Research Article

Cognition-Based Visualization of the Dynamics of Conceptual Models: The Vivid OPM Scene Player

Sergey Bolshchikov,[1] Judith Somekh,[2] Shay Mazor,[1] Niva Wengrowicz,[3] Mordechai Choder,[4] and Dov Dori[5],*

[1] *Faculty of Industrial Engineering and Management, Technion, Israel institute of Technology, Haifa, Israel*
[2] *Faculty of Industrial Engineering and Management, Technion, Israel institute of Technology, Haifa, Israel, and Division of Medical Sciences, Harvard Medical School, Harvard University, Boston, MA, USA*
[3] *Faculty of Industrial Engineering and Management, Technion, Israel institute of Technology, Haifa, Israel*
[4] *Faculty of Medicine, Technion, Israel, institute of Technology, Haifa, 32000, Israel*
[5] *Faculty of Industrial Engineering and Management, Technion, Israel institute of Technology, Haifa, Israeland Engineering Systems Division, Massachusetts Institute of Technology, Cambridge, MA, USA*

Received 24 December 2013; Revised 29 March 2015; Accepted 16 August 2015, after one or more revisions
Published online in Wiley Online Library (wileyonlinelibrary.com).
DOI 10.1002/sys.21321

ABSTRACT

Modeling plays an increasingly important role in the lifecycle of systems. Complex dynamic systems are difficult to model, preventing users from deeply understanding their intricate behavior. Existing conceptual modeling languages contain behavioral diagrams aimed to describe how the modeled system changes over time. However, most of these diagram types are static and do not directly reflect the system's behavior in space and time in a manner that is close to conceived reality. Models that are inherently visual and dynamic can potentially provide system architects and designers, as well as prospective customers, with profound understanding of the behavior of the system under development without requiring knowledge of any specific modeling language. Based on this conjecture, which is supported by cognitive neuroscience, we present Vivid Object-Process Methodology (OPM), a software module that generates and plays a "video clip" of the system under development from its OPM conceptual model. While requiring relatively little effort on the side of the modeler, this option explicates how the system behaves over time, providing a powerful tool for understanding and communicating complex systems dynamics. Testing Vivid OPM with human subjects, we found that it enhances the understandability of the system under study, especially in complex situations, where interaction is involved. The preliminary animation application we present can evolve into a powerful model-based 3-dimensional tool for visualizing systems of increasing complexity and sophistication, serving scientists,systems engineers, and students at all levels. © 2015 Wiley Periodicals, Inc. Syst Eng 00: 1–9, 2015

Key words: Object-Process Methodology (OPM); simulation; visualization; dynamic models; spatio-temporal models; cognition

1. INTRODUCTION

Conceptual modeling is increasingly recognized as a vital stage in the process of developing any complex system. It allows expressing the meaning of terms and concepts used

*Author to whom all correspondence should be addressed (e-mail: dori@ie.technion.ac.il).

by domain experts to discuss the problem and to find correct relationships between different concepts [Fowler, 1997]. 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. This poses a challenge for people whose abstract thinking is not sufficiently developed. 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, while an abstract thinker can think about the class of those objects.

Object-Process Methodology, OPM [Dori, 2002] is a conceptual systems modeling paradigm that caters to development, evolution, and lifecycle support of complex dynamic systems. OPM builds on a minimal set of concepts: Stateful objects—things that exist, and processes—things that happen and transform objects. In OPM a system model is described in terms of things that exist—objects, and things that happen to the objects—processes. Objects are what a system or product is, and they may be stateful, that is, have states. Processes express what a system does—how it transforms the objects, where transformation means generation of new objects, consumption of existing objects, or change of their state. OPM semantics is generic and domain independent; it is geared toward systems engineering, as it can model information, hardware, people, and regulation. The OPM model shows the structural and procedural links between the system's building blocks at any needed level of detail using a compact alphabet of concepts and symbols to represent them.

OPM is bimodal: It is developed and displayed graphically using a single type of diagram, while a textual interpretation of the graphic input is created automatically in response to the user input. The major system aspects—function, behavior, and structure—are integrated in OPM's single diagram type, while complexity is managed via intuitive yet formal refinement and abstraction mechanisms that create a tree of interrelated diagrams.

OPM explicitly addresses the system's dynamic-procedural aspect, which describes how the system changes over time. However, the resulting model is basically static, and as such, it does not fully reflect the behavior of the system being architected or designed.

OPCAT [Dori et al., 2010; OPCAT, 2013][1] is a freely available software environment that supports OPM-based conceptual modeling. The single model provides for clear and expressive animated simulation of the OPM model using OPCAT, which greatly facilitates design-level debugging [Yaroker et al., 2013]. The simulation module provides for animating and verifying the OPM model, but the animation is at the abstract, conceptual level. It shows how the model entities—process ellipses, object boxes, and state rountangles—change colors as processes transform objects while red dots slide along the links. The underlying model is still conceptual; it represents concepts as geometric figures with animation. 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 is, the more intuitive and deeper humans' understanding of the system is.

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 very intuitively. Thus, creation of a visual dynamic model from a static one, which represents changes of objects in space over time by mimicking the actual system's behavior, is likely to enable better comprehension of the system's behavior without requiring knowledge of any specific modeling language. This conjecture is supported by pertinent cognitive science literature surveyed briefly below.

To realize this visual dynamic model rendering, we have developed Vivid OPM—a software module that takes an OPM conceptual model and plays a vivid clip of the dynamics of the system under development or research. Prior to pursuing this path of development, we had considered several options for solving the problems of lack of truly dynamic models. These are briefly described below following a brief background of relevant cognitive processes.

2. MOTIVATION

The motivation is presented as a nonexhaustive literature review that covers first cognitive aspects of concrete versus abstract understanding. We then turn to a review of main conceptual executable modeling languages and frameworks with their advantages and disadvantages.

2.1. Cognitive Aspects of Concrete versus Abstract Understanding

Sein and Bostrom [1989] investigated individual differences and conceptual models in training novice users to evaluate the claim that conceptual models are effective in aiding users to build mental models of computer systems. They examined the influence of visual ability and learning mode on the mental model formation process of an electronic mail filing system among novice users. They found that high-visual subjects performed significantly better than low-visual subjects, and that abstract learners also performed better than concrete learners. Interaction effects showed that low-visual subjects were severely hampered by abstract models, but performed at the same level as high-visual subjects when provided with analogical models. Abstract learners benefited from the abstract model but were hampered by the analogical model. On the other hand, concrete learners performed better with the analogical models compared to abstract models. They concluded that there is a need to consider individual differences, such as visual ability and learning mode, in research on mental models and on human-computer interaction.

Binder et al. [2005] investigated the effects of word imageability and concreteness in order to understand how the brain processes conceptual knowledge. Using functional magnetic

[1]OPCAT is downloadable free from http://esml.iem.technion.ac.il/

resonance imaging (FMRI) while participants identified concrete and abstract words, they found that relative to nonwords, concrete and abstract words both activated a left-lateralized network of multimodal association brain areas previously linked with verbal semantic processing. Their results show overlapping but partly distinct neural systems for processing concrete and abstract concepts. Processing of abstract concepts was detected almost exclusively in the left hemisphere.

These and other similar results indicate that (1) there are individual differences among humans in the ability to process concrete versus abstract information, and (2) concrete and abstract words activate at least partially different brain areas. These conclusions provide strong neurocognitive theoretical basis for the development of a system that enables making a conceptual model concrete, catering to humans whose abstraction capabilities are not strong and therefore need concrete interpretation of the conceptual model using images and visualizations.

We next survey current executable modeling approaches and their relevance to the task we seek to implement.

2.2. Executable UML

Executable UML is a UML profile that graphically specifies a system "at the next higher level of abstraction, abstracting away both specific programming languages and decisions about the organization of the software" [Mellor, 2002]. The models are testable, and can be compiled into a less abstract programming language to target a specific implementation. Executable UML supports Model Driven Architecture (MDA) through specification of platform-independent models and the compilation of the platform-independent models into platform-specific models [Mellor, 2002].

A system is composed of multiple subject matters, known in Executable UML terms as domains. Executable UML is used to model a domain at the level of abstraction of its subject matter, independent of implementation concerns. The resulting domain model is represented by domain charts, class diagrams, state charts, and action language.

Advantages of Executable UML are:

- higher level of abstraction than 3GLs,
- provision for true separation of concerns,
- support of nondeterministic behavior, and
- connection between documentation and programming language, as the models are a graphical, executable specification of the problem space that is compiled into a target implementation.

Executable UML has the following disadvantages [Object Management Group, 2010]:

- It uses a limited number of constructs,
- it permits a limited amount of customization,
- it provides only visual notation for activity modeling, and
- Its activity diagrams are hard to comprehend.

2.3. 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 one to [IBM Corporation, 2009]:

- Perform system analysis to define and validate system requirements,
- Design and specify the system architecture,
- Perform systems analysis and design,
- Perform software analysis and design,
- Carry out software implementation, and
- Validate and simulate 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. It requires Statecharts, activity diagrams and textual behavior specifications, which would capture the behavior of the model, to simulate a model. Structural definitions like blocks, ports, parts, and links are used to create a simulation hierarchy of subsystems. The preparation for the simulation is a relatively complicated process, which requires the following five steps: Creating a component, configuring the component, generating code for the component, building the component application, and simulating the component application.

2.4. MagicDraw

MagicDraw is the business process, architecture, software and system modeling tool with teamwork support, based on UML 2. It provides analysis and design of Object Oriented (OO) systems and databases for business analysts, software analysts, programmers, quality assurance engineers, and documentation writers. Cameo Simulation Toolkit extends Magic Draw as a separate add-on with simulation and animation capabilities for validating system behavior by executing, animating, and debugging of UML 2 state machine and activity diagrams in the context of realistic mock-ups of the intended user interface [No Magic, Inc., 2013].

These two industrial solutions, Rhapsody and Cameo, have similar disadvantages:

- Built to support UML, they allow simulation and animation of mostly software and hardware systems, but they are not adequate for conceptual models of complex socio-technical systems in general.
- Both of them require several types of UML diagrams (activity diagrams and Statecharts as a minimum) in order to operate, while OPM allows to build simulation with its only type of diagram, the Object-Process Diagram (OPD).
- Both solutions require a complex sequence of step in order to build the animation.

2.5. System Dynamics

System Dynamics (SD) [Forrester, 1971] is an approach to understanding the behavior of complex systems over time through internal feedback loops and time delays which affect the behavior of the entire system. The use of feedback loops with stocks and flows differentiates system dynamics from other approaches to studying complex systems. These elements help describe how even seemingly simple systems display baffling nonlinearity. SD has certain limitations [Lane, 2000]:

- It does not always explain how flows influence stocks,
- Loops can be mislabeled, and
- It cannot provide a diagrammatic explanation of all dynamic phenomena.

2.6. Live Sequence Charts

Harel and Marelly [2003] have developed a methodology for scenario-based specification of reactive systems, in which the behavior is “played in” directly from the system’s GUI or some abstract version thereof, and can then be “played out.” The approach is supported and illustrated by the play-engine tool. As the behavior is played in, the play-engine automatically generates a formal version in the language of live sequence charts (LSC). Play-in focuses on requirements elicitation, enabling nonprofessional end-users to participate in the process. As the authors note, for more complex and sophisticated features, the user is expected to be more familiar with the language of LSCs, so playing in is like programming in an intuitive, visual and high-level programming environment. Play-out allows even more end users to make the GUI respond and validate the requirements by actually operating the application. While there are similarities between Play-in/Play-out and Vivid OPM, the former is designed for intuitive behavior specification, while the latter focuses on making dynamic complex system models explicit and more understandable.

Other sophisticated discrete event simulations include Vitech Core and ExtendSim, which provide dynamic simulation, as well as continuous simulations such as Simulink and Modelica, which provide extensive analysis capability and visualization. Surveying these is beyond the scope of this paper.

3. VIVID OPM: OBJECTIVE AND ARCHITECTURE

Vivid OPM aims to translate a conceptual OPM model into a spatio-temporal model that is driven by the conceptual model and represents the systems at a level that is closer to conceived reality than the level at which the conceptual model is. Vivid OPM comprises three main parts, presented in Figure 1: (1) OPCAT: the OPM-based systems modeling environment, (2) CLIENT: a visual web-based client side—a GUI platform that visualizes the system’s dynamic aspect driven by the OPCAT-resident OPM conceptual model of the system, and (3) SERVER: a communication server (JMS) which enabled the message exchange between OPCAT and web-based client. Figure 1 depicts the Vivid OPM architecture.

The OPM-based simulator resides in the Client part. It is a Java application that executes the OPM model, enabling comprehension and evaluation of behavioral aspects of the model of the system under development or research. The Vivid OPM Java plug-in, which is also part of the Client, tracks the objects’ state transitions and sends updates to the communication server. The plug-in is capable of halting the OPM simulator execution to ensure that changes in objects’ states do not overrun the graphics transition whenever the objects’ control file reports that a previous state change transition has not yet been completed.

The GUI on the Client-side platform is based on a Web browser. The underlying client-server capabilities support communication between the Web browser and the remote OPM simulator. The Web browser Client’s task is to visualize graphic elements of the model and to enable the user to control the model visualization. The server side holds all the graphic information and controls the OPM-based simulator. The client-server communication is based on AJAX/HTTP requests, the client-server controls are implemented with JAVA Servlets, and the Web client is written in JavaScript. The Web-based client side is generic, making it suitable for visualization of any OPM model. Upon page loading, the client receives an OPM model file uploaded by the user, and extracts from it all the objects and their states. The Vivid OPM GUI provides a user-friendly way to set the objects’ initial spatial arrangement and user-selected or generated icons for the animation.

The client’s JavaScript code periodically executes request to the server for current updates. The animation function is executed whenever an object is in transition between states. The JavaScript code executes the graphical transition effect, which can be movement, sizing, coloring, etc. Whenever an object’s graphical transition is completed, the client-side notifies the server to proceed with the JAVA simulation execution. This synchronization mechanism allows the OPM-based simulator to be aligned with the graphic transition execution.

4. A MOLECULAR BIOLOGY SYSTEM EXAMPLE

Systems biology is an emerging field of intensive research that combines biology the broad domain of systems science and engineering. Biological systems, especially at the molecular level, are most complex, so their models can potentially provide better understanding. A conceptual model of a sophisticated biological system is inherently complex in itself. Biology is therefore one of the attractive applications of Vivid OPM. Indeed, our first users have been biologists who seek to have an accessible tool for eliciting the models they have in mind and which are currently expressed with graphics that do not have an underlying conceptual model driving them. In fact, the original requirement for the ability to animate a conceptual model, which culminated in the development of Vivid OPM, indeed originated from a molecular biology research laboratory at the Technion. A conceptual model is expressed in a graphical language that is unfamiliar to biologists, adding a layer of abstraction and making it difficult for

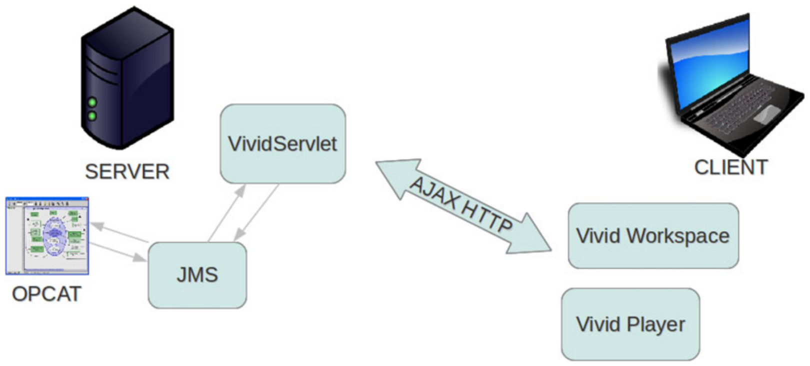

Figure 1. Architecture of Vivid OPM.

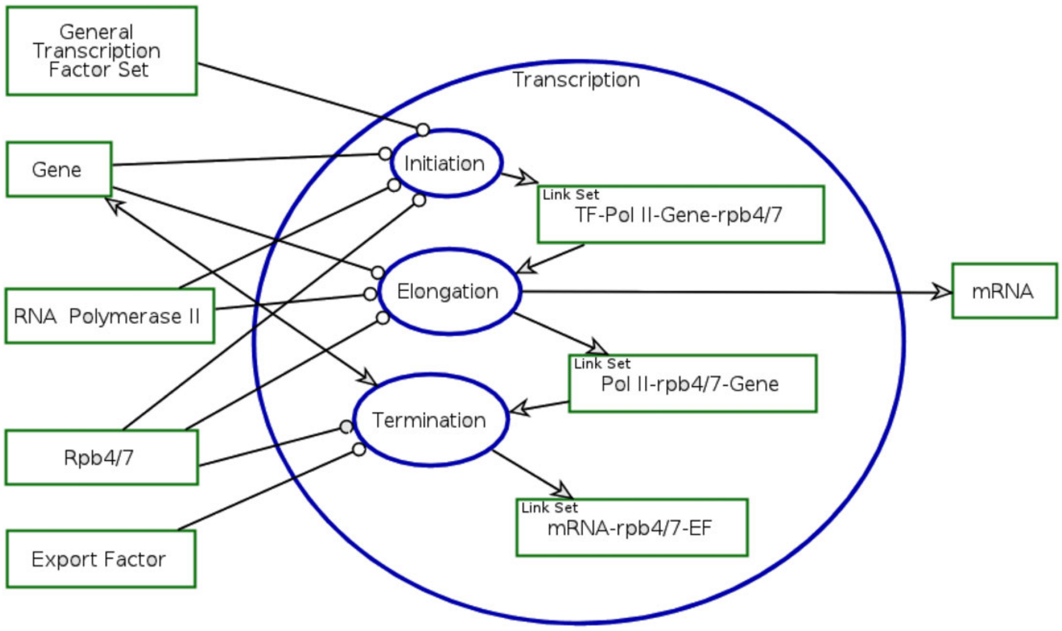

Figure 2. Transcription in-zoomed.

them to gain deep understanding of the system under study. The Vivid OPM example that follows is aimed at presenting the biology researcher with a spatio-temporal model of the molecular biology system which is more graphic, dynamic, and closer to perceived reality than a static conceptual model in an unfamiliar modeling language. The OPM diagrams describe the part of **Transcription** process of **mRNA** [Somekh et al., 2012; Somekh et al., 2014], which consists of three subprocesses shown in Figure 2. The Vivid OPM model is playable [Bolshchikov, 2013]. In the **Initiation** process (Fig. 3) we show only a partial model providing a stepwise order, in which the **Gene** is connected to several molecules: **RNA Polymerase II**, **Rpb4/7** and **General Transcription Factors**. The **Elongation** process (Fig. 4) consumes the **Link Set** between **General Transcription Factors**, **RNA Polymerase II**, **Rpb4/7**, and **Gene**. As the result, the **mRNA** is created. During a subprocess of Transcription that is yet to be defined, the mRNA binds Rpb4/7. The Rpb4/7/RNA complex dissociates from RNA Polymerase II during the polyadenylation subprocess. With the help of Export Factors, the Rpb4/7/mRNA complex is then translocated to the Cytoplasm.

The OPM-based formal conceptual model of creating mRNA has been validated by the built-in OPCAT simulation capability. However, the OPM model does not contain the graphical information required to animate a spatio-temporal model, such as icons for objects, the size of the drawing canvas, coordinates and size of objects, and their translation speeds. Using the Vivid OPM Workspace, the user sets up the configuration of the animation, including object and state spatial arrangements and icons, and background image, as demonstrated in Figure 5.

Figure 6 shows a point in time during the animation execution. On the left-hand side is the Vivid OPM animation, which is driven by the OPM conceptual model, shown on the right-hand side. As the colors that change when the animated simulation of the OPCAT software progresses indicate, in this particular snapshot, the OPM process of

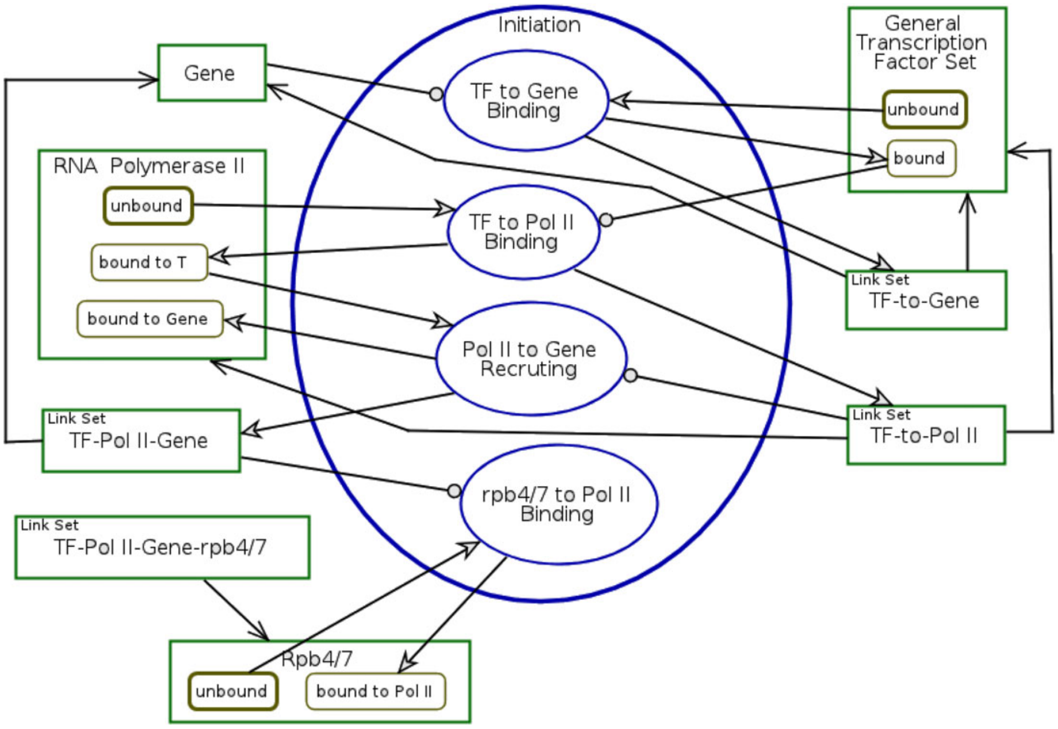

Figure 3. Initiation in-zoomed.

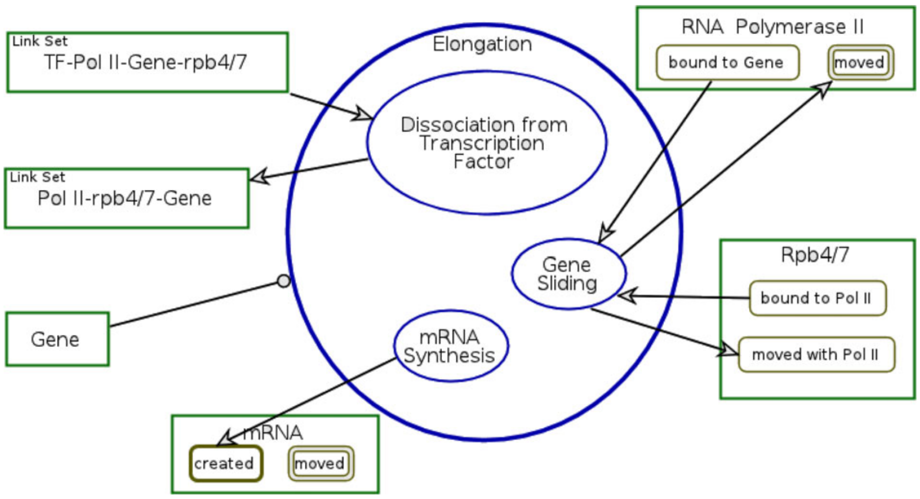

Figure 4. Elongation in-zoomed.

Transporting, the fourth from top inside the larger **Termination** process, is being executed. This is denoted by the fact that the **Transporting** ellipse is highlighted in dark (originally blue color). This process is changing the states of three objects: **mRNA**, **Rpb4/7**, and **Export Factor**. This state change is expressed by the six (red) dots moving along three pairs of input-output links. Thus, the state of **mRNA** changes from **created** to **moved**, the state of **rpb4/7** changes from **bounded to mRNA** to **moved with mRNA**, and the state of **Export Factor** changes from **bounded to mRNA** to **moved**. Figure 6 vividly shows two different aspects of model representation—the conceptual and the spatio-temporal. Such side by side combination is helpful for testing the correspondence between the OPM animation and Vivid OPM and in optional. End users are expected to look mainly at the Vivid OPM clip on the right.

5. EVALUATION OF VIVID OPM

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, we designed and conducted a controlled experiment [Dori et al., 2014]. In this section we

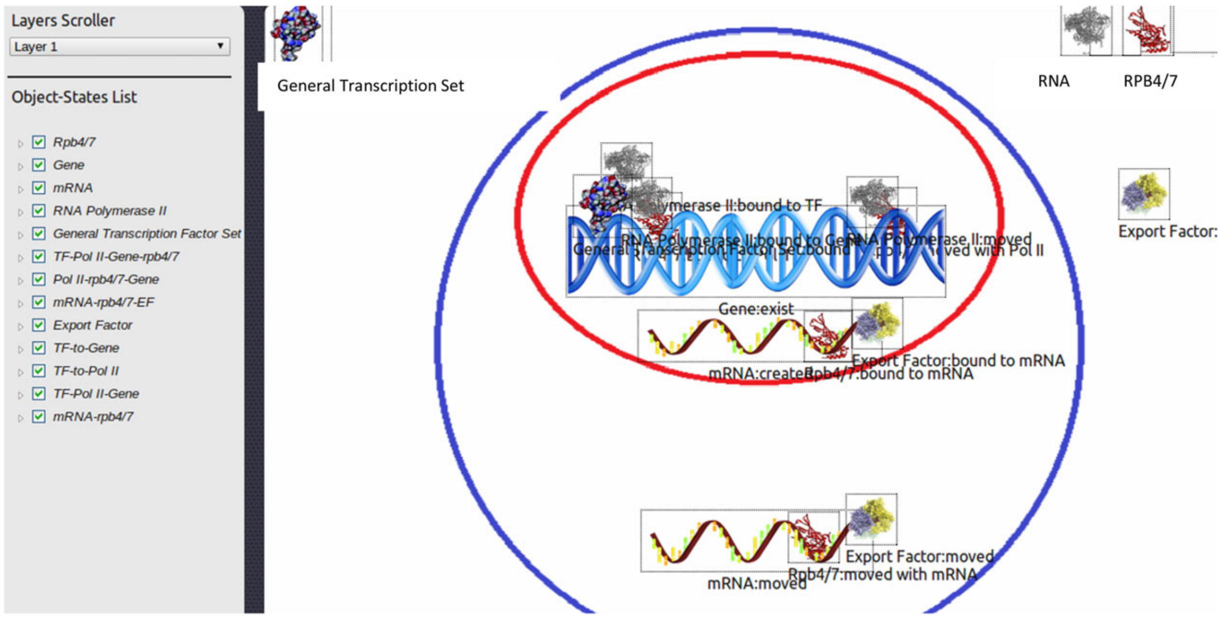

Figure 5. Example of a workspace animation configuration (some text overlaps).

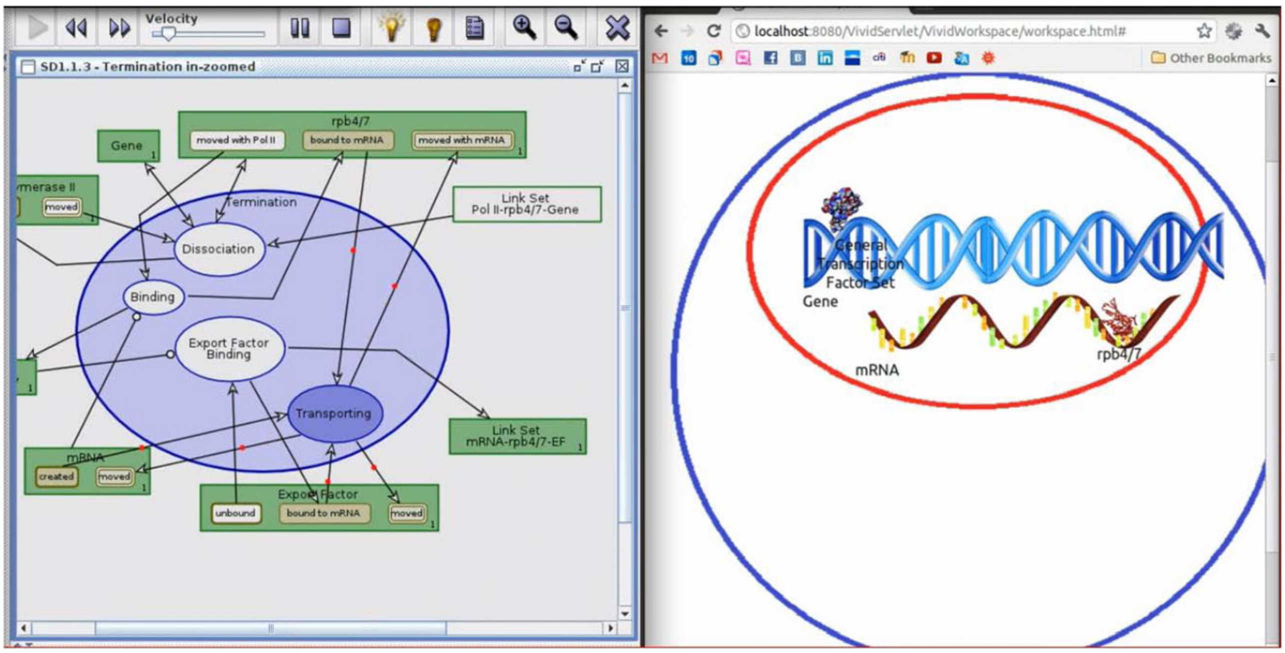

Figure 6. The Vivid OPM animation (right) driven by the OPM conceptual model (left) showing the transporting process.

briefly summarize the experiment and its outcomes. A total of 154 students from the faculty of Industrial Engineering and Management at the Technion, Israel Institute of Technology, took part in the research. We developed a dedicated Vivid OPM Evaluation System for this research. Students were presented with three examples. Each model was provided by a prerecorded Vivid OPM animation movie of the two systems used in the experiment. Figure 7 shows a snapshot of the pre-prepared movie of the ATM system as presented to the experimental students within the Vivid OPM Evaluation System.

Our research hypothesis was that there would be significant differences in students' achievement, 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.

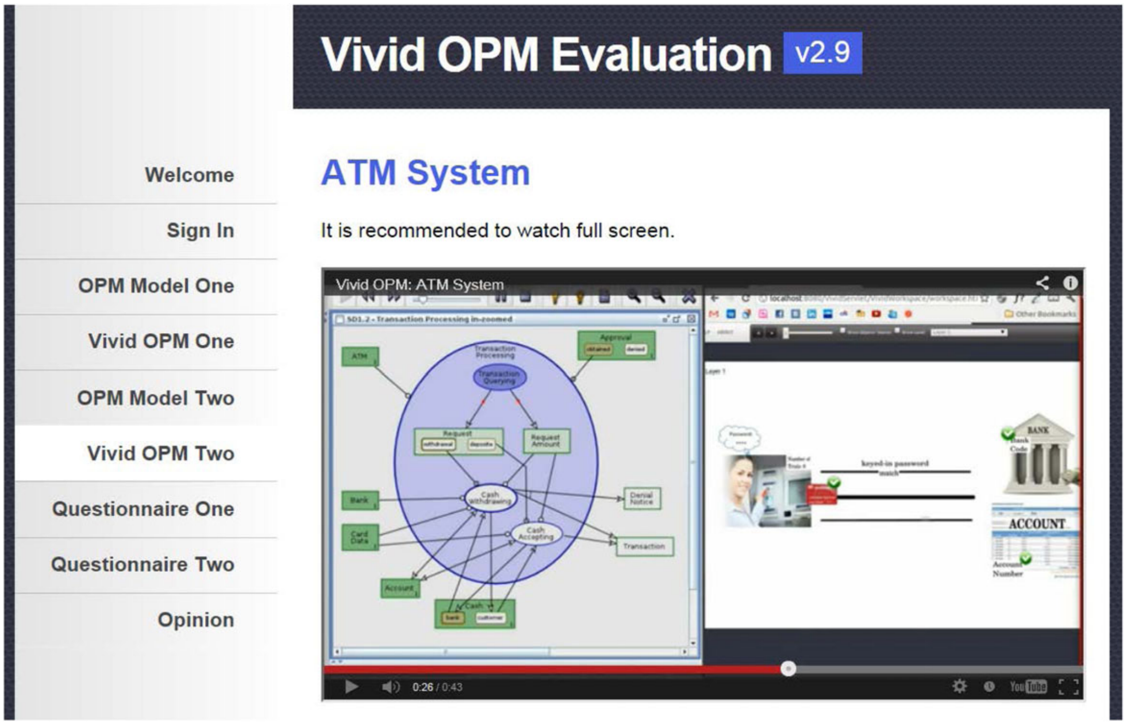

Figure 7. A snapshot of the pre-prepared movie of the ATM system as presented to the experimental students within the Vivid OPM evaluation system.

For data collection, we used a qualitative method. We conducted an achievement test, in which we examined the students' outcome in the course. The achievement test contained 18 closed questions that test the 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, (2) Behavior, (3) State change, and (4) Interaction.

We conducted explanatory factor analysis, which indicated four factors that match the four test question categories. The total explained variance was 41.67%. The four question categories 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 I. The MANOVA result was significant for research group by aspect, F(4,149) = 3.57, $p < 0.01$, $\eta_p^2 = 0.09$, indicating that there were significant differences between the experimental and control groups in the number of correct answers. In order to detect the source of the differences, we conducted a separate analysis of variance for each of the four question categories. The results indicated significant differences only in the interaction question category: F(1,152) = 14.11, $p < 0.001$, $\eta_p^2 = 0.09$. No significant differences were detected in the structure question category, behavior question category, and change of state question category.

Our experimental results indicate that overall, Vivid OPM enhances model understandability. However, it does 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 of Vivid OPM. Only when really complex questions are posed, such as those belonging to the interaction category, Vivid OPM provides a significant added value: 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. Its benefit is manifested in disambiguating complicated situations and aiding in making a model more comprehensible. While OPM with its graphic and textual modalities is sufficient for correctly answering questions of the first three categories—structure, behavior, and

Table I. Students' Correct Answers Means, Standard Deviations, and F Results for Differences between Experimental and Control Groups in the Four Aspects of Modeling

	Experimental Group: With Vivid OPM		Control Group: Without Vivid OPM		
Question Category	M	SD	M	SD	*F (1,152)*
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***

**$p \leq 0.001$.

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.

6. SUMMARY AND FUTURE DEVELOPMENT

We have developed, presented, tested, and performed a preliminary evaluation of a visualization addition to OPM, Vivid OPM. The Vivid OPM module enables animation of a conceptual OPM model that makes it closer to reality than the original OPM conceptual model. It translates an animated conceptual OPM model, which is relatively abstract, into a spatio-temporal clip that is readily understandable by domain experts who need not know the underlying OPM modeling language. The OPM model presents both the structure and the behavior of the system under study using a static set of diagrams. While the diagrams can be animated, the animation is of the diagrams: symbols of objects (denoted as boxes or rectangles), states (rounded-corner rectangles within the object boxes), and processes (ellipses), which 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. It is possible to determine the duration of each process separately. 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 which look like the objects, 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 more concrete and more comprehensible. The level of effort required to create the iconic view of the conceptual model depends on the complexity of the model and the extent of fidelity of the desired animation.

The obvious benefit of this presentation and play-out mode is that it provides dynamic visualization of the behavior of the system under study, which is driven by the conceptual model. Any change to the conceptual model is automatically reflected in the spatio-temporal Vivid OPM clip. The expected end users of Vivid OPM are scientists and system developers wishing to make the conceptual model concrete and "alive." Vivid OPM has been applied in several domains, including healthcare [Erkoyuncu, 2013], where an "as is" and several "to be" business models of using a hand-held scanner by primary care physicians were developed and presented to decision makers. A YouTube video clip of Vivid OPM of this system is available at https://www.youtube.com/watch?v=p-6IsKj7ZlU. These were accepted with great appreciation and were very instrumental in making business decisions. The idea of animating a conceptual model simulation can be adopted by simulation capabilities of certain diagram types of other languages such as state machine diagrams and activity diagrams of UML and SysML.

We plan to enrich the model in several directions, including improved graphics and multimedia, interaction, and support of 3D objects using CAVE [Dori 2012]. Multimedia enhancements include the ability to embed video and sound in the animation. Control of the simulation flow will enable saving the animation as a video clip, adding breakpoints, and playing in reverse, as well as adding haptic devices for enhanced realization. These enhancements will contribute to making Vivid OPM more effective as both an educational and a research tool.

REFERENCES

J.R. Binder, C.F. Westbury, K.A. McKiernan, E.T. Possing, and D.A. Medler, Distinct brain systems for processing concrete and abstract concepts, J Cogn Neurosci 17(6) (2005), 905–917.

S. Bolshchikov, Vivid OPM: mRNA Partial Life-cycle. http://youtu.be/3MgvUZXnYVs, accessed December 15, 2013.

D. Dori, *Object-process methodology—A holistic systems paradigm*, Springer Verlag, Berlin, Heidelberg, New York, 2002.

D. Dori, Extending the human spatiotemporal comfort zone with CAVERN – Computer-based Augmented Virtual Environment for Realizing Nature, J Multidiscip Res 4(3) (2012), 23–44.

D. Dori, S. Bolshchikov, and N. Wengrowicz, "Conceptual models become alive with Vivid OPM: How can animated visualization render abstract ideas concrete?," in D. Gianni, A. D'Ambrogio, and A. Tolk (Editors), *Modeling & Simulation-based Systems Engineering Handbook*, CRC Press, pp. 295–322, CRC Press, 2014.

D. Dori, C. Linchevski, and R. Manor, OPCAT – A software environment for object-process methodology based conceptual modelling of complex systems. *Proceedings of the 1st International Conference on Modelling and Management of Engineering Processes*, in P. Heisig, J. Clarkson, and S. Vajna (Editors). Cambridge, UK: University of Cambridge, 2010, pp. 147–151, July 19–20.

D. Dori and Mordechai Choder, "http://www.plosone.org/article/info:doi%2F10.1371%2Fjournal.pone.0000872" Conceptual Modeling in Systems Biology Fosters Empirical Findings: The mRNA Lifecycle. Proceedings of the Library of Science ONE (PLoS ONE), September 12, 2007. "http://www.plosone.org/article/info:doi%2F10.1371%2Fjournal.pone.0000872" http://www.plosone.oSYSCONrg/article/info:doi%2F10.1371%2F journal.pone.0000872

J.A. Erkoyuncu, S. Bolshchikov, D. Steenstra, R. Rajkumar, and D. Dori, Application of OPM for the healthcare sector. *Conference on Systems Engineering Research (CSER'13)*, in C.J.J. Paredis, C. Bishop, and D. Bodner (Editors). Procedia Computer Science 16, 2013, pp. 413–422.

J. Forrester, Counterintuitive behavior of social systems, Technol Rev 73(3) (1971), 52–68.

M. Fowler, *Analysis patterns: Reusable object models*, Addison-Wesley Longman, Essex, UK, 1997.

D. Harel and R. Marelly, Specifying and executing behavioral requirements: The play-in/play-out approach, Softw Syst Model 2(2) (2003), 82–107.

IBM Corporation, Systems Engineering Tutorial for Rational Rhapsody, 2009. http://publib.boulder.ibm.com/infocenter/rsdp/v1r0m0/topic/com.ibm.help.download.rhapsody.doc/pdf75/tutorial_Systems_Eng.pdf. Accessed December 15, 2013.

D. Lane, Diagramming conventions in system dynamics, J Oper Res Soc 51(2) (2000), 241–245.

S. Mellor and M. Balcer, *Executable UML: A foundation for model-driven architecture*, Addison Wesley, Boston, MA, 2002.

No Magic, Inc. (2013). Cameo Simulation Toolkit. http://www.nomagic.com/products/magicdraw-addons/cameo-simulation-toolkit.html, accessed December 15, 2013.

Object Management Group, Semantics of a Foundational Subset for Executable UML Models, http://www.omg.org/spec/FUML/1.0, OMG Document Number: ptc/2010-03-14, 2010.

OPCAT – OPM modeling software (2013). http://esml.iem.technion.ac.il/?page_id=874 Enterprise Systems Modeling Laboratory, Technion. Downloadable free of charge.

M.K. Sein and R.P. Bostrom, Individual differences and conceptual models in training novice users. Hum-Comput Interact 4(3) (1989), 197–229.

J. Somekh, M. Choder, and D. Dori, Conceptual model-based systems biology: Mapping knowledge and discovering gaps in the mRNA transcription cycle, PLoS One 7(12) (2012), e51430.

J. Somekh, G. Haimovich, A. Guterman, D. Dori, and M. Choder, Conceptual Modeling of mRNA Decay Provokes New Hypotheses. PLOS ONE, Sept. 2014. “http://www.plosone.org/article/info%3Adoi%2F10.1371%2Fjournal.pone.0107085” PLoS ONE 9(9): e107085. doi:10.1371/journal.pone.0107085.

Y. Yaroker, V. Perelman, and D. Dori, An OPM conceptual model-based executable simulation environment: implementation and evaluation. Syst Eng 16(4) (2013), 381–390.

Sergey Bolshchikov is a software engineer and developer advocate at Wix.com with great passion to technology and knowledge sharing. He is the co-organizer of the YouGottaLoveFrontend Conference and Ember-IL meetup group in Tel Aviv, Israel. He received his MSc in Information Systems Management from Technion, Israel Institute of Technology in 2013.

Judith Somekh, PhD, is a post-doctoral fellow at the Department of Biomedical Informatics, Harvard Medical School. She received her BSc in Information Systems Engineering in 2001. She received her MSc in 2007, and Ph.D. in 2013 in Information Systems Management from Technion, Israel Institute of Technology. Her research interests are systems biology with focus on modeling molecular biology systems and using computational approaches to understand the etiology of complex diseases in general and autism in particular.

Shay Mazor is an information systems engineer, currently working at the field of logistics systems analysis and their implementations in the SAP business product platform. He received his BSc in Information Systems Engineering from Technion, Israel Institute of Technology in 2012.

Niva Wengrowicz is Adjunct Lecturer at the Faculty of Industrial Engineering and Management, Technion, Israel Institute of Technology. She received her Ph.D. from Bar Ilan University in 2012 and was a Postdoctoral Fellow at the Department of Education in Science and Technology, Technion. She was also Visiting Scholar in the Engineering Systems Division, Massachusetts Institute of Technology (MIT). Her current research focuses on alternative assessment methods in higher education engineering programs using technology-enhanced learning environments.

Mordechai (Motti) Choder is Professor of Molecular Biology at the Faculty of Medicine, Technion, Israel Institute of Technology. During 2011-12012 he was Visiting Professor at Broad Institute, MIT. He received his PhD in Genetics in 1988 and his MSc in Biochemistry in 1982, both from Weizmann Institute of Science. He received his BSc in Biology from Tel Aviv University in 1978. He was Postdoctoral Fellow at Harvard and MIT during 1988-1991. His research concerns the cross-talk and integration between all distinct processes and the various stages of gene expression, which is critical for ability of the cell to function as a system. His studies have led us to novel concepts, including mRNA imprinting, Synthegradosome, and mRNA coordinators, which were published in top-tier journals.

Dov Dori is Chaired Professor in Industrial and Systems Engineering and Head of the Enterprise System Modeling Laboratory at the Faculty of Industrial Engineering and Management, Technion, Israel Institute of Technology. Since 2000 he has been intermittently Visiting Professor at MIT’s Engineering Systems Division. He received his PhD in Computer Science in 1988 from Weizmann Institute of Science, MSc in Operations Research from Tel Aviv University in 1981, and BSc in Industrial Engineering and Management from Technion in 1975. His research interests include model-based systems engineering, conceptual modeling of complex systems, systems architecture and design, software and systems engineering, and conceptual model-based systems biology. Prof. Dori invented and developed Object-Process Methodology (OPM), recently adopted as ISO 19450. He has authored over 300 publications, including journal and conference papers, books, and book chapters. Prof. Dori has mentored over 50 graduate students. He was Associate Editor of IEEE Transaction on Pattern Analysis and Machine Intelligence, and currently he is Associate Editor of Systems Engineering. He is Fellow of INCOSE–International Council on Systems Engineering, Fellow of IAPR–International Association for Pattern Recognition, Member of Omega Alpha Association–International Honor Society for Systems Engineering, and Senior Member of IEEE and of ACM.