# Chapter 2
# Text and Simulation Enhancements

> *We went next to the School of Languages. … The first Project was to shorten Discourse by cutting Polysyllables into one, and leaving out Verbs and Participle, because in Reality all things imaginable are but Nouns. … However, many of the most Learned and Wise adhere to the new Scheme of expressing themselves by Things.*
>
> Jonathan Swift, *Gulliver's Travels* (1726)

Winograd and Flores (1987) noted that "Nothing exists except through language… In saying that some 'thing' exists (or that it has some property) we have brought it into a domain of articulated objects and qualities that exist in language."

Indeed, language greatly enhances our ability to understand systems and communicate our understanding to others. This chapter presents two enhancements to OPM models: textual model representation and animated model simulation.

We introduce the object-process language (OPL) as the textual modality of OPM that complements the graphical representation through OPDs (object-process diagrams). We show the equivalence between this graphical specification and the natural language specification through OPL. The chapter also shows another important means of enhancing model understanding: its animated simulation.

## 2.1 OPL: A Subset of English

In Fig. 2.1, an effect link has been added between the **Automatic Crash Responding** process and the **Vehicle Occupants Group** object. This link expresses the fact that the process affected the object; that is, changed it in some way. As soon as the effect link is drawn, OPCAT (the OPM modeling software environment we use) automatically generates the first object-process language (OPL) sentence:

**Automatic Crash Responding** affects **Vehicle Occupants Group.**

This is a clear and unambiguous sentence in plain English that was generated immediately, in response to the modeler linking the two things, and it explains what OPCAT has "understood" from this operation.

Figure 2.1 shows a partial screenshot of OPCAT. The top pane contains the OPD and the bottom contains the OPL sentence generated in response to the modeler's insertion of the effect link.

In OPCAT, the default color of OPL words or phrases (word sequences) that represent objects is green; this is the same color as the corresponding object boxes in the OPD. Likewise, the default color of phrases representing process names in OPL is blue, which is also the color of the corresponding process ellipses.

The graphics-text conversion is a two-way street: just as it was possible to extract the OPL paragraph above from the OPD in Fig. 2.1, that OPD can be reconstructed from its OPL paragraph; that is, the collection of OPL sentences that specify in text what the OPD specifies in graphics. As an exercise, the

© Springer Science+Business Media New York 2016

D. Dori, *Model-Based Systems Engineering with OPM and SysML*, DOI 10.1007/978-1-4939-3295-5_2

graphics-text equivalence principle can be verified by reconstructing Fig. 2.1 versa. It is easier to edit the graphics and get the immediate textual feedback.

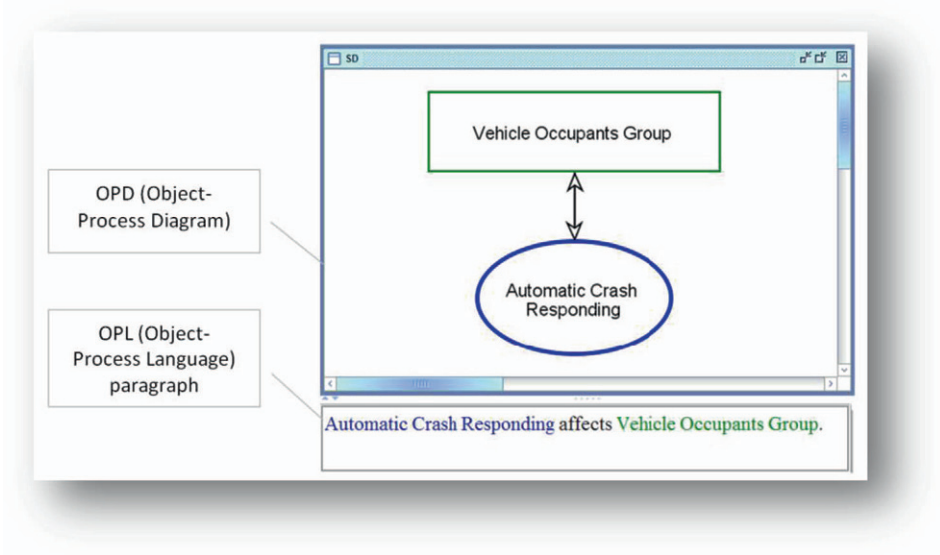


Fig. 2.1 A screenshot of OPCAT showing the OPD on top and the first object-process language (OPL) sentence generated in response to the modeler's insertion of the effect link

The ability to switch back and forth between graphics and text means that OPM system specification writers and modelers are less likely to make costly design errors. Moreover, readers of the textually specified model are more likely to fully comprehend the system and detect mistakes or omissions. The specification reader can fill gaps in his or her understanding of the system that may have formed while examining one modality by looking at the other one. In doing so, the reader reinforces familiarity with the specification and can more easily detect design errors or omissions.

## 2.2 States and Animated Simulation

States are important entities in a conceptual model. In this section, we introduce states and show how an OPM model can be simulated by animation to play out its dynamics, particularly its state transitions. So far, we have modeled the fact that the object **Vehicle Occupants Group** was affected by the process **Automatic Crash Responding**. To identify the actual nature of this effect, we need to be more specific. To this end, at any point in time during which we inspect a certain object, that object may be in a certain situation, which we refer to as a *state*.

> *A **state** is a situation or position at which an object can be, or a value it can assume, for some positive amount of time.*

Since states are possible situations or values *of an object*, they have no "life" of their own; they have meaning only in the context of the object to which they belong and within which they reside.

### 2.2.1 The Effect of a Process on an Object

The *effect* that a process has on an object is a *change in the state* of that object. In other words, the process transforms the object from its *input state* (the state before the process occurred) to its *output state* (the state after the process occurred). Therefore, at any given point in time, a stateful object is in *one of its states* or *in transition between two of its states*.

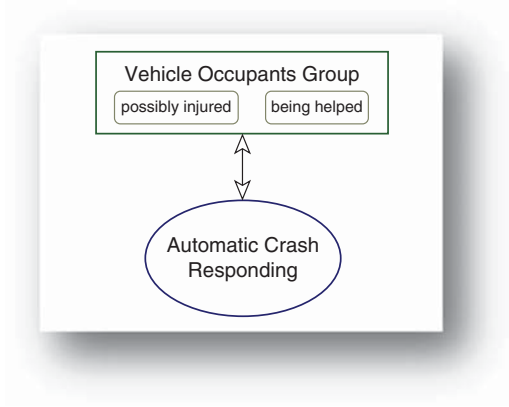


Fig. 2.2 The states of **Vehicle Occupants Group** are added: **possibly injured** is the input state and **being helped** is the output state

In order to be specific about the effect that the **Automatic Crash Responding** process has on the **Vehicle Occupants Group** object, we first need to come up with appropriate names for the states of this object, both before and after the occurrence of the process. Immediately after the crash, the **Vehicle Occupants Group** is **possibly injured**; after **Automatic Crash Responding**, they are **being helped**. These are the corresponding input and output states of the **Vehicle Occupants Group** object with respect to the **Automatic Crash Responding** process.

Figure 2.2 shows the OPD after adding the two states of **Vehicle Occupants Group**, with **possibly injured** being the input state and **being helped**—the output state. The symbol of state is a rounded corner rectangle—rountangle[1] for short—that encloses the state name, which starts with a lower-case letter. The state symbol is drawn inside the rectangle of the object that "owns" the state. This adding of states created the following new OPL sentence:

**Vehicle Occupants Group** can be **possibly injured** or **being helped.**

[1]This term was coined by D. Harel.

OPL sentences in this book are written in **Arial font**. Reserved phrases, such as "can be", are in regular font while all the rest (including commas and periods) are in bold font.

### 2.2.2 From an Implicit Effect to an Explicit State Change

Having specified the two states of the object **Vehicle Occupants Group**, we can now be more specific about the effect—the state change—that the occurrence of the process **Automatic Crash Responding** has brought about. We do this by replacing in Fig. 2.3 the single bidirectional *effect link* with an *input-output link pair* (also called input-output specified effect link), which is a pair of unidirectional arrows, one from the input state to the process and the other from the process to the output state.

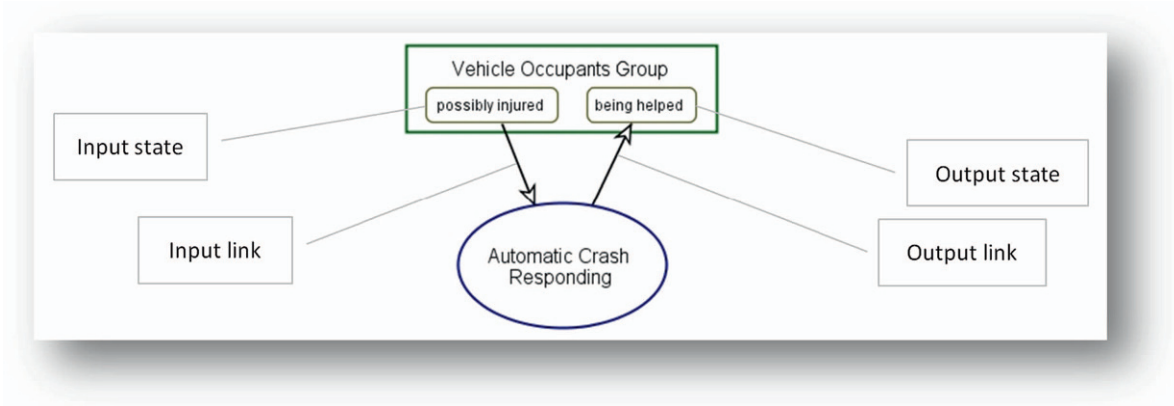


Fig. 2.3 The effect link in Fig. 2.2 is replaced by an input-output link pair that explicates the change of state of the **Vehicle Occupants Group** from the **possibly injured** input state to the **being helped** output state

The arrow from the *input state* to the process is the *input link*, and the arrow from the process to the *output state* is the *output link*. Together, these two links constitute the *input-output link pair*. In response to this editing of the model, the previous OPL sentence, "**Automatic Crash Responding** affects **Vehicle Occupants Group.**", is replaced by the following OPL sentence:

**Automatic Crash Responding** changes **Vehicle Occupants Group** from **possibly injured to being helped.**

By specifying the affecting process and the affected object with its input and output states, this sentence accurately reflects the details of the dynamics of the system.

### 2.2.3 State Naming

The names of states must reflect the various relevant situations in which their "owning" object can be at any given point in time. For example, **off**, **standby**, and **on** are three possible states of a machine, which would lead to the following OPL sentence:

**Machine** can be **off, standby,** or **on.**

As noted, the convention is that, unlike names of things (objects or processes), which have a capital letter at the start of each word, state names are always written in lower case. More descriptive names,

such as "**possibly injured**" or "**being helped**", as in our case study, should be given. It is important to check that the resulting OPL sentence makes sense.

## 2.3 Animated Execution of the OPM Model

One of the most attractive and useful features of an OPM model, which enables it to be visualized and tested, is its executability; that is, the ability to simulate a system by executing its model via animation in a properly designed software environment.[2]

Figure 2.4 shows three stages of executing the OPM model in Fig. 2.3. The screenshot on the left-hand side shows the system before the **Automatic Crash Responding** process occurs. At this stage, **Vehicle Occupants Group** is at its input state, **possibly injured**, which is marked by the state being solid (colored brown).

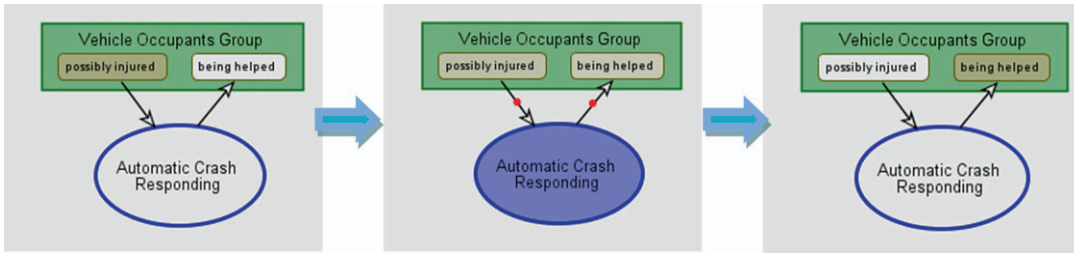


Fig. 2.4 Executing the OPM model shown in the OPD of Fig. 2.3. *Left:* the system before the Automatic Crash Responding process starts. *Center:* the process in action; the object is in transition from its input state to its output state. *Right:* the system after the Automatic Crash Responding process has terminated

The screenshot in the center of Fig. 2.4 shows the process in action, marked as solid (colored blue). During the time that the **Automatic Crash Responding** process is active (that is, when it executes), the object **Vehicle Occupants Group** is in transition from its input state, **possibly injured**, to its output state, **being helped**. This is marked by both states being semi-solid (light brown).

Observing the animation in action reveals that the input state gradually fades out while the output state becomes solid. At the same time, two red dots, shown in the middle of both arrows, travel along the input-output link pair, denoting the "control" of the system; that is, where the system is at each time point. One red dot travels from the input state to the affecting process. At the same time, the second dot travels from the process along the output link to the output state. Finally, the screenshot on the right shows the system after the **Automatic Crash Responding** process had terminated. At this stage, **Vehicle Occupants Group** is at its output state, **being helped**.

The animated execution of the system model has several benefits. Firstly, it is a dynamic visualization aid, which helps both the modeler and the target audience to follow and understand the behavior of the system over time. Secondly, similar to a debugger of a programming language, it facilitates verification of the system's dynamics and spotting of logical design errors in its flow of control. Therefore, it is

[2]You may want to try it yourself if you have modeled the system with OPCAT: Click the "Test System" film icon.

highly recommended that the system model be animated frequently as it is being constructed, so that design errors do not accumulate, but are corrected as soon as they are made.

## 2.4 Summary

- OPM has two equivalent representation modalities: the graphic—object-process diagram (OPD) and the textual—object-process language (OPL).
- The OPL sentences and the OPD complement each other, as they appeal to the parallel visual and verbal cognitive processing channels of the human brain.
- A state is a situation at which an object can be.
- An effect link indicates some state change of the linked object by the linked process.
- An input-output link pair indicates the specific state from and to which the process changes the object.
- An OPM model is amenable to animated simulation, which facilitates understanding the system's dynamic aspect and testing its logical flow.

## 2.5 Problems

Continuing with the Baggage Handling System that you modeled in Chap. 1, let us assume that the current model is presented in Fig. 2.5.

1. Match a corresponding OPL sentence for each link in the OPD of the Baggage Management System in Fig. 2.5.
2. Add the states **unloaded** and **loaded** to the object **Baggage**.
3. Replace the effect link between the process in your model and **Baggage** by an input-output link pair.

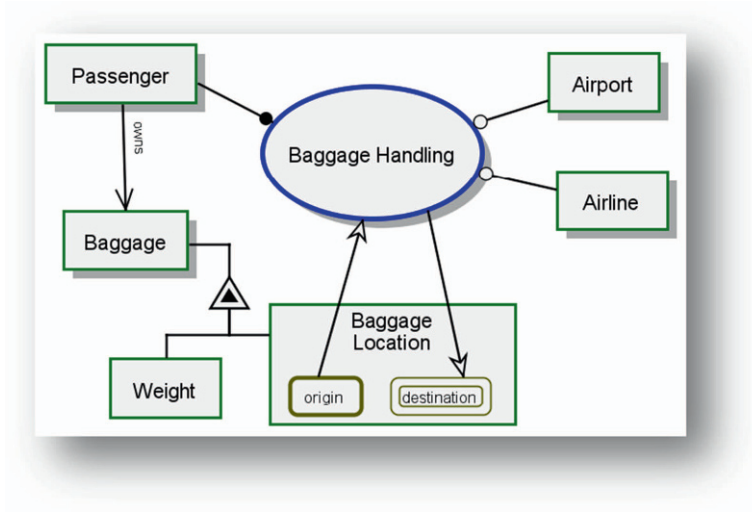


Fig. 2.5 System Diagram—top-level Object-Process Diagram (OPD) of the Baggage Handling System

4. Update the OPL sentence to reflect the change you made.
5. Add to your model and link the remaining objects from the specification above.
6. Using OPCAT, simulate the system by animating it.

http://www.springer.com/978-1-4939-3294-8

Model-Based Systems Engineering with OPM and SysML

Dori, D.

2016, XXII, 411 p. 175 illus., Hardcover
ISBN: 978-1-4939-3294-8