccups, e.g. transforming use cases to sequence diagrams)
  - Architecture-centric process (which necessitates early specification of an architectural blueprint)

RUP: Strengths and Weaknesses

- ***Strengths*** **(Contd.)**
  - Customizability addressed
  - Risk-based development, aimed at mitigating the risks before undertaking the tasks
  - Support for structural, behavioural and functional modeling at all levels (problem domain to objects; logical to physical)
  - Rich modeling language (UML), especially in structural and behavioural modeling features
  - Support for formalism (through UML/OCL)

RUP: Strengths and Weaknesses

- ***Weaknesses***
  - Very complex process
  - The process is confusing to those involved. The iterative-incremental nature of the process further complicates the issue.
  - Although advertised as customizable, configuring the process is a formidable task in itself.
  - Since the process is very complex, not having a maintenance phase, on the grounds that it can be performed by iterating the whole process as a cycle, is not convincing.
  - Prohibitive number of models
  - Strict adherence to UML, which is not necessarily constructive, especially since UML is not perfect and can exacerbate the model inconsistency problem.

References

- Dori, D., *Object-Process Methodology: A Holistic Systems Paradigm*. Springer, 2002.
- Kruchten, P., *Rational Unified Process: An Introduction*, 3rd ed. Addison-Wesley, 2003.