Chapter 1

A Taste of OPM

> *...We're limited only by our imaginations and what we think we can make computers do.*
>
> Ben Kovitz (1998)

Imagine that an intelligent, friendly extra-terrestrial creature has just landed on planet Earth, without any prior knowledge of earthly physical and societal systems. Trying to figure out what his senses are telling him and how to construct a model of the surrounding reality, this creature is in a situation not unlike that of a human system developer, who is beginning to evolve a new system in an unfamiliar domain. Lacking sufficient knowledge about the domain that hosts the system and its environment, the analyst must start with an arbitrary collection of facts. The collected observations are not overly refined nor are they extremely abstract. The knowledge and understanding of the system is gradually improved through activities such as observing the current state of affairs and practices, inquiring, interviewing professionals in the field, and reading relevant documents.

Soon enough, details about the system begin to mount. An intuitive yet formal way of documenting this growing amount of collected information must be in place. Subsequent, higher order cognitive activities follow, including understanding, modeling, analyzing, designing, presenting and communicating the analysis findings and design ideas. These mental activities rely heavily on a solid infrastructure for organizing the accumulated knowledge and creative ideas. Long before the system becomes particularly intricate, it is very helpful to display these results in a medium other than the brain.

Systems and products are becoming increasingly complicated. Technology has been so pervasive that even commonly used products feature high computational power, embedded within increasingly miniature, precise and involved hardware. Systems of an infrastructure nature, such as air traffic control, the Internet, and electronic economy, are orders of magnitude more complex than products individuals normally use. Understanding natural, artificial, and social systems requires a well-founded, yet intuitive methodology that is capable of modeling these complexities in a coherent, straightforward manner. The same methodology should be useful for designing new systems and improving existing ones. These systems are intricate enough as it is; there is no need to add confusion by using a method that is itself complex.

Artificial systems require development and support efforts throughout their entire lifecycle. Systematic specification, analysis, design and implementation of new systems and products are becoming ever more challenging and demanding, as contradicting requirements of shorter time-to-market, rising quality, and lower cost, are on the rise. These trends call for a comprehensive methodology, capable of tackling the mounting challenges that the evolution of new systems poses. The development of Object-Process Methodology was motivated by this call.

The idea of developing systems in a unified frame of reference is not new. For example, Bauer and Wössner (1981) noted that systematic development of basic concepts leads to methods that cover the entire system's lifecycle. Object-Process Methodology, or OPM for short, takes a fresh look at modeling complex systems that comprise humans, physical objects and information. OPM is a formal paradigm to systems development, lifecycle support, and evolution. It caters to people's intuition and train of thought. Preliminary ideas of OPM were expressed in (Dori, 1995; 1996) and applied in such diverse areas as computer integrated manufacturing (Dori, 1996A), image understanding (Dori, 1996B), modeling research and development environments (Meyersdorf and Dori, 1997), algorithm specification (Wenyin and Dori, 1999), document analysis and recognition (Dori, 1995A; Wenyin and Dori, 1998), and modeling electronic commerce transactions (Dori, 2001).

Natural and artificial systems alike exhibit three major aspects: function (what these systems do), structure (how they are constructed), and behavior (how they change over time). Since OPM does not make any assumptions regarding the nature of the system in question, it can be applied in any domain of human study or endeavor. In the case of artificial systems, it provides a framework for the entire system's lifecycle, from the early stages of requirement elicitation and analysis, through further development and deployment, all the way to termination and initiation of a new generation.

OPM combines formal yet simple graphics with natural language sentences to express the function, structure, and behavior of systems in an integrated, single model. The two description modes OPM uses are semantically equivalent, yet appeal to two different parts of the brain, the visual and the lingual. OPM is a prime vehicle for carrying out the tasks that are involved in system development. It does so in a straightforward, friendly, unambiguous manner. The design of OPM has not been influenced by what current programming languages can or cannot do, but rather, what makes the most sense. Due to the resulting intuitiveness, OPM is communicable to peers, customers and implementers. At the same time, the formality of OPM makes it amenable to computer manipulation for automatically generating large portions of the conceived system, notably program code and database schema.

Where does the name Object-Process Methodology come from? Objects and processes are the two main building blocks that OPM requires to construct models. A third OPM entity is *state*, which is a situation at which an object can be and therefore a notch below object. Objects, processes and states are the only bricks involved in

building systems. The links connecting these three entities act as the mortar that holds them together.

As for methodology, Computer Desktop Encyclopedia (2001) defined it as "the specific way of performing an operation that implies precise deliverables at the end of each stage." OPM specifies a way of understanding and developing systems. Its deliverables include a set of Object-Process Diagrams (OPDs) and a corresponding collection of sentences written in Object-Process Language (OPL). Three more general definitions of the term "methodology" are found in The American Heritage Dictionary (1996):

(a) A body of practices, procedures, and rules used by those who work in a discipline or engage in an inquiry; a set of working methods
(b) The study or theoretical analysis of such working methods
(c) The branch of logic that deals with the general principles of the formation of knowledge.

OPM contains important elements of each of these three definitions.[*] These elements are treated and discussed throughout the book. This chapter introduces OPM's basic principles and features and demonstrates its modeling power using an example of a wedding.

1.1 The Wedding Example: A Sneak Preview of OPM

To gain familiarity with the basic ideas of OPM and its two methods of representation, we will describe a wedding system at increasing levels of complexity. This example introduces some of the most important concepts that OPM is built upon, such as the independence of processes, their effect on objects, and the relations among objects and processes. Step by step, we will create an OPD (Object-Process Diagram) of this system and its corresponding set of OPL (Object-Process Language) sentences. In this chapter, each OPD is accompanied by a legend that introduces OPM's commonly used symbols.

1.2 OPM Building Blocks: Objects, Processes, and States

OPM is built of just three types of entities: objects, processes and states, with objects and processes being higher-level building blocks. Objects exist, and processes

[*] Definition (a) above overlaps with The American Heritage Dictionary (1996) definition of "method," as a means or manner of procedure, especially a regular and systematic way of accomplishing something; the procedures and techniques characteristic of a particular discipline or field of knowledge. However, since OPM contains the additional elements of theoretical analysis methods and formation of knowledge, "methodology" is more appropriate.

transform the objects by generating, consuming, or affecting them. States are used to describe objects, and are not stand-alone things.

The symbols for objects and processes are rectangles and ellipses, respectively. The name of the object or process is recorded inside the corresponding symbol. Figure 1** shows **Person** as an example of an object and **Marrying** as an example of a process. The first letter in object and process names is always capitalized. To differentiate objects from processes, process names end with the suffix **ing**, indicating that they are active, dynamic things.

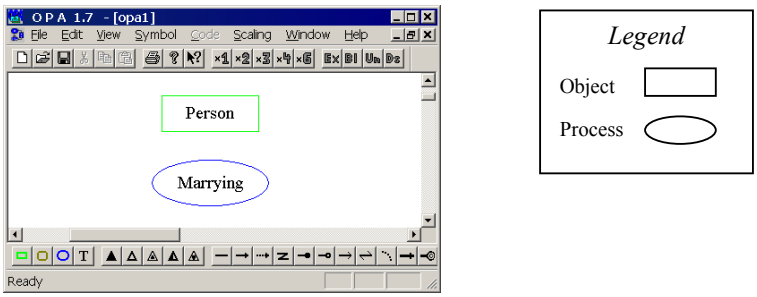

Figure 1. The object and process symbols applied to **Person** and **Marrying**, respectively

At any point in time, each object is at some *state*. In our example, we restrict the possible states of **Person** to be **single** or **married**. The OPD in Figure 2 shows the symbol of state as a rounded corner rectangle, or "rountangle" for short,* enclosing the state name, which starts with a lower-case letter. The state symbol is drawn inside the rectangle symbolizing the object that "owns" the state. The sentence describing Figure 2 in Object-Process Language (OPL) lists the possible states of the object:

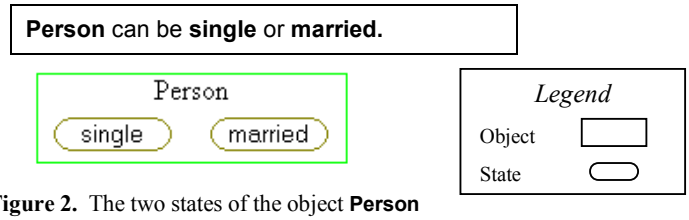

Figure 2. The two states of the object **Person**

** The OPDs in Figure 1 and in the rest of this book were drawn using OPCAT – Object-Process CASE Tool. This Computer Aided Software Engineering (CASE) tool supports OPM by drawing OPDs and checking the legality of the various links. It is downloadable from http://iew3.technion.ac.il/%7Edori/opcat/index-continue.html. To reduce clutter and save space, subsequent OPDs will be presented without the surrounding OPCAT user interface that appears in Figure 1.

* This word is due to D. Harel.

The **bold** words in the OPL sentence above, such as **Person**, denote *non-reserved words*, whereas the non-bold words, such as can be, are *reserved words*, which are part of the sentence structure. Any OPL sentence consists of non-reserved words – domain-specific words, which the system architect uses for the specific system – and reserved words, which link the non-reserved words and provide for creating a natural language sentence. In a way, the reserved words glue the system-specific terms (the non-reserved words) together in a meaningful way. OPL sentences are certainly far more readable than a script of any computer programming language. These sentences may not be the most natural, nor grammatically perfect. However, as we shall see, they are carefully designed to convey a clear and straightforward meaning through well-phrased and humanly understandable constructs.

A process can transform an object in three different ways: by creating it, by destroying it, or by affecting it in some way. When a process affects an object, it changes the state of that object. Links within the OPD express this graphically. Figure 3 contains two such links, ⟶▷, arrows with elongated hollow triangular heads. An *input link* goes from the input state to the affecting process, and an *output link* goes from that process to the output state. In our example, **single** is the input state. It is linked with an input link to the process **Marrying**. From **Marrying**, an output link leads to **married**, the output state. The appropriate OPL sentence follows.

Marrying changes **Person** from **single** to **married.**

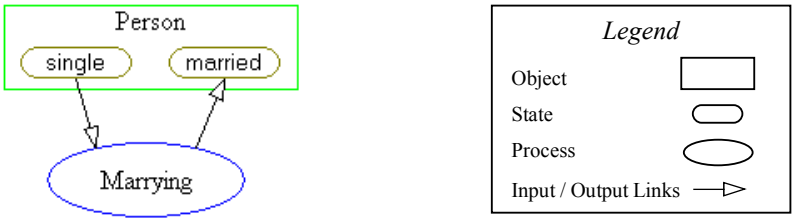

Figure 3. The process **Marrying** affects the object **Person** by changing its state from **single** to **married**.

Note that the sentence has the same meaning as the combination of the two links in Figure 3, from the **single Person** through **Marrying** to the **married Person**. These links are procedural links. Procedural links connect processes to objects or to states of objects (as is the case here) in a variety of ways.

1.3 Specialization and Inheritance

Specialization is the relation between a general thing and a type of that thing. The process **Marrying** depends upon a **Woman** and a **Man** getting married.* **Man** and **Woman** are different types, or specializations, of **Person**. The white equilateral triangle (△) in Figure 4 denotes this. This link is a structural link, which is fundamentally different than the procedural links we used in Figure 3. Structural links relate objects to other objects, and processes to other processes, but not objects to processes. The same semantics that the new link expresses in Figure 4 is expressed in the following OPL sentence:

Man and **Woman** are **Persons.**

This sentence demonstrates the differences between natural, spoken language and OPL. A more natural sentence would be "**Man** and **Woman** are **People**." But OPL in its basic form does not handle such exceptions. Yet, the sentence is still clear and comprehensible.

Since **Woman** and **Man** are specializations of **Person**, they *inherit* the states of **Person**. We have seen that **Person** can be in one of two states: **single** or **married**. Figure 4 expresses that both **Woman** and **Man** are specializations of **Person**. Through inheritance, each can be **single** or **married**. We have seen that **Marrying** affects **Person** by transforming it from **single** to **married**. Since **Woman** and **Man** are specializations of **Person**, they inherit not only the states of **Person**, but also the transformation from **single** to **married** that **Person** undergoes. This effect is inherited to both **Woman** and **Man**: **Marrying** affects each one of them by transforming her or him from **single** to **married**.

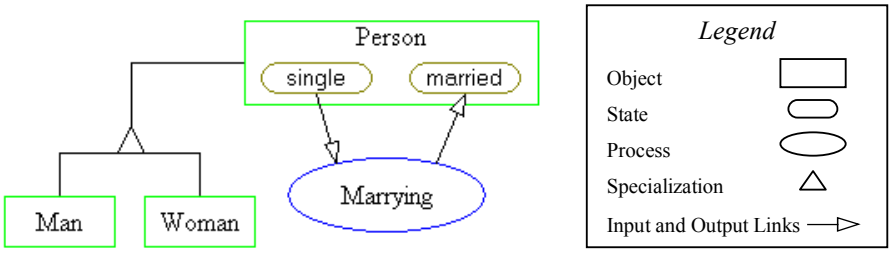

Figure 4. Man and **Woman** as specializations of **Person**

* While the wedding here is traditional in the sense that it is between people of different genders, it does not imply any bias against other engagement types.

1.4 Aggregation and the Result Link

The **Marrying** process yields a new object, **Couple**, which consists of **Woman** and **Man**. The aggregation symbol, shown as a black equilateral triangle (▲) in Figure 5, denotes this whole-part relation between the whole, the **Couple,** and its parts, **Woman** and **Man**. Aggregation is another structural link, expressing a different relation than specialization. The arrow from **Marrying** to **Couple** is a *result link*, a procedural link denoting the fact that a new object has been generated as a result of the occurrence of the process. These two links are shown connected to the new object **Couple** in Figure 5, and are reflected by these two OPL sentences:

Couple consists of **Man** and **Woman.**
Marrying yields **Couple.**

The reserved words in these two sentences express the relation type. The phrase "consists of" in the first sentence relates the whole, **Couple**, to its parts, **Man** and **Woman**. The same phrase would be used for any other whole-part relationship, for example: **House** consists of **Walls** and **Roof**. In the same manner, the reserved phrase yields, which, in this example, is a single word, connects a process and its resulting object, as in **Cutting** yields **Slice**. The link from the process **Marrying** to the object **Couple** denotes that **Couple** is the result of the occurrence of **Marrying**. This link is therefore called the result link. As we progress along the book, we will encounter different types of OPL sentences. We will become familiar with the reserved phrases these sentence types use and with the graphical constructs that are equivalent to these sentences.

The result link, the input link and the output link are similar in their use, but not identical. All three are procedural links, connecting processes to objects or states. The input link connects an object's original state with a process. The output link, in turn, connects that process to a different state of the same object; this state is the output state. The result link denotes the creation of an entirely new object as an outcome of the process. While graphically all three look alike, their distinction is inferable from their context, i.e., the combination of their source and destination. The result link originates from a process and ends at an object, the input link from a state to a process, and the output link from a process to a state. This is an example of the context sensitivity of the graphic symbols in an OPD: depending on its context in the OPD, the same symbol can have more than one meaning. And so, the same symbol may serve a few purposes, albeit related ones. Context sensitivity enables a small set of symbols to express rich semantics.

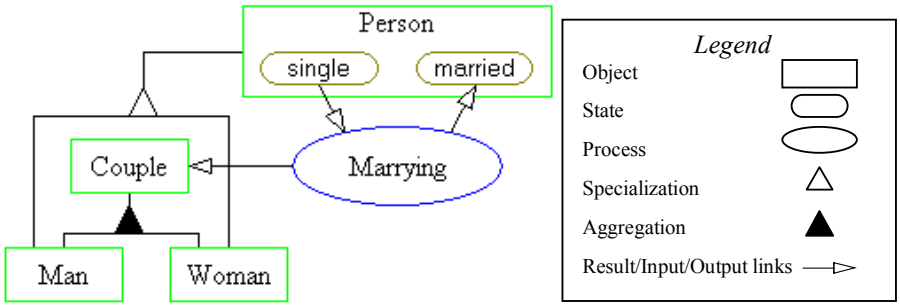

Figure 5. Marrying yields **Couple**, which consists of **Woman** and **Man**, each being a specialization of **Person**.

Person can be **single** or **married.**
Marrying changes **Person** from **single** to **married.**
Man and **Woman** are **Persons.**
Marrying yields **Couple.**
Couple consists of **Man** and **Woman.**

Frame 1. The OPL paragraph of the OPD in Figure 5

Figure 5 is the final OPD of the wedding system. It expresses the same semantics as the OPL paragraph in Frame 1, which is the collection of the OPL sentences in this chapter. Both representations of the wedding system are equally valid, and can be used in conjunction or separately. Moreover, when using the two representations concurrently, one of the modes can compensate for potential misunderstandings in the other. If a reader is not familiar with a particular OPD symbol, for example, the appropriate OPL sentence can explain its meaning.

Summary

Through the wedding example we have been exposed to basic ideas of OPM. The constructed set of Object-Process Language (OPL) sentences and the Object-Process Diagram (OPD) demonstrate that small is beautiful. Using a small set of symbols, we have been able to specify clearly and intuitively, yet formally, exactly what the wedding system is and how it operates. Recognizing processes (**Marrying**, in our example) as independent building blocks alongside objects enables coherent modeling of the system's dynamics. The behavior of the system is integrated with the static object model and complements it.

The two OPM representation modes, the graphic and the textual, provide complementary, crisp representations of the system. The OPL sentences and the OPD complement each other and appeal to the two sides of the brain at once. The single diagram format provides for a concise and economical symbol set with a compact yet semantically rich vocabulary. We saw that three procedural links, the input, output and result links, are expressed by the same symbol.

We have seen the three OPM entities: object, process and state. Objects are things that exist, while processes are things that affect objects. States are situations at which objects can be. Processes transform objects in one of three ways: generating, consuming or affecting them. The effect a process has on an object is manifesting through a change in the object's state. Before the **Marrying** process started, both the **Man** and the **Woman** were **single**, and after it ended, they were **married**. The object **Couple**, which did not exist prior to **Marrying**, was generated as a result of its occurrence. In OPL, the three entity types are also easily distinguishable. Objects and processes are capitalized, states are non-capitalized, and processes end with **ing**.

Two types of links connect entities with each other: *structural links* and *procedural links*. Procedural links, explained in Chapter 5, express the behavior of the system. In the OPD of Figure 5, In the OPD of Figure 5, we saw the three types of procedural links discussed above: result link, input link and output link. Structural links, explained in Chapters 6 through 8, express persistent, long-term relations among objects or among processes in the system. Figure 5 contains two examples of structural links: aggregation and specialization, which are symbolized by a black triangle and a white triangle, respectively. Since the structural and the procedural links are expressed in the same diagram, they provide a complete picture of the system in a single graphic model, which is complemented by a textual one.

Problems

1. In the wedding example, we defined **Person** as having two states, **single** and **married**. Name two other states that might be applicable for **Person** in this context.

2. Draw an OPD of **Person**, with the states you came up with in problem 1 as well as the ones it had before.

3. Write an OPL sentence expressing the OPD in problem 2.

4. Create a system with **Person** and **Divorcing**. Show how the added process affects the object. You may wish to use your answers to problems 1–3.

5. We may want to hide the states of an object. In Figure 6(a) the states of the object **Person** from Figure 3 have been suppressed.

a. Explain the motivation of suppressing the states of an object, as done in Figure 6(a) relative to Figure 3.

b. What happened to the edges of the input and output links relative to Figure 3?

6. In Figure 6(b), we have replaced the input and output links by a single, bidirectional effect link. Which is more efficient in your opinion?

7. What is the relationship between the input and output links and the effect link?

8. Compare the semantics of the effect link in Figure 6(b) with the links in Figure 3. Which conveys more information? Which is more succinct?

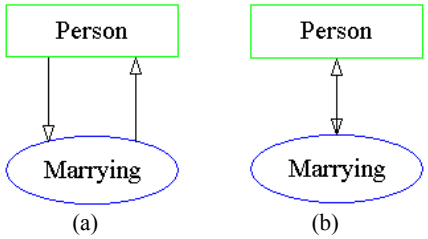

Figure 6. Problem 5. (a) Figure 3 with states hidden. (b) The effect link replaces the two links.

9. A person withdraws money from an ATM using his or her cash card.

a. Identify the objects in this system.

b. Identify the processes in this system.

c. Draw an OPD of the system.

