chapter twelve

Conceptual models become alive with Vivid OPM

How can animated visualization render abstract ideas concrete?

Dov Dori, Sergey Bolshchikov, and Niva Wengrowicz

Contents

Introduction

Conceptual modeling is increasingly recognized as a vital stage in the process of developing any complex system. It allows for expressing the meaning of terms and concepts used by domain experts to discuss the problem and to find correct relationships between different concepts (Fowler 1997). Conceptual modeling explains not only the structure of the system under study but also, not less importantly, the dynamic aspect of the system—its behavior. Conceptual models are inherently abstract, requiring higher-order thinking capabilities to overcome the layers of abstraction represented in a conceptual model by the entities with their symbols and relations among them. Such abstract thinking is indispensable for understanding conceptual models, but it is removed from facts of the *here and now* and from concrete examples of the concepts being considered.

Marton (1981) made a fundamental distinction between a first-order perspective, which pertains to describing various aspects of the world, and a second-order perspective, aimed at describing various aspects of people's experiences of various aspects of the world. Analogously, at a different level—the conceptual model level—we distinguish between the conceptual model itself and the way people understand and interpret it. While abstract thinkers have the cognitive tools to process well-prepared models with relative ease, concrete thinkers require a certain amount of scaffolding to facilitate the understanding of conceptual models.

Abstract thinkers can reflect on ideas and relationships separate from the objects that share those relationships. For example, a concrete thinker can think about this particular object, whereas an abstract thinker can think about the class of those objects. This difference in abstraction capabilities poses a challenge for the majority of people, who are not used to conceptual modeling and whose abstract thinking is not geared for examining and comprehending such models. This is the main motivation behind the development of the conceptual model visualization and concretization mechanism, Vivid object-process methodology (OPM), which we developed and assessed, as described in this chapter.

Background

Object-process methodology

OPM (Dori 2002) is a compact holistic conceptual modeling approach to the representation and development of complex systems, which is simple and intuitive while also being formal. OPM, the emerging ISO 19450 standard, is a formal yet intuitive paradigm for systems architecting, engineering, development, life cycle support, communication, and evolution. The application of OPM ranges from simple assemblies of elemental components to complex, multidisciplinary, dynamic systems. OPM was originally aimed for use by information and systems engineers for knowledge management and representation of multidisciplinary man-made sociotechnical industrial and information systems. OPM notation supports conceptual modeling of systems. Its holistic approach includes refinement mechanisms using a single kind of diagram and equivalent text to describe the functional, structural, and behavioral aspects of a system.

OPM provides two semantically equivalent modalities of representation for the same model: graphical and textual. A set of interrelated object-process diagrams (OPDs) constitutes the graphical model, and a set of automatically generated sentences in a subset of the English language constitutes the textual model expressed in the object-process language (OPL). In a graphical-visual model, each OPD consists of OPM elements, depicted as graphic symbols. The OPD syntax specifies the consistent and correct ways to manage the arrangement of those graphical elements. Using OPL, OPM generates the corresponding textual model for each OPD in a manner that retains the constraints of the graphical model. As OPL's syntax and semantics are a subset of natural English, domain experts can easily understand the textual model.

OPM has formal syntax and semantics, and this formality serves as the basis for model-based systems engineering. An OPM model consists of a set of hierarchically structured OPDs, whose arrangement mitigates a systems' complicated expression. Each OPD is obtained by refining (in-zooming or unfolding) of a thing (object or process) in its ancestor OPD, thus providing a more elaborated context for that thing. Any diagram can include new things and display things from other diagrams, where some or all of the details, which are unimportant in the context of the specific diagram, are not presented. It is sufficient for some detail to appear once in some OPD for it to be true for the system in general even though it does not appear in any other OPD.

The domain-independent nature of OPM enables the entire scientific and industrial community to model, develop, investigate, and analyze systems in their domains, providing for people with different skills and competencies to employ a common intuitive yet formal framework. The OPM approach provides a framework for system lifecycle management. In particular, it facilitates a common view of the system under construction, test, integration, and daily maintenance, facilitating a multidisciplinary environment. Using OPM, companies can improve their overall, big-picture view of the system's functionality, flexibility in assignment of personnel to tasks, and managing exceptions and error recovery.

OPM is founded on two elementary building blocks. These are (1) stateful objects—things that exist, possibly at one of their states, which represent the system's structure and (2) processes—things that happen to objects and transform them, representing the system's behavior. Processes transform objects by (1) creating them, (2) consuming them, or (3) changing their states. The two semantically equivalent modalities, graphical and textual, specify each OPM model. The graphical modality is a set of one or more self-similar and interrelated OPDs. The textual modality of the model is a corresponding set of sentences in OPL—a subset of natural English. This is a verbal mirror image of the graphical model that can facilitate its comprehension by nonexpert viewers, as all it requires is understanding simple English. In an appropriate software environment, such as the Object-Process CASE Tool (OPCAT) (Dori et al. 2010), the graphical model is automatically translated into a textual model while it is being developed and edited by the modeler.

OPM things: Objects and processes

Elements, the basic building blocks of any system modeled in OPM, are of two kinds: things and links. The modeling elements object (possibly in a specific state) and process are OPM things. The modeling element link designates an association between two things.

An OPM object is a thing that exists or can exist once constructed, physically or informatically. Associations among objects are structural relations. They constitute the object structure of the system being modeled, that is, the static, structural aspect of the system. Objects may exist temporally. Once created, they exist in their entirety, until their existence terminates by consumption (disappearance, destruction, or elimination).

An OPM process is a pattern of object transformation—a thing that expresses the behavioral, dynamic system aspect: how processes transform objects in the system and how the system functions to provide value. Processes transform objects by creating them, consuming them, or changing their state. Thus, in an OPM system model, processes complement objects by providing the dynamic, procedural aspect of the system.

A nontrivial process comprises a hierarchical network of subprocesses. Every level of the process hierarchy induces a partial order on the processes, that is, some processes are sequential, such that one process must end before one or more other processes start, while other processes may occur in parallel or as alternatives. A function is a process that provides value, that is, benefit at cost. At any level in the process hierarchy, a process in a system should provide value as part of its ancestor process.

Emergence, created by combining objects and processes, gives rise to a function that exceeds, and is not a simple sum of, the functions of its parts. The structure-behavior combination of the system, which attains its function, is the system architecture. The underlying idea behind the embodiment of the system architecture is the system concept.

We know of the existence of an object when we can name it and refer to its unconditional, relatively stable existence. However, the object's transformation, that is, its creation, change over time, or consumption, cannot occur unless a process acted on that object.

An object state is a particular situation at which an object can be at some point during its lifetime. At any given point in time during the life of an object, the object is in one of its states or in transition between two of its states.

Unification of function, structure, and behavior

The OPM structure model of a system is an assembly of the physical and informatical (logical) objects connected by long-lasting associations among them, called structural relations. During the lifetime of a system, creation and destruction of aggregation-participation relations—a kind of structural relations—may occur.

The OPM behavior model of a system, referred to as its dynamics, reflects the mechanisms that act on the system over time to transform systemic (internal to the system) and/or environmental (external) objects. At any point in time, the state of a system is the aggregation of the states of its constituent objects and processes. The combination of system structure and behavior—the system's architecture—enables the system to perform a function, which is the value the system delivers to at least one stakeholder, who is the system's beneficiary.

OPM integrates the functional (utilitarian), structural (static), and behavioral (dynamic) aspects of a system into a single, unified model, which is nonetheless expressed bimodally in both graphics and text. In each OPD of the OPM model, structure and behavior can coexist without highlighting one at the expense of suppressing the other. Maintaining focus from the viewpoint of overall system function, this structure-behavior unification provides a coherent single frame of reference for understanding the system of interest, enhancing its intuitive comprehension while adhering to formal syntax and deep semantics.

Function-as-a-seed OPM principle

The top-level value-providing process of a modeled system expresses the function of the system as perceived by the system's main beneficiary or beneficiary group. Modeling with OPM begins by defining, naming, and depicting the function of the system as its top-level process. The structure and behavior of the system emerges from this function. Identifying and labeling this top-level process, the system's function, is a critical first step in the methodology part of OPM, which prescribes how an OPM model should be constructed. An appropriate function name clarifies and emphasizes the central goal of the modeled system and the value that the system is expected to provide for its main beneficiary. Such a deliberation, which often provokes a debate between the system architecture team members at this early stage, is extremely useful, as it exposes differences and often even misconceptions among the participants regarding the system that they set out to architect, model, and design.

By using a single holistic and hierarchical model for representing structure and behavior, clutter and incompatibilities can be significantly reduced even in highly complex systems, thereby enhancing their comprehensibility. OPM has proven to be better in visual specification and comprehension quality for representing complex reactive systems compared to object model template, a unified modeling language (UML) predecessor (Peleg and Dori 2000). OPM is supported by OPCAT (Dori et al. 2003; Dori et al., 2010), a software environment that is used in this work to model the transcription case study presented in the Section "Experimental design: The Vivid OPM evaluation system." OPM operational semantics are defined by a translation into a state transition system (Perelman et al. 2011), and a related OPCAT simulation environment was developed (Yaroker et al. 2013). OPM is in final stages of ratification as ISO Standard 19450. OPM main elements with their semantics and examples from the biology domain are presented in Table 12.1.

OPM operational semantics

The OPCAT simulation environment supports concurrent, synchronous, and discrete time execution. The execution we used in this work is qualitative in nature with one instance defined for each object (e.g., molecule) and each process. This enables analyzing the behavior and qualitative underlying mechanisms of the system under study. Although multiple instances can be defined in OPCAT simulation and quantitative aspects can be inspected, these are out of the scope of this work.

Processes are executed in a synchronous manner, one after the other, according to a defined timeline. The default timeline, within the context (in-zoomed frame) of each process, is from top to bottom. Alternative scenarios or loops, which override the default timeline, can be modeled using an invocation link. Concurrency is supported, and processes whose ellipse topmost points are located at the same height in the diagram are executed concurrently.

Table 12.1 shows the OPM entities with their symbols, definitions and operational semantics. An example from the biology domain is added, based on the work by Somekh et al. (2012).

Table 12.2 shows OPM procedural links: links connecting an object or its state with a process, including their symbols, definitions, and operational semantics. Table 12.3 shows OPM structural links: links connecting an object with an object or a process with a process, including their symbols, definitions, and operational semantics.

Each process has a (possibly complex) precondition and a postcondition. A process is triggered (attempted to be activated according to its place in the timeline), and its precondition is then checked. Only if the precondition is satisfied, the process is executed. On normal process termination, the postcondition must hold. The precondition of a process is

Table 12.1 OPM Entities with Their Symbols, Definitions, and Operational Semantics

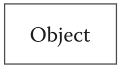
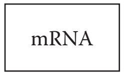
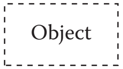
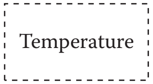
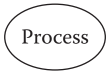
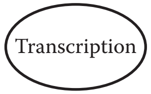
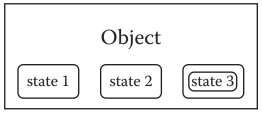
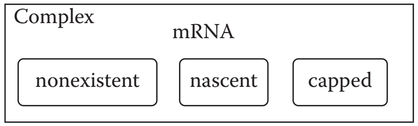
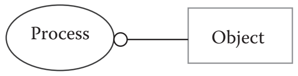
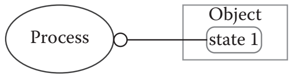
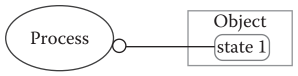
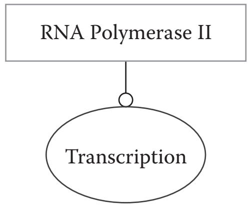
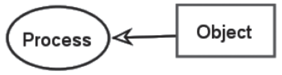
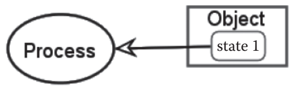
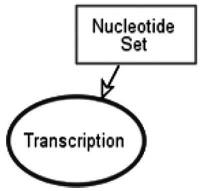
Entity name	Entity symbol	Definition and operational semantics	Biological example
Object: Systemic		An object that consists of a matter or a piece of information	 mRNA is a molecule
Object: Environmental		An object that is external to the system, randomly generated	 Signals and environmental effects on system
Process		A pattern of transformation that objects undergo	 Transcription is a biological process
State: Initial/Regular/ Final		An **initial** (state 1)/**regular** (state 2)/**final** (state 3) situation at which an object can exist for a period	 mRNA has three states: nonexistent (initial state), nascent, and capped

Table 12.2 OPM Procedural Links: Links Connecting an Object or State with a Process

Link name	Link symbol	Definition and operational semantics	Biological example
Instrument/ Condition link represents precondition	 (A)  (B)  (C)	A link denoting a condition required for a process execution. The condition can be the existence of an object (A) or the existence of an object in some state (B). The condition is checked when the process is triggered.(C) When a "c" inside the circle is added, then if the condition does not hold, the process is skipped and the next processes (if any) try to execute. (A), (B) if the condition does not hold, the systems halt (a mode used for checking model consistency)	 The RNA polymerase II being existent is a precondition for transcription process to be executed
Consumption link represents precondition and postcondition	 (A)  (B)	A link denoting that a process consumes an object (A) or an object at some state (B). The object (A) or object's state (B) existence is a precondition for process execution. Their nonexistence is the process postcondition	 The nucleotide set being existent is a precondition for transcription process to be executed. After execution nucleotide set is consumed (nonexistent)

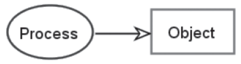
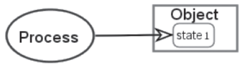
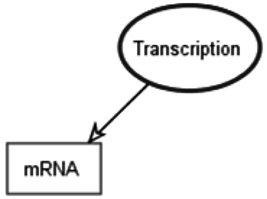
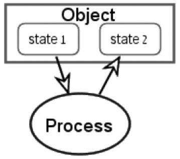
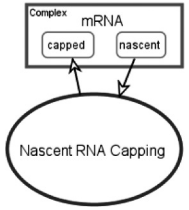
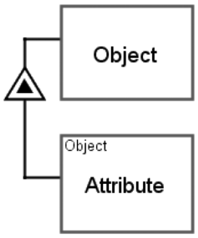
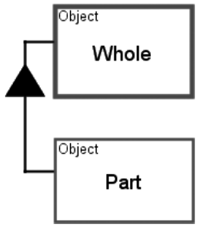
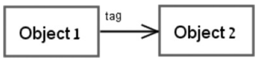
Creation link represents postcondition	 (A)  (B)	A link denoting that a process creates an object (A) or an object at some state (B). Their existence is the process postcondition	 Transcription process execution postcondition is the mRNA being created (existent)
Changing object state links (input and output links) represents precondition and postcondition		Links denoting that a process changes an object from state 1 to state 2. The input link consumes state 1 and the output link creates state 2	 The mRNA nascent state is a precondition for the nascent RNA capping process execution. The change to the capped state is a postcondition for the process execution

Table 12.3 OPM Structural Links: Links Connecting an Object with an Object

Characterization		A fundamental structural relation representing that an element exhibits an attribute object
Participation (consists of)		A fundamental structural relation representing that an object (whole) consists of one or more objects (part[s])
General structural link		A unidirectional association between objects that holds for a period, possibly with a tag denoting the association semantics

expressed by its preprocess object set—the set of objects, which must exist, some possibly in specific states, for the process to start. The postcondition is defined similarly by the postprocess object set.

Logical expressions (AND, OR, XOR) between objects in the pre- and postprocess object set can be defined. By default, the logical relation between the objects in the pre- or postprocess object set is a logical AND, meaning that all the objects in the preprocess object set must exist in their defined states for the process precondition to be true. It is possible to change this default definition by using the XOR and OR relation between various objects. Process execution can also depend on random signals. To model this, we connect the process to an environmental object without or with a specific state. This environmental object is added to the preprocess object set, defining the process precondition. OPM semantics also include event links, for modeling reactive systems, and time exception links.

Unified modeling language

The UML (Fowler 2004; Object Management Group 2014) is a standardized visual specification language for object modeling. UML is software oriented and supports the object-oriented paradigm. The language comprises a graphical notation with strong aspect-separated views used to create an abstract model of a system. Even the behavioral aspect in UML is spread over several different types of diagrams: sequence, collaboration, activity, and statecharts, based on Harel's work (1987). This separation makes it difficult to comprehend the system's overall picture and to keep the different views consistent. Both the numerous types of diagrams (UML 2 has 13) and the software orientation make the notation hardly appropriate as a basis for the task at hand of translating a conceptual model to an animated clip.

xUML (Mellor and Balcer 2002) is an executable modeling language that profiles UML with a semantics extension called action specification language (ASL). The extension aims to capture the language determinism and ensure its executability. Several vendors support their own ASL, but no standard semantics has been accepted.

Petri nets

Petri nets (also known as a place/transition nets or P/T net) is one of several mathematical representations of discrete distributed systems. As a modeling language, it graphically depicts the structure of a distributed system as a directed bipartite graph with annotations. As such, a Petri net has place nodes, transition nodes, and directed arcs connecting places with transitions. Petri nets do not describe explicitly the data flow or objects in the model and this characteristic makes it inappropriate for the general task of conceptual modeling.

IBM Rational Rhapsody

IBM Rational Rhapsody lets systems engineers capture and analyze requirements quickly and then design and validate system behaviors. A Rational Rhapsody systems' engineering project includes the UML and SysML and allows the following (IBM Corporation 2009):

- Performing system analysis to define and validate system requirements
- Designing and specifying the system architecture
- Systems analysis and design
- Software analysis and design
- Software implementation
- Validation and simulation of the model to perform detailed system testing

Rational Rhapsody enables the visualization of the conceptual model via simulation. Simulation is the execution of behaviors and associated definitions in the model. To simulate a model, it requires statecharts, activity diagrams, and textual behavior specifications, which capture the behavior of the model. Structural definitions such as blocks, ports, parts, and links are used to create a simulation hierarchy of subsystems. The preparation for the simulation is relatively complicated process containing the following steps:

- Creating a component
- Creating a configuration for the component
- Generating the component's code
- Building the component application
- Simulating the component application

MagicDraw

MagicDraw is the business process, architecture, software, and system modeling tool with teamwork support, based on UML 2. It is designed for business analysts, software analysts, programmers, quality assurance engineers, and documentation writers, and provides analysis and design of object-oriented systems and databases (No Magic 2013b).

MagicDraw has Cameo Simulation Toolkit, a separate add-on providing simulation and animation capabilities. It extends MagicDraw to validate system behavior by executing, animating, and debugging UML 2 State machines and activity diagrams in the context of realistic mock-ups of the intended user interface (No Magic 2013a). These industrial solutions have the following disadvantages:

- Backed by UML, they provide for simulation and animation of software and hardware system models, but not of conceptual models of systems in general.
- They require several types of UML diagrams (activity and statecharts as a minimum) to operate, compared with OPM, which has only one type of diagram.
- These solutions require a complex sequence of steps to build the animation.

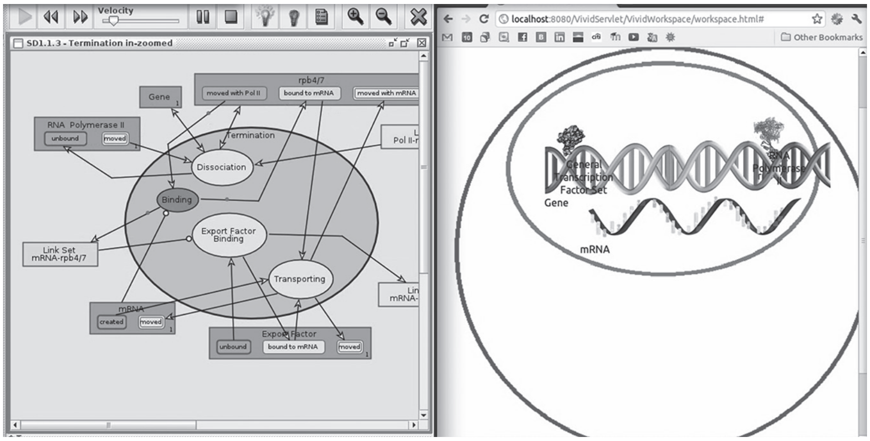
Vivid OPM

OPM explicitly addresses the system's dynamic-procedural aspect, which describes how the system changes over the time. The single OPM model provides for clear and expressive animated simulation of the OPM model using OPCAT, which greatly facilitates design-level debugging. However, the resulting model is basically static, and as such, it does not fully reflect the behavior of the system being architected or designed. The simulation module provides animation and enables visual verification of the OPM model, but the animation is at the abstract, conceptual level. It shows the model entities—process ellipses, object boxes, and state rountangles—changing colors as processes transform objects while red dots slide along the links. The underlying model is still conceptual; it represents concepts as animated geometric shapes. Humans, however, grasp moving pictures of the real things in the model much more intuitively than looking at their symbols. The more visual and dynamic a model, the more intuitive and deeper is humans' understanding of the system.

Motivation and benefits

To enhance complex system dynamics comprehension, we wish to decrease the abstraction level of models of those systems and bring the dynamic aspect closer to reality by actually playing what the conceptual model expresses formally but not intuitively enough.

Our conjecture when setting up to design and implement Vivid OPM was that creation of a visual dynamic model from a static one, which represents changes of objects in space along time by mimicking the actual system's behavior, is likely to enhance comprehension of the system's behavior without requiring knowledge of any specific modeling language. To realize this visual dynamic model rendering, we have developed Vivid OPM—a software module that takes an OPM conceptual model and plays a video clip of the dynamics of the system under development or research. The produced video clip does not require any knowledge of modeling language or technical background, which might prevent stakeholders from understanding the conceptual model. Figure 12.1a depicts a screenshot of OPCAT simulation in action. Figure 12.1b shows the corresponding Vivid OPM



(a) (b)

Figure 12.1 Vivid OPM animation with OPCAT simulation. Part (a) describes the OPCAT simulation and part (b) describes the Vivid animation.

animation, which is driven by the OPM model in Figure 12.1a. This figure demonstrates the drastic difference in what it takes a biologist to understand the conceptual model and its visualization.

Architecture

Vivid OPM (Bolshchikov 2011; Bolshchikov et al. 2011) is a software program that uses a conceptual OPM model of a system to generate an animation of the systems, bringing the conceptual model closer to reality. As noted, understanding a Vivid OPM animation does not require knowledge of OPM or any other modeling methodology.

Vivid OPM comprises three main parts (see Figure 12.2). On the one side is OPCAT—the OPM-based systems modeling environment with built-in simulation and the Vivid OPM plug-in that controls the simulation. On the other side is the visual web-based client—a GUI platform that visualizes the system's dynamic aspect driven by the OPCAT-resident OPM conceptual model of the system. In the middle is a Tomcat Apache server, which has two purposes: (1) it provides servlets, which are Java classes used to extend capabilities of an Oracle server (Oracle 2013), for the client side to upload files and run the animation; (2) it exchanges messages between the client side and OPCAT regarding the animation status.

Two communication channels connect the system's three parts. The client side is connected with the Apache Tomcat server via asynchronous http requests using a polling strategy for animation. The client side sends periodically an update request to the server. The server side and OPCAT talk over Java Messaging Service, which enables exchanging messages between Java applications.

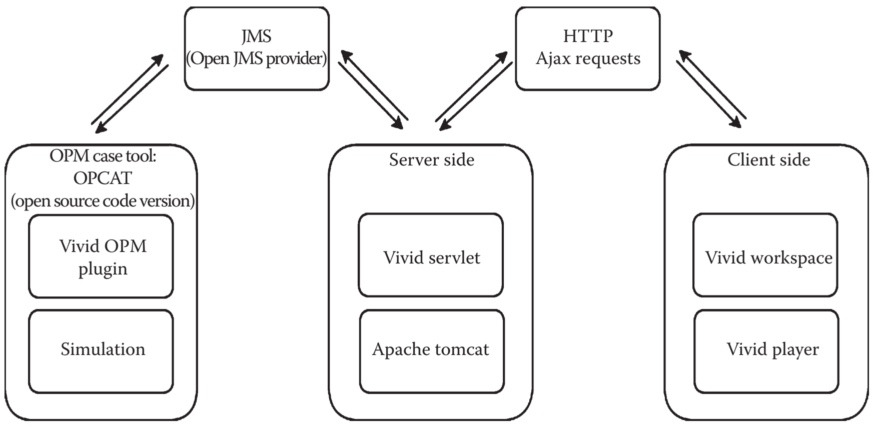


Figure 12.2 Architecture of Vivid OPM.

Vivid OPM modeling workflow

The workflow of Vivid OPM consists of two stages: configuration and animation. Preparing for animation requires configuration to provide missing information, which is not specified in the conceptual model, for example, object images and their initial spatial layout.

The configuration stage starts as the user uploads the OPM model file to the server for which the Vivid OPM component needs to be added. The Vivid OPM script then extracts all the objects in the model from the file and displays them on a canvas. The modeler can then upload a desired icon for each object. By default, all the states of an object have the same icon, but this can be changed so there would be a different icon for each state. Using a dragging mechanism, the user specifies the spatial arrangement for an object's states. As soon as the configuration is ready, it is saved into a separate file on the server. If a model has been previously configured, Vivid OPM will read the file instead of parsing the model again. Thus, the configuration step needed to be done only once per model, and later on the file can be reused.

Once the configuration is completed, Vivid OPM can be launched for animation. The OPM model and the Vivid OPM animation operate simultaneously. Triggering the Java-based simulation process in OPCAT on the server side, Vivid OPM periodically executes update requests arriving from the client side. On receiving an update from the server, a Java plug-in pauses the simulation of OPCAT for Vivid OPM to complete the current animation step. The animation executes by transitioning an object between its states. The JavaScript code executes over time the spatial transition effect: moving of an object, resizing it, changing its color, and so forth. Whenever an object's graphical transition is completed, the client side notifies the server to proceed with the Java-based simulation execution of OPCAT. This synchronization mechanism ensures that the simulation of OPCAT is aligned with the execution of the graphic effects of Vivid OPM.

Evaluation of Vivid OPM

The objective of developing and implementing Vivid OPM was to provide modelers with a tool that can enhance the understandability of the OPM conceptual model, which is often problematic for most people who are not experienced with conceptual modeling in general and OPM-based conceptual modeling in particular.

To assess the value of the animation that Vivid OPM plays from a given OPM model in terms of enhancing model comprehension, we have designed and conducted a controlled experiment, whose goal was to assess the effect of adding Vivid OPM to an OPM conceptual model in terms of its contribution to better understanding of the OPM model.

This section is divided into five subsections. The Section "Experiment population and background" describes the research population. In the Section "Experimental design: The Vivid OPM evaluation system," we describe the experimental design, which includes description of the application that we built particularly for the experiment, as well as description of the two OPM models used in the experiment. In the Section "Experiment hypothesis and method," we state the research hypothesis and methods. The Section "Data analysis and results" contains the results and their interpretation. Finally, the Section "Conclusions and discussion" states the conclusions of the experiment.

Experiment population and background

The experiment was carried out in the fall semester of 2011–2012 at Technion, Israel Institute of Technology. A total of $n = 154$ students from the faculty of Industrial Engineering and Management took part in the research, which was conducted in three different courses. We refer to these groups, respectively, as Course A ($n_1 = 13$), Course B ($n_2 = 132$), and Course C ($n_3 = 9$).

The first group consisted of $n_1 = 13$ students who took course A, titled "mini-project in industrial engineering." These were freshmen with almost no prior knowledge of OPM. Throughout the course they constructed two OPM models of a real-life project (Erkoyuncu et al. 2013). The second, major group, included $n_2 = 132$ students in course B, "analysis and specification of information systems." Their background included basic knowledge of OPM, UML, and SysML. The last group $n_3 = 9$ included third- and fourth-year undergraduate students, as well as several graduate (MSc) students in the Information Management Engineering program, who took course C, "methodologies in information systems development." These students had relatively extensive knowledge of conceptual modeling in OPM, which they had applied during the course in real-life projects, such as the EU FP7 TALOS—border control system project. Each of the three groups received a 45-minute lecture during the semester about Vivid OPM, describing its use, architecture, and capabilities.

The experiment was carried out using the Vivid OPM evaluation system. As described in the Section "Experimental design: The Vivid OPM evaluation system," this is a web-based system, which was especially designed and developed for the sake of this experiment. Upon login to the Vivid OPM evaluation system, each student was automatically and arbitrarily referred to the one out of two possible groups: the experimental group ($n = 76$), who received the OPM model along with the Vivid OPM animation and the control group ($n = 78$), who received the OPM model without the Vivid OPM animation. Table 12.4 presents

Table 12.4 Distribution of Students between the Test Group and the Control Group in Each Course

Course	Course name	Experimental group (with vivid OPM)	Control group (without vivid OPM)
A	Mini-project in industrial engineering	6	7
B	Analysis and specification of information systems	63	69
C	Methodologies in information systems development	7	2

the distribution of students between the experimental group and the control group in each course.

Experimental design: The Vivid OPM evaluation system

This section describes the Vivid OPM evaluation system (Bolshchikov 2011) that was designed and built especially for the purpose of conducting the experiment. This application was built for several purposes:

- Managing the experiment that was performed concurrently by subgroups of 21 students
- Dividing participating students equally into the experimental and control groups
- Storing the questionnaire responses
- Providing basic analytics regarding the amount of correct responses

Figure 12.3 depicts the Vivid OPM evaluation system architecture.

As Figure 12.3 shows, the Vivid OPM evaluation system consists of four modules: Registering, Models, Questionnaire, and Analysis. As soon as the application is launched, the Registration module opens. Here, the user fills in his/her ID and name. The Registering module then assigns each student to a group, based on prior information about how many people need to be in each group.

The experiment was conducted on two OPM models from two disparate domains: one from the biological domain—mRNA life cycle and the other from the banking domain—an automated telling machine (ATM). The rationale behind this choice was to test a sample of the diverse domains in which OPM can be applied and to cancel out possible variability in the levels of students' familiarity with each domain. Specifically, an OPM model and its Vivid OPM animation of an ATM might not provide

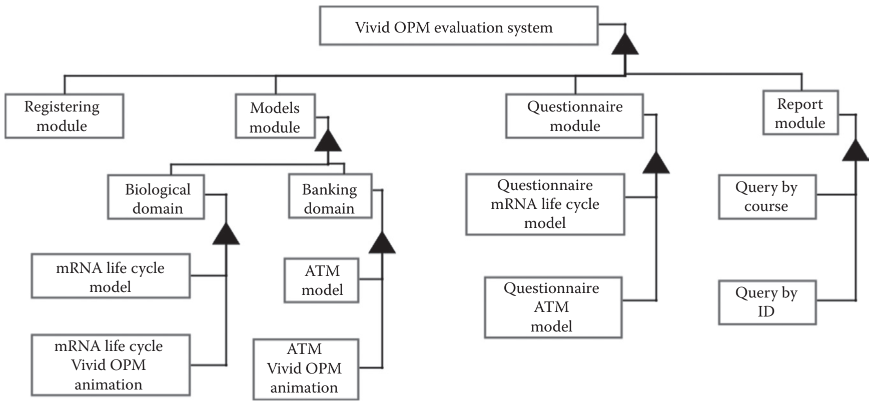


Figure 12.3 Architecture of the Vivid OPM evaluation system.

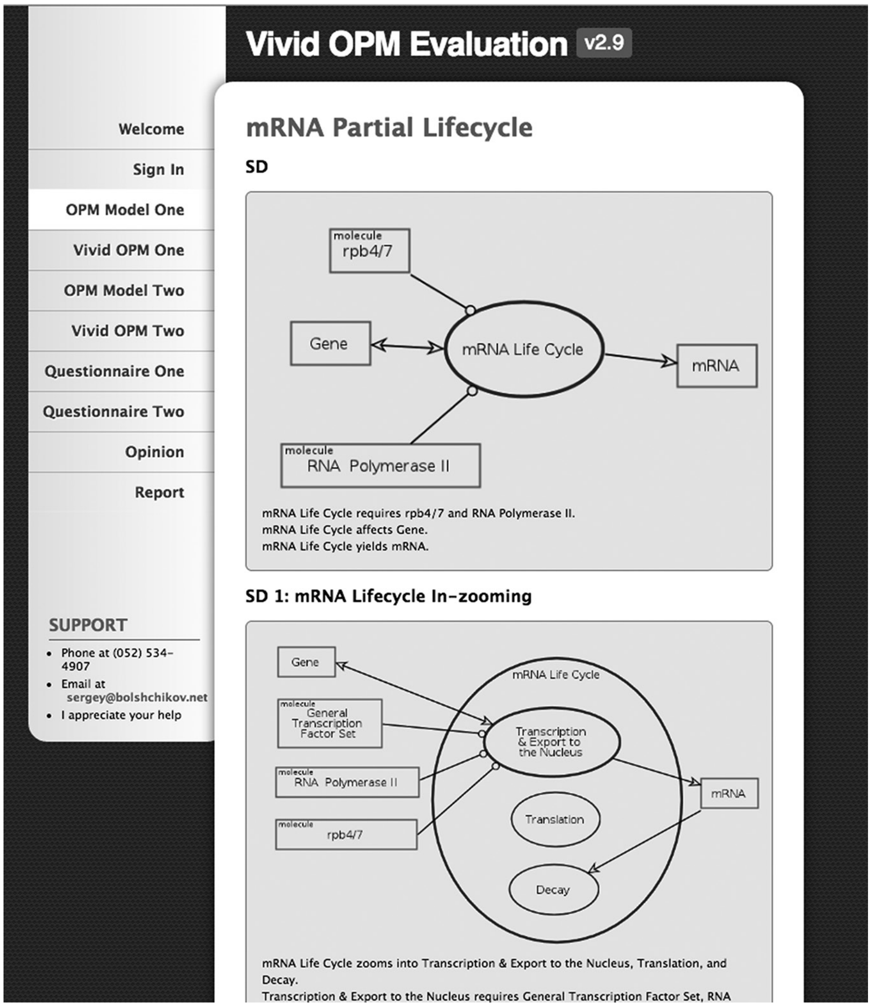
reliable results as most of the students are familiar with its operation from their daily lives, but they are much less familiar with a molecular biology system.

Figure 12.4 shows the system diagram (SD; top level) and the next level down (SD1) of the mRNA life cycle system as presented to the experimental students within the Vivid OPM evaluation system. Our Vivid OPM evaluation system automatically assigned the mRNA life cycle system as "OPM Model One" and "Vivid OPM One" to half of the experimental students and the ATM model system as "OPM Model Two" and "Vivid OPM Two" to the rest of the experimental students. A similar assignment was made for the control group students. This took care of cancelling out any potential learning that might occur during the experiment and cause bias in the outcomes.

Figure 12.5 shows the menu that a control group student sees. Comparing this to the menu in Figure 12.4, we see that a control group student cannot access Vivid OPM Model One and Vivid OPM Model Two.

As Figure 12.4 shows, each OPM model was presented using its two modalities, namely, each OPD was accompanied beneath it with its corresponding, automatically generated text of OPL paragraph, composed of OPL sentences. The presentation of the textual modality next to each graphical modality model piece facilitates the understanding of the OPM model even before adding the potential enhancement of the Vivid OPM part, increasing the challenge of proving that the addition of the Vivid OPM component is of value in terms of enhancing model understandability.

Each of the models was provided by a prerecorded Vivid OPM animation clip of the two systems used in the experiment. These are represented by the two objects at the bottom of Figure 12.3: mRNA life cycle Vivid OPM animation and ATM Vivid OPM animation. The two Vivid OPM



mRNA Life Cycle requires rpb4/7 and RNA Polymerase II.
mRNA Life Cycle affects Gene.
mRNA Life Cycle yields mRNA.

mRNA Life Cycle zooms into Transcription & Export to the Nucleus, Translation, and Decay.
Transcription & Export to the Nucleus requires General Transcription Factor Set, RNA

Figure 12.4 mRNA life cycle system as presented to the experimental and control students within the Vivid OPM evaluation system.

animation clips were uploaded to the website of the Vivid OPM evaluation system, such that each experimental group student could watch it as many times as she or he deemed necessary. Figure 12.6 shows a snapshot of the preprepared clip of the ATM system as presented to the experimental students within the Vivid OPM evaluation system.

The questionnaire module contained two blocks: the ATM model questionnaire and the mRNA life cycle model questionnaire, each composed of nine questions. Each question can have more than one correct answer.

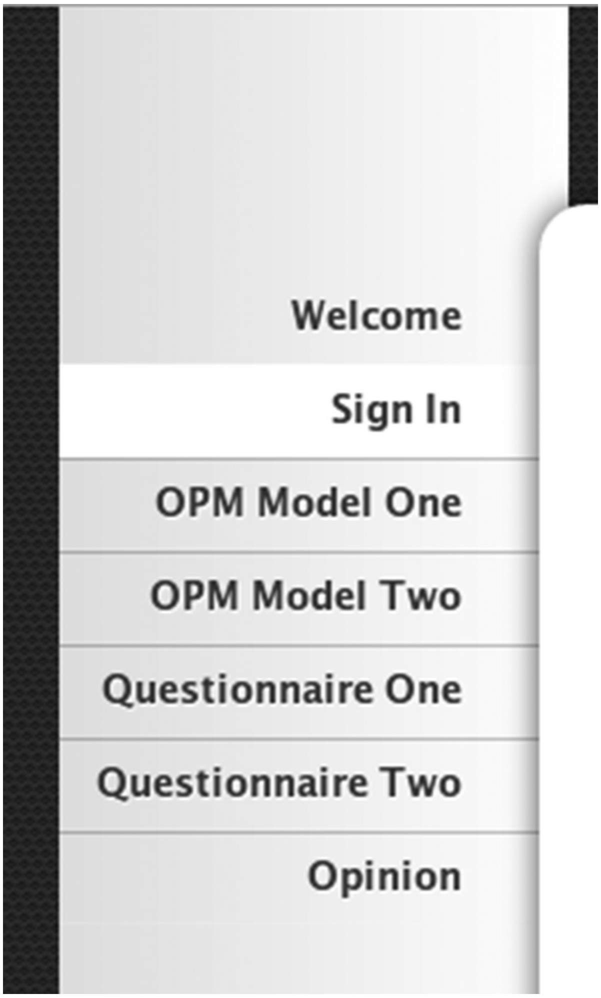
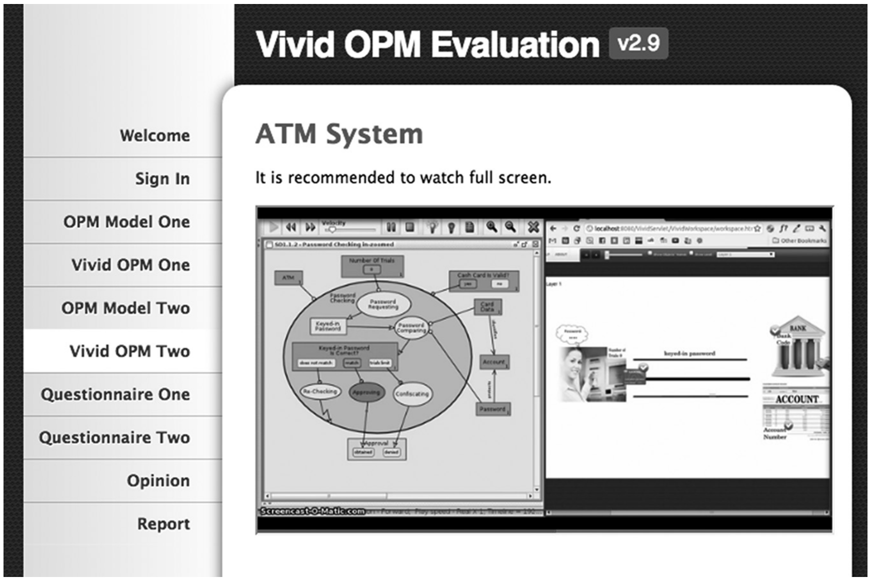


Figure 12.5 Menu that a control group student sees.

Figure 12.7 shows a screenshot of the top part of the ATM system questionnaire, which, in this case, happens to be "Questionnaire Two." As an example, the first question from the ATM system questionnaire is: "Who initially has Cash?" with the following response options:

- ATM
- Bank
- Account
- Customer

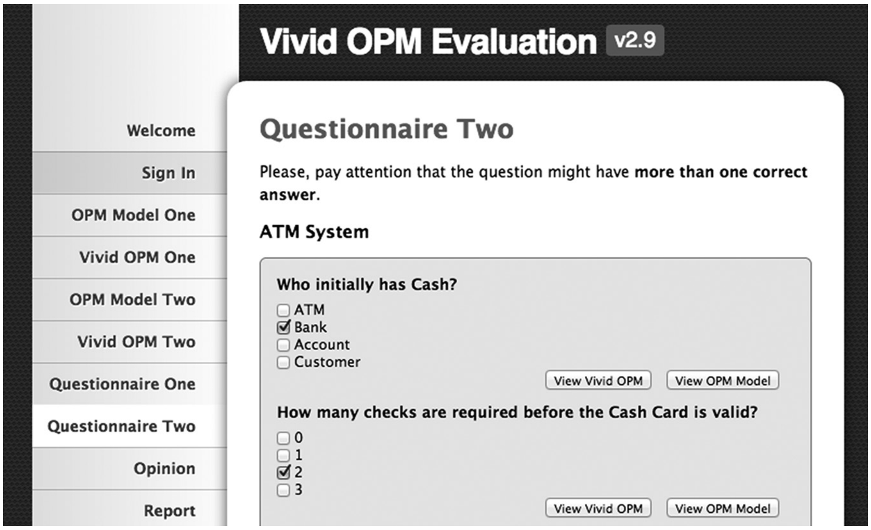


Vivid OPM Evaluation v2.9

ATM System

It is recommended to watch full screen.

Figure 12.6 A snapshot of the preprepared movie of the ATM system as presented to the experimental students within the Vivid OPM evaluation system.



Vivid OPM Evaluation v2.9

Questionnaire Two

Please, pay attention that the question might have **more than one correct answer.**

ATM System

Who initially has Cash?
- ☐ ATM
- ☑ Bank
- ☐ Account
- ☐ Customer

How many checks are required before the Cash Card is valid?
- ☐ 0
- ☐ 1
- ☑ 2
- ☐ 3

Figure 12.7 A screenshot of the top part of the ATM system questionnaire.

Examining Figure 12.8, one can see that the object Cash, depicted at the bottom of the OPD, has two states: bank and customer, with bank being the initial state. This is denoted by the bold state frame and is also indicated by the OPL sentence "State bank is initial." Hence, the correct answer is Bank, as it is the initial state of Cash.

Each student was supposed to answer a total of 18 questions, 9 for each model. As Figure 12.7 shows, while answering the questions, students had the option to browse the OPM model diagrams without limitation by clicking the "View OPM Model" button, and—for the experimental group students only—also to watch without limitation the Vivid OPM animations, by clicking the "View Vivid OPM" button.

The last module is the Analysis module. It is invisible to the experiment subjects, as it is intended for initial analysis of the subjects' responses, such as calculating the amount of correct and incorrect answers of students in each course. This data is presented in Table 12.5 for further analysis.

Experiment hypothesis and method

Our research hypothesis was that there would be significant differences in students' achievements, as reflected in their responses, between the experimental group, who used Vivid OPM, and the control groups, who did not use Vivid OPM. The differences would be such that the Vivid OPM groups would achieve significantly higher grades than the control groups.

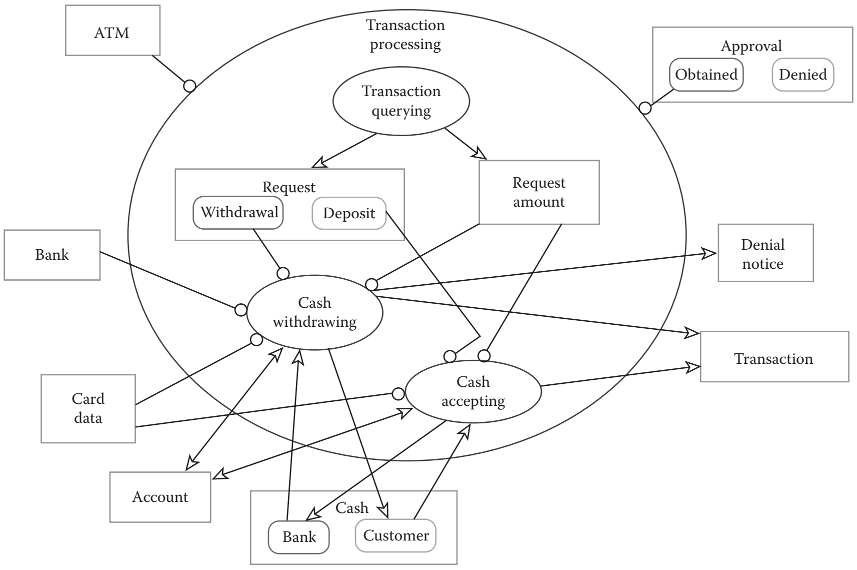


Figure 12.8 Transaction querying of the ATM system in-zoomed.

Table 12.5 Students' Means and Standard Deviations of Correct Answers for the Four Question Categories

Question category	M	SD
Structure	3.13	1.2
Behavior	1.75	0.82
Change of state	3.98	1.09
Interaction	2.48	1.18
Total	11.34	2.92

For data collection, we used a qualitative method. We conducted an achievement test, in which we examined the students' outcomes in the course. The achievement test contained 18 closed questions that tested their ability to read and understand complex system models. This test was validated by three systems engineering experts.

Each one of the 18 questions belonged to one of the following four categories, listed in increasing order of difficulty level:

1. Structure: This category relates to structural aspects of the system. It included five questions regarding issues such as the state of an object and static relations between objects. An example of a question from this category is: "What is the relation between molecules General Transcription Factor Set and RNA Polymerase II?"
2. Behavior: This category relates to behavioral aspects of the system. It included four questions regarding processes and their impact on objects. An example of a question from this category is: "What process breaks RNA Polymerase II apart from rpb4/7 and Gene?"
3. State change: This category relates to aspects pertaining to state changes. It included five questions about the change of object states. An example of a question from this category is: "Which molecule returns to its initial condition?"
4. Interaction: This category relates to relative complex mutual actions and reactions between objects. It included four questions concerning interactions between objects through processes. An example of a question from this category is "Which molecules move with mRNA?"

Data analysis and results

Exploratory factor analysis with varimax rotation was performed for the 18 test questions. The analysis indicated four distinct content worlds that match the four test question categories: structure, behavior, change of state, and interaction. The total explained variance was 41.67%. We summarized the number of correct answers in the test as a whole and the correct answers in each aspect for each student. Means and standard deviations are presented in Table 12.5.

Our research hypothesis was that there would be significant differences in course grades between students who were given the system and questions relating to it with the OPM model and the additional Vivid OPM and those who were given the system and questions relating to it with the OPM model only and without the Vivid OPM component. We expected the grades of the experimental students, who used Vivid OPM, to be significantly higher.

To test this hypothesis, we first used one-way analysis of variance (ANOVA) to test if there were differences in the number of correct answers between the three courses. The analysis of variance showed that the effect of course was not significant, $F(2151) = 0.652$, $p > .05$, indicating that there were no differences between the three courses. This enabled us to coalesce the experimental and control groups in each course into one large experimental group (with Vivid OPM) and one large control group (without Vivid OPM).

Further, the four question categories, namely, structure, behavior, change of state, and interaction, were subject to a one-way multivariate analysis of variance (MANOVA) with the two research groups, experimental (with Vivid OPM) and control (without Vivid OPM). The students' mean of correct answers in each aspect and their standard deviations by research group are presented in Table 12.6.

The MANOVA result was significant for research group by aspect, $F(4149) = 3.57$, $p < .01$, $\eta_p^2 = 0.09$, indicating that there were significant differences between the experimental and control groups in number of correct answers.

To detect the source of the differences, we conducted a separate ANOVA for each of the four question categories. The results indicated significant differences only in the interaction question category: $F(1152) = 14.11$, $p < .001$, $\eta_p^2 = 0.09$. No significant differences were detected in the

Table 12.6 Students' Means, Standard Deviations, and *F* Results of Correct Answers for Differences between Experimental and Control Groups for the Four Question Categories

Question category	Experimental group (with Vivid OPM)		Control group (without Vivid OPM)		
	M	SD	M	SD	F(1152)
Structure	3.20	1.24	3.06	1.17	0.471
Behavior	1.75	0.85	1.74	0.80	0.002
State change	4.08	1.00	3.88	1.17	1.22
Interaction	2.83	1.09	2.14	1.24	14.11**

**$p \leq 0.001$.

structure question category, behavior question category, and change of state question category.

Conclusions and discussion

Our experimental results indicate that, in line with our research hypothesis, Vivid OPM enhances model understandability. Refining this conclusion by looking into each one of the four question categories reveals that Vivid OPM did not enhance model understandability in three of the four aspects: structure, behavior, and change of state. This is likely so due to the simplicity and user-friendly presentation of OPM even without the additional animation that Vivid OPM provides. Only when really complex questions are posed, such as those belonging to the interaction category, Vivid OPM, with its intuitive interface and animated actions, does provide a significant added value. It is this complex side of the question spectrum where the real benefit of Vivid OPM is exposed: When the question to be answered involves understanding a complex situation, in which several objects interact through processes, Vivid OPM contributes to providing a correct answer, indicating its benefit as a means of disambiguating complicated situations and aiding in making the model more comprehensible. Although OPM with its graphic and textual modalities is sufficient for correctly answering questions of the first three categories—structure, behavior, and state change—when it comes to the really difficult questions, those of the interaction category, Vivid OPM is of real and significant help in providing a correct answer.

Summary and future work

In this chapter, we have developed, presented, tested, and evaluated a visualization, in addition to OPM, called Vivid OPM. Vivid OPM enables animation of a conceptual OPM model, making it closer to reality than the original OPM conceptual model. The OPM model presents both the structure and the behavior of the system under study using a static set of diagrams. Although the diagrams can be animated, the animation is of the geometric shapes representing objects and processes in the diagrams. In this type of diagram animation, symbols of objects (denoted as boxes or rectangles), states (rounded-corner rectangles within the object boxes), and processes (ellipses) change colors as objects become existent and change states, and as processes are being executed. At the same time, red dots run along procedural links connecting objects or states with processes, symbolizing the passage of time.

In contrast, the type of animation that Vivid OPM provides is more concrete. It involves the dynamics of the object icons themselves, not of their symbols. This is a principal change: We animate icons that look like the objects themselves, not their conceptual symbols with the names recorded inside them. This is a big step toward making the abstract conceptual model concrete. Furthermore, the object icons perform movements, rotations, sizing, and color changes, as dictated by the OPM model and the Vivid OPM extension, making the dynamic aspect of the model closed to *the real thing* so it is more concrete and more comprehensible.

We tested OPM and Vivid OPM against OPM only on a research population of 154 information systems engineering students using a specially developed system for this task, which is also described in detail in the Section "Experimental design: The Vivid OPM evaluation system."

We found that there is a significant advantage in the addition of the Vivid OPM component to the OPM model when it comes to being able to respond to questions related to interactions, which are the most complex category of the four question categories we defined. The fact that Vivid OPM is most helpful in answering the most complex questions is a testimony of its value. When the questions are simple enough, there is no significant added value to Vivid OPM, because OPM itself is sufficiently intuitive and self-explanatory. However, as the question complexity increases, the real discriminatory added value of Vivid OPM becomes evident.

Future work in the direction of model-driven animation includes extending the two-dimensional Vivid OPM to three dimension using computer-based augmented virtual environment (CAVE) technology. A vision for this system is presented in the work by Dori (2012). In this vision, an intelligent multimedia device dubbed computer-based augmented virtual environment for realizing nature (CAVERN) is proposed as a quantum leap in molecular biology research. CAVERN is a system that leverages state-of-the-art technologies that include CAVE, supercomputing, electron microscopy, conceptual modeling, and biological text mining. A new notion of the human spatiotemporal comfort zone is presented along with a fourth multimedia learning assumption—the limited spatiotemporal comfort zone. Within this zone, people can use their senses to follow and understand complex systems that are currently accessible only through indirect observations. CAVERN translates nanolevel processes modeled in OPM into scenarios of human-size interacting molecules. A conceptual blueprint of CAVERN expressed via an OPM model.

References

Bolshchikov, S. 2011. Vivid OPM Evaluation. Retrieved from http://vividopmevaluation.appspot.com.

Bolshchikov, S., Renick, A., Mazor, S., Somekh, J., and Dori, D. 2011. OPM model-driven animated simulation with computation interface to Matlab. *20th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprise* (WETICE), pp. 193–198. Paris, France.

Dori, D. 2002. *Object-Process Methodology—A Holistic Systems Paradigm*. Berlin, Heidelberg, Germany: Springer Verlag.

Dori, D. 2012. Extending the human spatiotemporal comfort zone with CAVERN—Computer-based augmented virtual environment for realizing nature. *Journal of Multidisciplinary Research*, 4(3), 23–44.

Dori, D., Linchevski, C., and Manor, R. 2010. OPCAT—A software environment for object-process methodology based conceptual modelling of complex systems. *1st International Conference on Modelling and Management of Engineering Processes*, pp. 147–151. Cambridge, United Kingdom.

Dori, D., Reinhartz-Berger, I., and Sturm, A. 2003. Developing Complex Systems with Object-Process Methodology using OPCAT. *Lecture Notes in Computer Science*, 2813, 570–572.

Erkoyuncu, J., Bolshchikov, S., Steenstra, D., Rajkumar, R., and Dori, D. 2013. Application of OPM for the healthcare sector. *Conference on Systems Engineering Research* (CSER'13). Atlanta, GA.

Fowler, M. 1997. *Analysis Patterns: Reusable Object Models*. Boston, MA: Addison-Wesley Professional.

Fowler, M. 2004. *UML Distilled: A Brief Guide to the Standard Object Modeling Language*. Boston, MA: Addison-Wesley Professional.

Harel, D. 1987. Statecharts: A visual formalism for complex systems. *Science of Computer Programming*, 8(3), 231–274.

IBM Corporation. 2009. Systems Engineering Tutorial for Rational Rhapsody.

Marton, F. 1981. Phenomenography—Describing conceptions of the world around us. *Instructional Science*, 10, 177–200.

Mellor, S. and Balcer, M. 2002. *Executable UML: A Foundation for Model-Driven Architecture*. Boston, MA: Addison-Wesley.

No Magic, Inc. 2013a. Cameo Simulation Toolkit. Retrieved from http://www.nomagic.com/products/magicdraw-addons/cameo-simulation-toolkit.html.

No Magic, Inc. 2013b. MagicDraw. Retrieved from http://www.nomagic.com/products/magicdraw.html.

Object Management Group. 2014. UML® Resource Page. Retrieved from http://uml.org/

Oracle. 2013. The Java EE 6 Tutorial: Java Servlet Technology. Retrieved from http://docs.oracle.com/javaee/6/tutorial/doc/bnafe.html.

Peleg, M. and Dori, D. 2000. The Model Multiplicity Problem: Experimenting with Real-Time Specification Methods. *IEEE Transaction on Software Engineering*, 26(8), 742–759.

Perelman, V., Somekh, J., and Dori, D. 2011. Model Verification Framework with Application to Molecular Biology. *Symposium on Theory of Modeling and Simulation (DEVS 2011)*, Boston, MA, April 4–9, 2011.

Somekh, J., Choder, M., and Dori, D. 2012. Conceptual Model-Based Systems Biology: Mapping Knowledge and Discovering Gaps in the mRNA Transcription Cycle. *PLoS ONE*, 7(12): e51430. doi:10.1371/journal.pone.0051430, December 20, 2012.

Yaroker, Y., Perelman, V., and Dori, D. 2013. An OPM Conceptual Model-Based Executable Simulation Environment: Implementation and Evaluation. *Systems Engineering*, 16(4), 381–390.