Overview of Object-Process Methodology (OPM)

Lecture 30, v02

John G. Artus

BSEE

MSSE

INCOSE ESEP

About This Courseware

- This courseware is presented as a brief overview of the Object-Process Methodology (OPM) developed by Dr. Dov Dori
- The majority of the material presented in this lecture is sourced from the book "*Object-Process Methodology, A Holistic Systems Paradigm*" by Dr. Dov Dori
- I, John Artus, make no claim of ownership of the material sourced from this textbook
- I, John Artus, am using the material sourced from this textbook, and other indicated sources, as content for this courseware for educational purposes only
- For further information and details on OPM, the reader is directed to the following two textbooks
  - *Object-Process Methodology, A Holistic Systems Paradigm,* by Dr. Dov Dori
  - *Model-Based Systems Engineering with OPM and SysML,* by Dr. Dov Dori
- This courseware lecture material has been sourced, interpreted, assembled, formatted, and copyrighted by John G. Artus for use in this educational context
- Anyone may freely access, and reuse this material in an educational context provided the copyright owner, John G. Artus, is recognized as the interpreter, assembler, and formatter of the source material used in the generation of this courseware, and provided that Dr. Dov Dori is recognized as the author of the textbook "*Object-Process Methodology, A Holistic Systems Paradigm*" from which the majority of the content of this courseware has been sourced, and that Dr. Dov Dori is recognized as the developer of the Object-Process Methodology

Objects, Processes, and States

- OPM is built on top of three types of entities
  - An object is a thing that exists or has the potential of existence, physically or mentally
  - A process transforms objects by, consuming, producing, or affecting them
  - A state is a situation within which an object can exist at a given moment in time
- States are used to describe (stateful) objects, and are not standalone things
  - Stateful means that the object has memory of a past state of existence
  - Previous transactions are remembered and may affect the current transaction
  - Stateful objects are those that have memory of more than one defined state of existence
  - Stateful objects therefore have the ability to change state based on the memory of the conditions that define the state
    - I declare that I am now in State X, because my current conditions are consistent with my memory of the definition of that state
- At any point in time, each stateful object exists in one of its defined states

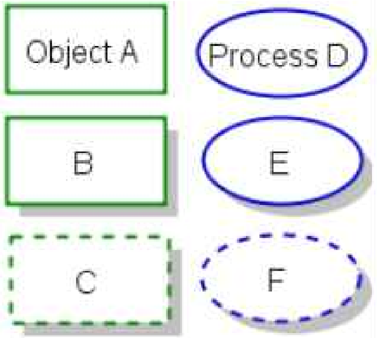
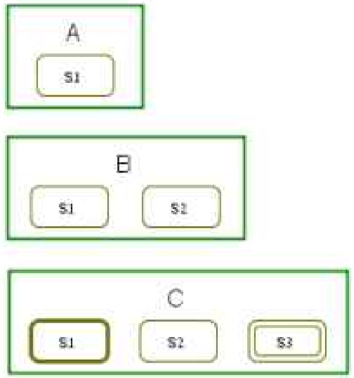
Premise of OPM

- The premise of OPM is that everything in the universe (and everything inherent in a system) is ultimately either an object or a process
  - Objects (form) — things that exist in one or more states
  - Processes (function) — things that happen and transform objects by creating or consuming them or by changing their states
- OPM explicitly shows the connections between objects and processes
- OPM is a formal paradigm to systems development, lifecycle support, and evolution
- In OPM, a system is represented simultaneously in formal graphics and in a natural language
  - Object-Process Diagrams (OPDs) – formal yet simple visual models
    - Every OPD construct is expressed by a semantically equivalent OPL sentence or part of a sentence and vice versa
  - Object-Process Language (OPL) - constrained natural language sentences
    - OPL is a dual-purpose language, oriented towards humans as well as machines

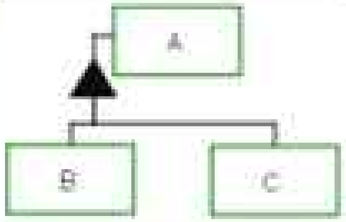
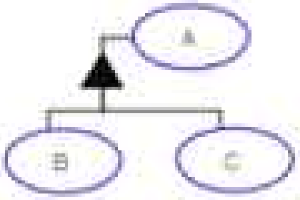
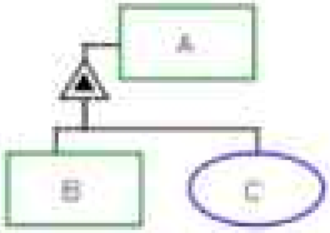
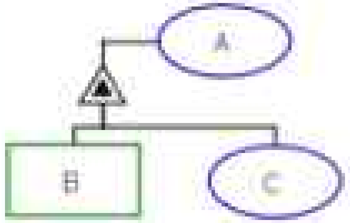
OPM Conventions and Syntax Summary

- This section includes a very brief overview of the OPM graphical syntax
  - This is intended as a reference guide, not a tutorial on how to use the OPM graphical syntax
  - It is recommended that students access the references listed in this presentation for more information and specific examples of OPM graphical syntax usage
- The symbol for objects and processes are rectangles and ellipses, respectively
- The first letter of object and process names is always capitalized
- Process names are expressed as gerunds (ends in "-ing") indicating they are active, dynamic things
- The symbol for state is a "roundangle" (rounded-corner rectangle)
- State names start with a lower-case letter

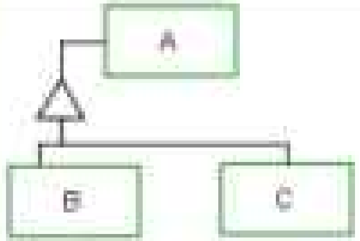
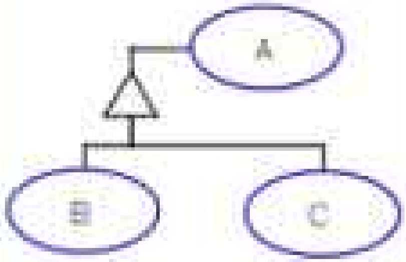
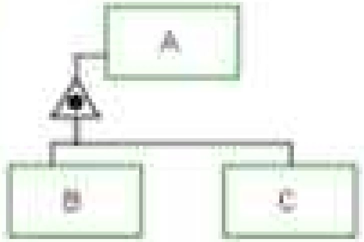
OPM Entities

Name	Symbol	OPL	Definition
Things: **Object**, **Process**		**B** is physical. (shaded rectangle) **C** is physical and environmental. (shaded dashed rectangle) **E** is physical. (shaded ellipse) **F** is physical and environmental. (shaded dashed ellipse)	An **object** is a thing that exists. A **process** is a thing that transforms at least one object. Transformation is object generation or consumption, or effect—a change in the state of an object.
State		**A** is **s1**. **B** can be **s1** or **s2**. **C** can be **s1**, **s2**, or **s3**. **s1** is initial. **s3** is final.	A **state** is situation an object can be at or a value it can assume. States are always within an object. States can be initial or final.

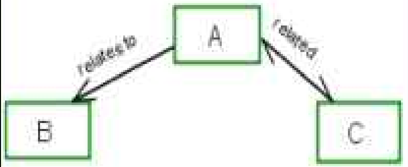
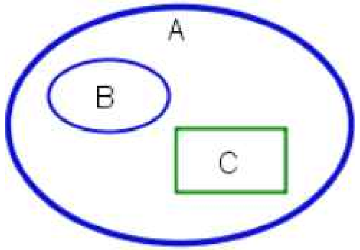
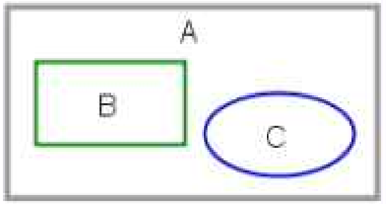
OPM Fundamental Structural Relations

	Name	Symbol	OPL	Semantics
Fundamental Structural Relations	Aggregation-Participation		A consists of B and C.	A is the whole, B and C are parts.
			A consists of B and C.	
	Exhibition-Characterization		A exhibits B, as well as C.	Object B is an attribute of A and process C is its operation (method). A can be an object or a process.
			A exhibits B, as well as C.	

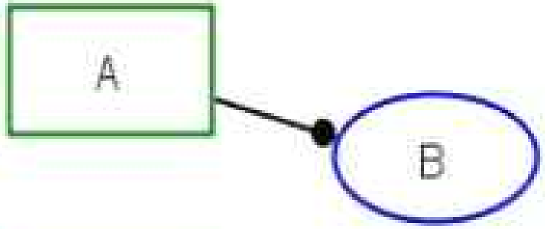
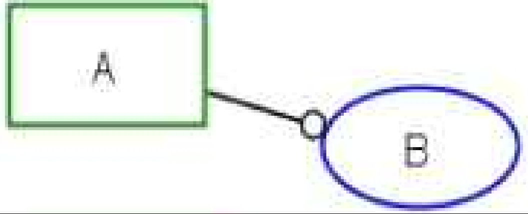
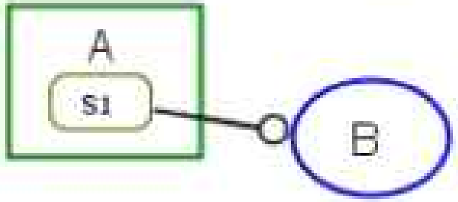
OPM Fundamental Structural Relations (continued)

	Name	Symbol	OPL	Semantics
Fundamental Structural Relations	Generalization - Specialization		B is an A. C is an A.	A specializes into B and C. A, B, and C can be either all objects or all processes.
			B is A. C is A.	
	Classification-Instantiation		B is an instance of A. C is an instance of A.	Object A is the class, for which B and C are instances. Applicable to processes too.

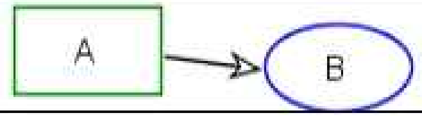
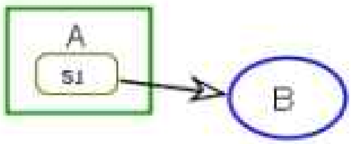
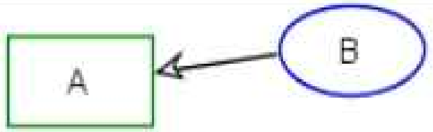
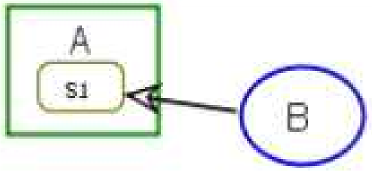
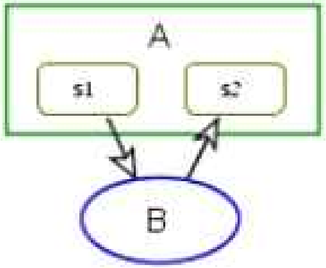
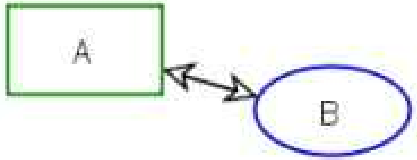
OPM Complexity Management

Unidirectional & bidirectional tagged structural links		**A** relates to **B**. (for unidirectional) **A** and **C** are related. (for bidirectional)	A user-defined textual tag describes any structural relation between two objects or between two processes.
In-zooming		**A** exhibits **C**. **A** consists of **B** **A** zooms into **B**, as well as **C**.	Zooming into process A, B is its part and C is its attribute.
		**A** exhibits **C**. **A** consists of **B**. **A** zooms into **B**, as well as **C**.	Zooming into object A, B is its part and C is its operation.

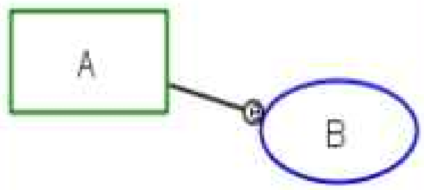
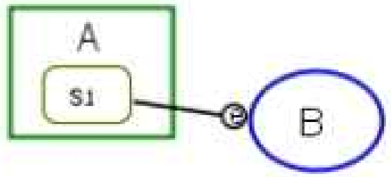
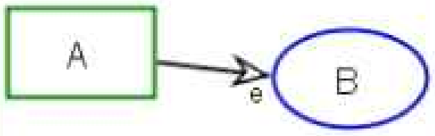
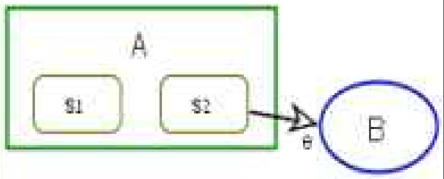
OPM Enabling Procedural Links

Name		Symbol	OPL	Semantics
Enabling links	Agent Link		**A** handles **B**.	Denotes that the object is a human operator.
	Instrument Link		**B** requires **A**.	"Wait until" semantics: Process B cannot happen if object A does not exist.
	State-Specified Instrument Link		**B** requires **s1** **A**.	"Wait until" semantics: Process B cannot happen if object A is not at state s1.

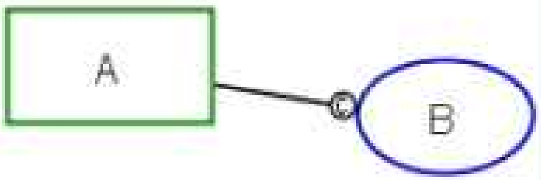
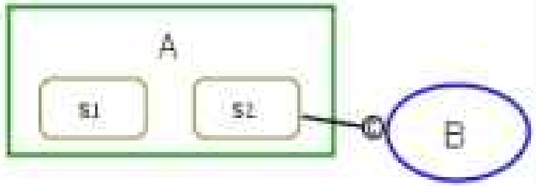
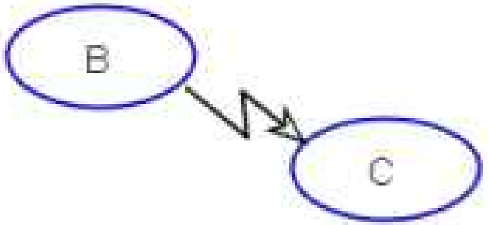
Transforming Procedural Links

Transforming links	Consumption Link		**B** consumes **A.**	Process B consumes Object A.
	State-Specified Consumption Link		**B** consumes **s1 A.**	Process B consumes Object A when it is at State s1.
	Result Link		**B** yields **A.**	Process B creates Object A.
	State-Specified Result Link		**B** yields **s1 A.**	Process B creates Object A at State s1.
	Input-Output Link Pair		**B** changes **A** from **s1** to **s2.**	Process B changes the state of Object A from State s1 to State s2.
	Effect Link		**B** affects **A.**	Process B changes the state of Object A; the details of the effect may be added at a lower level.

OPM Event Procedural Links

Name	Symbol	OPL	Semantics
Instrument Event Link		**A** triggers **B**. **B** triggers **A**.	Existence or generation of object A will attempt to trigger process B once. Execution will proceed if the triggering failed.
State-Specified Instrument Event Link		A triggers **B**. when it enters **s1**. **B** requires **s1** **A**.	Entering state s1 will attempt to trigger the process once. Execution will proceed if the triggering failed.
Consumption Event Link		**A** triggers **B**. **B** consumes **A**.	Existence or generation of object A will attempt to trigger process B once. If B is triggered, it will consume A. Execution will proceed if the triggering failed.
State-Specified Consumption Event Link		**A** triggers **B** when it enters **s2**. **B** consumes **s2** **A**.	Entering state s2 will attempt to trigger the process once. If B is triggered, it will consume A. Execution will proceed if the triggering failed.

OPM Condition and Invocation Procedural Links

Name	Symbol	OPL	Semantics
Condition Link		B occurs if A exists.	Existence of object A is a condition to the execution of B. If object A does not exist, then process B is skipped and regular system flow continues.
State-Specified Condition Link		B occurs if A is s1.	Existence of object A at state s2 is a condition to the execution of B. If object A does not exist, then process B is skipped and regular system flow continues.
Invocation Link		B invokes C.	Execution will proceed if the triggering failed (due to failure to fulfill one or more of the conditions in the precondition set).

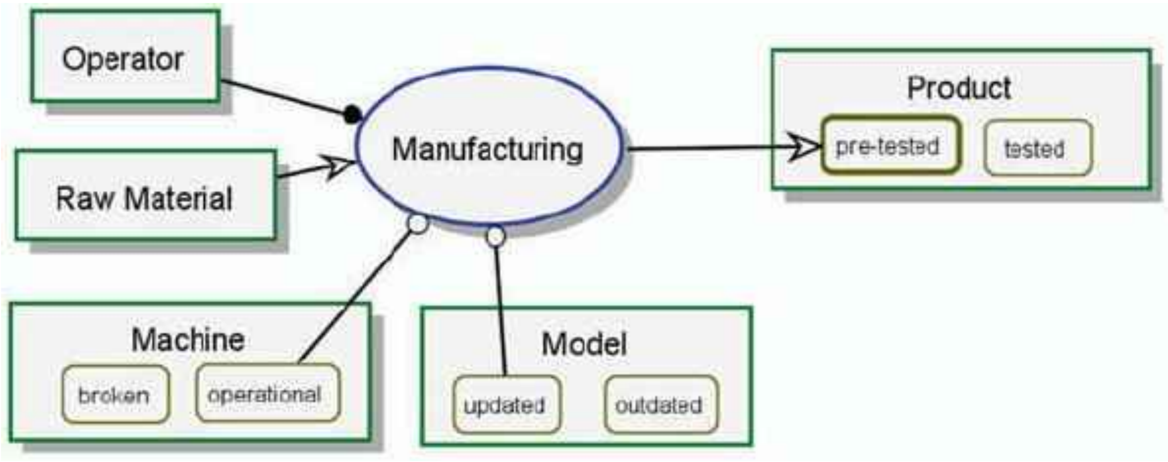
Object Process Language (OPL)

- The Object–Process Methodology includes a native language associated with the OPM graphical syntax, known as the Object Process Language (OPL)
- OPL serves two goals
  - One goal is to convert the set of Object-Process Diagrams (OPDs) into a natural language script
    - For communicating analysis and design results back to the prospective users and customers and getting their feedback and approval
  - The other goal of OPL, which is attained by its formality, is to provide the infrastructure needed for automated application generation
    - Including code, database scheme, and user interface generation
- To enhance OPM's expressive power, we associate with each OPD a collection of sentences in OPL as a textual, natural interpretation of the OPD's graphic representation
  - Each graphic symbol has a textual OPL equivalent
- The syntax of OPL is designed such that the resulting text constitutes plain natural, albeit syntactically restricted, English sentences

Object Process Language (OPL) (continued)

- The OPD-OPL pair follow the graphics-text equivalence principle
  - The OPD and its OPL paragraph contain the same information and are therefore reconstructible from each other
- Shown below is a simple example of the OPL for the given OPD
- Refer to the references listed in this presentation for more information on the OPL

Object Process Diagram (OPD)



Object Process Language (OPL)

Machine can be **operational** or **broken**.
Model can be **updated** or **outdated**.
Operator handles **Manufacturing**.
Manufacturing requires **operational Machine** and **updated Model**.
Manufacturing consumes **Raw Material**.
Manufacturing yields **pre-tested Product**.

An OPL paragraph of an OPD is a collection of sentences that express the content of the OPD

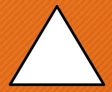
Recognition of OPM

- International Recognition
  - Formal international recognition of OPM as a modeling language has been established through the publication of ISO 19450:2024 Object-Process Methodology
  - While not a textbook for learning OPM, nor a manual for using OPM, it does precisely define the elements of the language
  - For learning the language, there are an extensive number of alternative resources available, such as the ones mentioned in the slide References
- Commercially-Available Modeling Tool
  - A commercially available modeling tool called OPCAT has been developed to support the modeling of systems with OPM
  - OPCAT is available in two executable formats:
    - Cloud execution (https://www.opcloud.tech/)
    - Standalone installation (https://esml.technion.ac.il/opm/opcat-installation/)

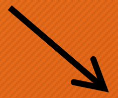
OPM Drawing Elements

 Aggregation – Participation

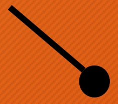
 Exhibition – Characterization

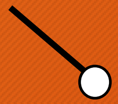
 Generalization – Specialization

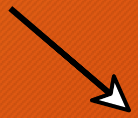
 Classification – Instantiation

 Uni-directional Structural Link

 Bi-directional Structural Link

 Enabling Link (Agent)

 Enabling Link (Instrument)

 Transforming Link

For use in drawing OPM diagrams using MS PowerPoint

References

1. Dori, Dov (2002). *Object-Process Methodology, A Holistic Systems Paradigm*, Springer-Verlag Berlin Heidelberg, New York, NY
2. Dori, Dov (2016). *Model-Based Systems Engineering with OPM and SysML*, Springer-Verlag Berlin Heidelberg, New York, NY
3. Crawley, Edward; Cameron, Bruce; Selva, Daniel (2016). *System Architecture: Strategy and Product Development for Complex Systems*, Pearson Higher Education Inc, Hoboken, NJ
4. OPCAT Version 3.0 Getting Started Guide. Retrieved from https://web.mit.edu/deweck/Public/opcat/OPCAT_Manual_3.0.pdf
5. ISO 19450:2024 https://www.iso.org/standard/84612.html