Model-Based Protocol Engineering: Specifying Kerberos with Object-Process Methodology

Yaniv Mordecai
Faculty of Industrial Engineering and Management
Technion – Israel Institute of Technology
Haifa, Israel
yanivmor@technion.ac.il

Dov Dori
Engineering Systems Division
Massachusetts Institute of Technology
Cambridge MA, USA;
Faculty of Industrial Engineering and Management
Technion – Israel Institute of Technology
Haifa, Israel
dori@mit.edu

***Abstract*—We overview and demonstrate a framework for model-based protocol engineering. Engineering standards, protocols, and standardized procedures, especially in computing and communication, must be structured, well-defined, and simple to understand, so they can gain wide acceptance and adoption. Yet, standards and protocols often lack an underlying formalism required to ensure their integrity, consistency, and traceability. Lack in these aspects can cause delay in standard publication, distribution, adoption, implementation, and utilization, and might discredit the standard and harm its sponsors' reputation. Model-based protocols are easier to understand, implement, test, and integrate with existing products, solutions, and systems. Our approach draws on the recently adopted ISO 19450 standard of Object-Process Methodology (OPM)—a holistic systems engineering paradigm for modeling complex, dynamic systems. We demonstrate the benefits of our model-based approach by employing OPM to specify the widely-accepted Kerberos protocol for networked computer user and device authentication.**

***Index Terms*—Model-Based Systems Engineering, Model-Based Protocol Engineering, Object-Process Methodology, Kerberos.**

I. Introduction

Computer and communication protocols are important structural and procedural anchors in the inherently unstructured world of interconnected software and hardware. The flexibility of software code, the agility of software-defined hardware, and the multitude of technologies, vendors, and applications is a constant motivation to attempt formulating new, specialized, proprietary solutions to existing problems. Such solutions often have significant commercial implications due to the need to connect billions of devices around the world. In this highly dynamic environment, the prominence of standard, widely-accepted protocols is key to the interoperability and broad adoption of emerging technologies. Protocol formulation also requires robust and solid standardization framework foundations. Indeed, many protocol standardization bodies "reinvent the wheel" by defining dedicated semantics and syntax to describe their protocols, making it difficult to incorporate the protocols into the architecture and design of existing systems, products, and solutions.

Object-Process Methodology (OPM) is a holistic conceptual modeling framework that captures the functional, structural, and behavioral aspects of a system using minimal ontology and simple semantics, with a single diagram type and bimodal graphical-textual representation. OPM has been adopted as ISO 19450 publically available specification (PAS) for model-based systems engineering and as the basis for a new generation of model-based technical standards, and is currently in its final publication stage as PAS that is freely downloadable.

Recent efforts to introduce an OPM model-based standardization approach into the traditional standardization domain have proven to add value to structured enterprise and industry standards in terms of coherency, completeness, and traceability—critical attributes of standards-based operation. The alternative, a wearisome text-based effort to define the standard protocol and maintain its integrity and consistency, often results in deficiencies, which are discovered eventually through field trial and error rather than by careful design. These deficiencies render the protocol unusable in certain settings and focus attention on correcting errors detected in the protocol rather than introducing improvements and extensions to it.

To tackle these problems, we introduce a model-based protocol engineering framework, which enables the structured, formal, and interoperable definition, architecting, design, development, configuration, documentation, and maintenance of computer software and hardware protocols. Founded on solid, well-defined conceptual modeling basis, our approach not only ensures the protocol's internal consistency and the elimination of potential contradictions and ambiguities; it also enables the incorporation of the protocol into the design of new systems and products. We illustrate our framework via a high-level visual formal specification of the Kerberos protocol, which is used for authentication of devices across computer communication networks.

The rest of this paper is organized as follows: Section II provides a brief description of OPM and its benefits in tackling

protocol engineering. The Kerberos protocol is shortly described in section III along with reasons for using it to demonstrate our approach. In section IV we describe a preliminary OPM model of Kerberos. Section V summarizes the paper and discusses future research and applications.

II. Object-Process Methodology

Our modeling framework, Object–Process Methodology (OPM) [1], is a holistic conceptual modeling methodology for architecting, design, and analysis of complex systems, products, and processes. OPM is among the six leading modeling and design methodologies [2], and as a leading conceptual modeling framework [3], tried and embedded as a modeling framework in various complex socio-technical systems programs in domains such as aerospace and defense, information systems, medicine, biochemistry, embedded software, and space exploration. OPM is domain-independent; it provides an integrated representation of form, function, and behavior. Catering to complex system and process modeling, OPM's static-dynamic and structural-procedural unified view has proven to alleviate complexity management [4].

OPM's free CASE tool, OPCAT[1] [5] is a free modeling tool implementing OPM notation. provides robust simulation capabilities, shown to enhance understanding of behavioral and procedural aspects, which are otherwise difficult to perceive in static visual models [6].

OPM accommodates the modeling, design, authoring, and formal verification of standard processes and procedures, providing a basis for a new model-based standards authoring paradigm [7], [8]. Recently, OPM has been approved—after a technical due diligence process that has lasted for several years—as ISO 19450 standard—technical publically available specification (PAS): Automation systems and integration – Object-Process Methodology[2], and as the basis for a new generation of model-based ISO standards.

OPM is bimodal: it is both graphical and textual: each model notion captured visually in an OPM diagram (OPD) is accompanied by an automatically generated formal textual description in Object-Process Language (OPL). OPM's modeling elements are **Process** (ellipse), **Object** (rectangle), and **state** (*rountangle*). These elements can be connected by structural and procedural **Links**, graphically capturing various types of relations. An OPM metamodel describing objects, processes, and structural links is shown in Fig. 1. Procedural links and states are shown in Fig. 2. The OPL sentences were copied into the diagram on top of their respective visualizations (they are not part of the original diagram).

OPDs are hierarchically organized, interdependent, and self-similar. Each OPD can be extended to lower levels through unfolding, optimized for structural and functional decomposition, in-zooming, for procedural and behavioral elaboration, or view derivation, used for elaboration of model aspects around pivotal model elements.

OPCAT provides a graphical user interface to manage and draw OPDs, and auto-generate formal textual OPL specifications for each diagram and for the entire model.

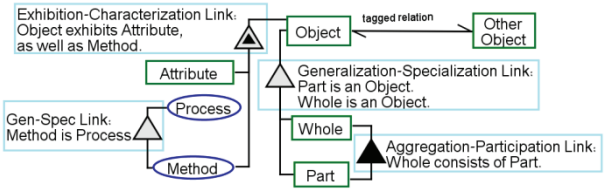

Fig. 1. OPM Notation (a) – Objects, Processes, and Structural Relations

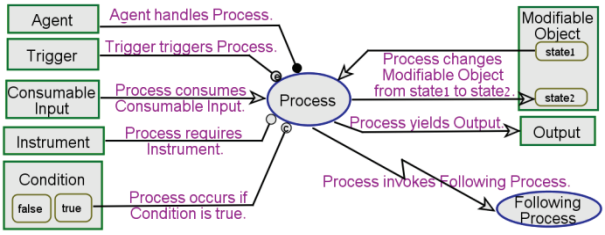

Fig. 2. OPM Notation (b) – Procedural Relations and State Dynamics

III. Kerberos

The Kerberos protocol is a common, widely-accepted protocol for computer user and device authentication across insecure computer networks [9], [10]. Kerberos was invented and developed at MIT with the support of leading information technology industry corporates, including Microsoft, Google, Apple, and Oracle. The Kerberos protocol is relatively simple and straightforward, and it has been applied in several other cases to test methodological frameworks [11].

A client computer attempting to connect to a server is required to provide proof of its authenticity, verifying its identity and access to the server. In order to be authenticated, the client must contact the Kerberos 3rd party service and receive a ticket indicating that it is who it says it is. A simple diagram of Kerberos is shown in Fig. 3 [9].

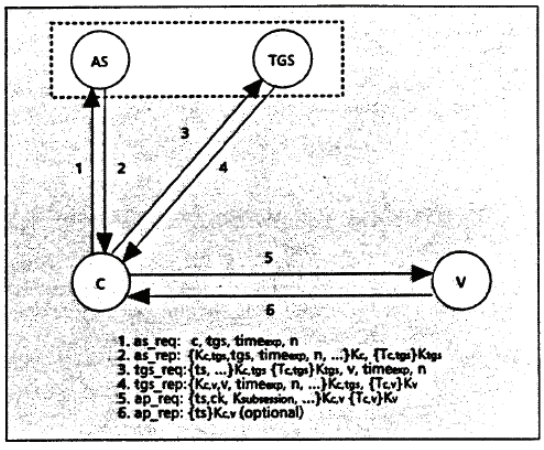

Fig. 3. Simplified View of Kerberos Authentication Flow [9]

[1] Downloadable for free from http://esml.iem.technion.ac.il/

[2] See ISO 19450 page on ISO website: http://www.iso.org/iso/home/store/catalogue_ics/catalogue_detail_ics.htm?ics1=25&ics2=040&ics3=01&csnumber=62274

Node C is a client, node V is an Application Server, node AS is the Kerberos Authentication Server, and node TGS is the Kerberos Ticket Granting Server. Initially, a thorough authentication process is required in order to issue a fundamental authentication ticket-granting ticket (TGT) to the client (steps 1-2). Then, each time the client needs to contact the server, it presents the Kerberos service with the TGT and receives a one-time authentication ticket (steps 3-4). The authentication ticket is provided in turn to the server, so the server can now allow access to the client (steps 5-6).

The MIT Kerberos Consortium has not provided a well-formed model-based representation of the Kerberos protocol [10], but a more detailed model of Kerberos, using UML sequence-diagram-like notation is shown in Fig. 4 and [12]. This is originally a single diagram, split in this paper only for formatting reasons. The first part (Fig. 4) shows first a preconditions block, which is not part of the sequence diagram. The client's request to the Authentication Server is shown next, but no response from the AS is shown. The second part (Fig. 5) shows the client's request to the Ticket Granting Server, followed by the TGS's internal authentication process and the ticket generating process. Then the client's access request is sent to the Application Server (a file server in this case), followed by the file server's internal authentication process and final access confirmation. After this process ends, a secure service session may begin.

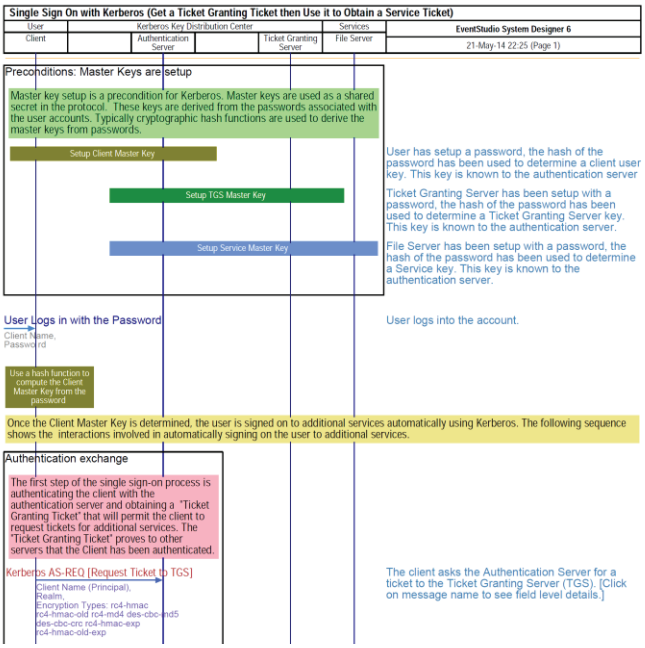

Fig. 4. Kerberos Sequence Diagram – part 1 [12]

There are several issues with this UML-like visualization. This view was not acquired from a refereed academic publication, but its availability to a wide audience on the web intensifies the problem it manifests. Without validating the process itself, especially the internal processes, there are several notation problems, a mixture of semantics from the sequence diagram, activity diagram, and some informal diagrams, a confusing use of rectangular blocks for the description of different types of entities (processes and objects), extensive use of in-diagram free text, and lack of capability to execute the visualized model (which is only available in pdf format). In addition, the sequence diagram does not show branches of the process, i.e., what happens when one of the subsequences fails. It does not capture more than the first two levels of the internal processes conducted by each participant, and it does not help alleviate visual cognitive load as more information is added.

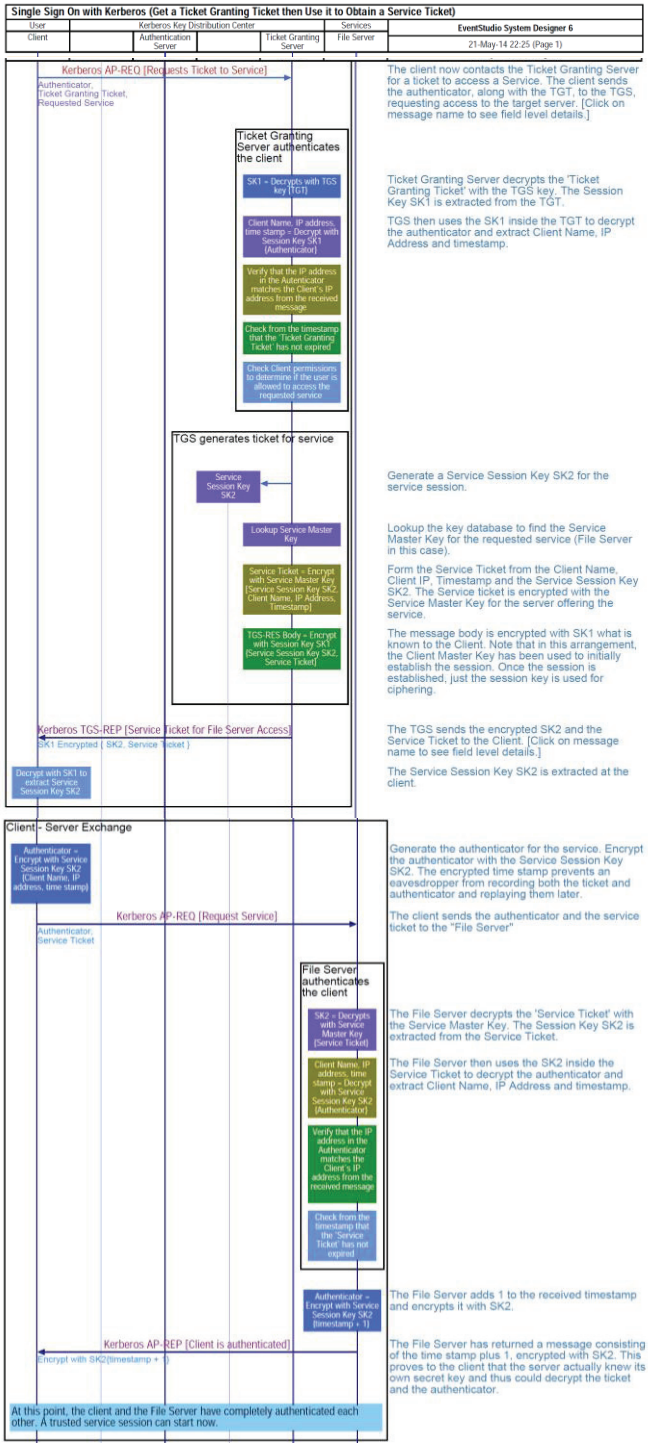

Fig. 5. Kerberos Sequence Diagram – part 2 [12]

IV. An Object-Process Model of Kerberos

A preliminary high-level specification of the Kerberos authentication protocol is illustrated visually and textually in Fig. 6 and Fig. 7, respectively. The object-process diagram (OPD) in Fig. 6 shows (1) the primary actors – **Kerberos Server**, **Client**, and **Application Server**, defined as physical objects, visualized as shaded rectangles, (2) the various processes (methods) each of these actors exhibits, and (3) the informatical objects that are exchanged among the actors. The model was created using OPCAT. Therefore, the textual specification in Fig. 7 is equivalent to and coordinated with the visual description: Each OPL statement reflects an aspect captured visually in the OPD. We have also auto-numbered the OPL statements for assisting with formal verification (but numbering is not necessarily kept when the model changes).

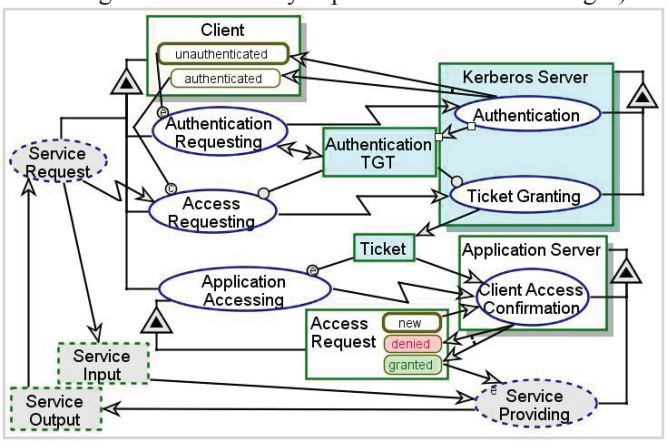

Fig. 6. Top-Level OPM Diagram of the Kerberos Authentication Protocol

ID	Statement
1000	Client is physical.
2000	Client can be unauthenticated or authenticated.
2100	unauthenticated is initial.
3000	Client exhibits Authentication Requesting, Application Accessing, Access Requesting, and Service Request.
3100	Authentication Requesting requires unauthenticated Client.
3200	Authentication Requesting affects Authentication TGT.
3300	Authentication Requesting invokes Authentication.
3400	Application Accessing exhibits Access Request.
3410	Access Request can be denied, granted, or new.
3411	new is initial.
3420	Access Request triggers Service Providing when it enters granted.
3500	Application Accessing requires Ticket.
3600	Application Accessing invokes Client Access Confirmation.
3700	Access Requesting occurs if Client is authenticated.
3800	Access Requesting requires Authentication TGT.
3900	Access Requesting invokes Ticket Granting.
4000	Service Request is environmental.
4100	Service Request consumes Service Output.
4200	Service Request yields Service Input.
4300	Service Request invokes Access Requesting.
5000	Client triggers Authentication Requesting when it enters unauthenticated.
6000	Kerberos Server is physical.
7000	Kerberos Server exhibits Authentication and Ticket Granting.
7100	Authentication yields Authentication TGT.
7200	Authentication yields either unauthenticated Client or authenticated Client.
7300	Ticket Granting requires Authentication TGT.
7400	Ticket Granting yields Ticket.
8000	Application Server is physical.
9000	Application Server exhibits Client Access Confirmation and Service Providing.
9100	Client Access Confirmation consumes new Access Request and Ticket.
9200	Client Access Confirmation yields either denied Access Request or granted Access Request.
9300	Service Providing is environmental.
9400	Service Providing consumes Service Input and granted Access Request.
9500	Service Providing yields Service Output.
10000	Ticket triggers Application Accessing.
11000	Service Output is environmental.
12000	Service Input is environmental.

Fig. 7. Auto-Generated OPL Top-Level Specification of Kerberos

The **Authentication** process, exhibited by **Kerberos Server** and visualized as an ellipse, changes the state (rountangle) of **Client** from unauthenticated to authenticated and issues an **Authentication TGT**, which is an informatical object. A **Client** has to be authenticated in order to hold a valid **Authentication TGT**. Access requests are triggered by service requests, which are environmental, i.e., external to the protocol. Using its **TGT**, the **Client** than asks the **Kerberos Server** for a one-time access **Ticket**. The **Ticket** is provided, and the **Client** can use it to request access from the **Application Server**. The **Application Server** confirms the **Client's** authentication and grants (or denies) the **Access Request**. This allows for the **Client**'s **Service Request** to engage the **Application Server**'s Service, which is also external to the protocol.

Each process can be further extended in a separate OPD, and to elaborate the subprocesses it includes, as well the inputs it requires and the outputs it provides. This approach helps alleviate complexity by exposing more information at lower levels in a hierarchically-organized manner and using a single, uniform notation for all diagrams. For instance, the Client Access Confirmation process that is executed by the Application Server is zoomed into and extended as shown in Fig. 8.

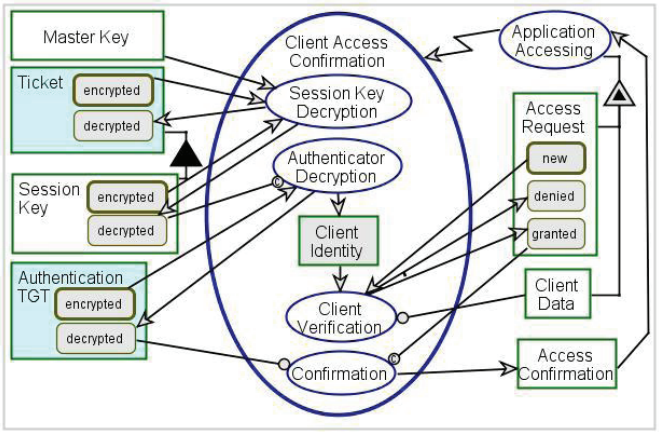

Fig. 8. In-zoomed OPM Diagram of the Application Server's Client Access Confirmation process

ID	Statement
1000	Authentication TGT can be encrypted or decrypted.
1100	encrypted is initial.
2000	Ticket can be encrypted or decrypted.
2100	encrypted is initial.
3000	Ticket consists of Session Key.
3100	Session Key can be decrypted or encrypted.
3110	encrypted is initial.
4000	Application Accessing exhibits Access Request and Client Data.
4100	Access Request can be denied, granted, or new.
4110	new is initial.
5000	Application Accessing consumes Access Confirmation.
6000	Application Accessing invokes Client Access Confirmation.
7000	Client Access Confirmation exhibits _ and Client Identity.
8000	Client Access Confirmation consists of Session Key Decryption, Authenticator Decryption, Client Verification, and Confirmation.
9000	Client Access Confirmation zooms into Session Key Decryption, Authenticator Decryption, Client Verification, and Confirmation, as well as Client Identity.
9100	Session Key Decryption changes Session Key from encrypted to decrypted and Ticket from encrypted to decrypted.
9200	Session Key Decryption consumes Master Key.
9300	Authenticator Decryption occurs if Session Key is decrypted.
9400	Authenticator Decryption changes Authentication TGT from encrypted to decrypted.
9500	Authenticator Decryption yields Client Identity.
9600	Client Verification requires Client Data.
9700	Client Verification consumes new Access Request and Client Identity.
9800	Client Verification yields either granted Access Request or denied Access Request.
9900	Confirmation occurs if Access Request is granted.
10000	Confirmation requires decrypted Authentication TGT.
10100	Confirmation yields Access Confirmation.

Fig. 9. Textual description in OPL of the Application Server's Client Access Confirmation process OPD.

The corresponding textual specification in OPL is provided in Fig. 9. This clause may include both recited statements from previous OPL clauses relating to aspects that reappear in the current diagram, and additional statements referring to new model aspects defined at this level. In this case, we can see for instance the addition of encryption state information regarding the Kerberos Ticket and TGT objects. Once this information is added to a particular diagram, it applies to the entire model, including diagrams that have already been defined. Therefore, it is important to use the model as a vivid representation of the system rather than a set of images.

Due to the limited extent of this paper, we do not provide the complete Kerberos protocol model, but the mechanism to further extend the model of the protocol is clear and simple, as more self-similar, lower-level OPDs can be added to the model in order to specify internal subprocesses of each process or function in the model. Dedicated views can be generated in order to highlight interesting or important aspects, such as end-to-end state transition diagrams. All the diagrams in the OPM model are cross-validated, associated, and integrated, so any OPD in the final model is consistent with the rest of the OPDs.

The entire model can be simulated and validated, verified, and tested using OPCAT's simulation mechanism. In addition, the protocol model can be integrated into other OPM system models that implement the protocol. The external Service Request and Service Providing processes, as well as their associated inputs and outputs, are placeholders for actual functional system processes. Thus, the basic protocol model can also be developed to display both larger-scope aspects of the system and details of processes of interest. In the Kerberos case this may be less relevant since Kerberos is an infrastructure protocol which is independent of functional data and interaction semantics and purposes. However, when modeling standard business transactions, such as financial credit transactions, payment transfers, or order placing, which depend on user inputs, protocol executing results are affected by the input, making an even more compelling case for the need for robust integration with the system into which a standard is integrated.

V. Summary

We have discussed the problem of lack of using adequate model-based methods to define and formalize standards in general and hardware-software protocols in particular. As a case in point, we analyzed existing models of the Kerberos protocol for computer authentication and identified several deficiencies, inconsistencies, and visualization issues. We then introduced OPM and its potential role in modeling standard technological protocols and integrating them into complex system models. We have shown how OPM successfully copes with modeling challenges such as consistent notation, complexity handling, unified functional-structural representation, and graphical-textual specification equivalence.

As OPM is already approved as an ISO standard, we are transitioning from the methodological validation phase to the practical applicability demonstration phase, in which we show that OPM can be usefully applied to model various types of standards and protocols, serving as the foundation for a new generation of model-based engineering standards. Future work also focuses on improving and enriching OPL, making it more natural and appealing to human readers. For this purpose, domain-related vocabularies need to be integrated into the language. The primary limitation of our framework is its current low adoption rate and the need to migrate from the common UML de-facto standard. Recent progress with ISO standardization can be a motivating factor for organizations and individuals to make this valuable transition.

Acknowledgment

This research was funded by the European Union's 7th Framework Programme (FP7/2007-2013) under grant agreement No. 262044 – VISIONAIR.

References

[1] D. Dori, *Object-Process Methodology: A Holistic Systems Approach*. Berlin, Heidelberg, New York: Springer, 2002.

[2] J. Estefan, "Survey of model-based systems engineering (MBSE) methodologies," 2007.

[3] D. Dori, "Object-Process Methodology for Structure-Behavior Codesign," in *Handbook of Conceptual Modeling*, D. W. Embley and B. Thalheim, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 209–258.

[4] D. Dori, I. Reinhartz-berger, and A. Sturm, "Developing Complex Systems with Object-Process Methodology Using OPCAT 1 The Basis : Object-Process Methodology," pp. 570–572, 2003.

[5] D. Dori, C. Linchevski, and R. Manor, "OPCAT – An Object-Process CASE Tool for OPM-Based Conceptual Modelling," in *1st International Conference on Modelling and Management of Engineering Processes*, 2010, pp. 1–30.

[6] Y. Yaroker, V. Perelman, and D. Dori, "An OPM conceptual model-based executable simulation environment: Implementation and evaluation," *Syst. Eng.*, vol. 16, no. 4, pp. 381–390, 2013.

[7] A. Blekhman and D. Dori, "Model-Based Requirements Authoring - Creating Explicit Specifications with OPM," in *6th International Conference on Systems Engineering*, 2011.

[8] A. Blekhman, D. Dori, and R. Martin, "Model-Based Standards Authoring," in *21st INCOSE International Symposium*.

[9] B. Neuman and T. Ts'o, "Kerberos: An authentication service for computer networks," *Commun. Mag. IEEE*, vol. 32, no. 9, pp. 33–38, 1994.

[10] F. Ricciardi, "Kerberos Protocol Tutorial v1.0.3." MIT Kerberos Consortium, 2007.

[11] B. Księżopolski, D. Rusinek, and A. Wierzbicki, "On the modelling of Kerberos protocol in the Quality of Protection Modelling Language (QoP-ML)," *Ann. UMCS, Inform.*, vol. 12, no. 4, pp. 69–81, Jan. 2012.

[12] EventHelix, "Kerberos sequence diagram," 2014. [Online]. Available: Source: http://www.eventhelix.com/RealtimeMantra/Networking/kerberos. [Accessed: 01-Oct-2014].