# Software Development Methodologies

Lecturer: **Raman Ramsin**

## Lecture 4

## Integrated Methodologies: OPM

# 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

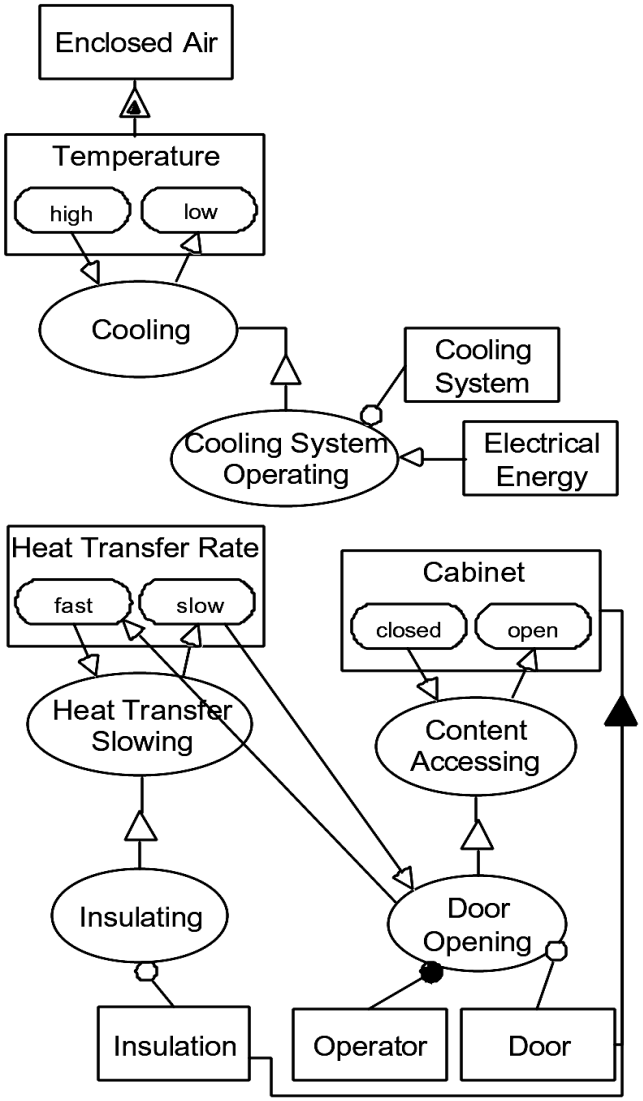
- *Assimilating:* introducing the implemented system into the user environment, mainly involving:
  - Training
  - generation of appropriate documents
  - data and system conversion
  - acceptance testing.
- *Using and Maintaining*
- *Evaluating Functionality:* [typically performed during the Using-and-Maintaining 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

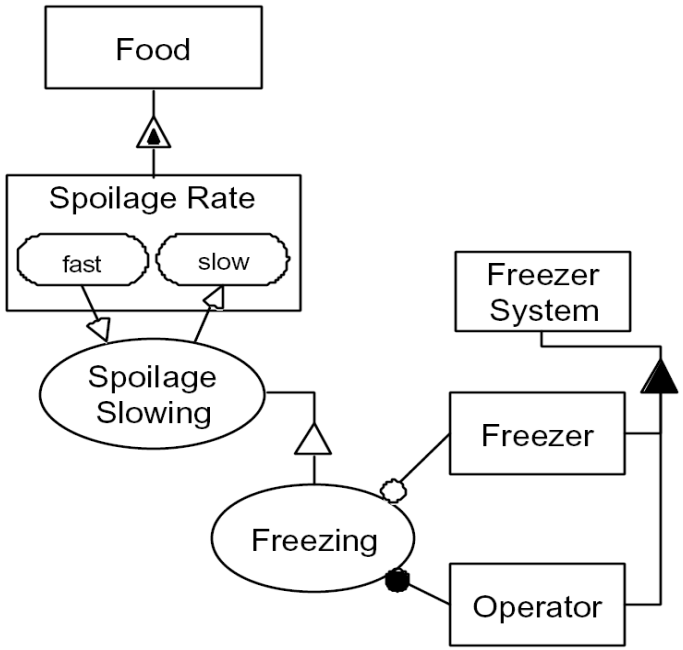
OPD




[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

# References


- Dori, D., *Object-Process Methodology: A Holistic Systems Paradigm*. Springer, 2002.