Zbigniew ZIELIŃSKI[1], Andrzej STASIAK[1], Włodzimierz DĄBROWSKI[2,3]

Military University of Technology (1), Warsaw Institute of Technology, ISEP (2), Polish-Japanese Institute of Information Technology (3)

A Model Driven Method for Multilevel Security Systems Design

Summary. *The article presents the application of methods of simulation of UML models for analysis and designing specialized computer systems, processing data with multilevel security. The integration of security models with models of system described in UML enables their simulation, which allows identifying security problems at the stage of modeling. By using UML extensions and ALF language it is possible to build a topological model and perform efficient simulations of topological models. The authors used an IBM simulator for simulation.*

Streszczenie*. W artykule przedstawiono zastosowanie metod symulacji modeli UML do analizy i projektowanie komputerowych systemów specjalizowanych przetwarzających danych z ochroną wielopoziomową. Integracja modeli bezpieczeństwa z modelami systemu opisanych w UML umożliwia ich symulację, która pozwala na identyfikację problemów związanych z bezpieczeństwem na etapie modelowania. Za pomocą rozszerzeń i języka UML ALF jest możliwe zbudowanie topologii modelu i efektywnego wykonywania symulacji modeli topologicznych. (****Zastosowanie metod symulacji modeli UML do analizy i projektowanie komputerowych systemów specjalizowanych przetwarzających danych z ochroną wielopoziomową****)*

Keywords: multilevel security systems, system modeling, UML models simulation, configuration topology
Słowa kluczowe: systemy z ochroną wielopoziomową, modelowanie systemów, symulacja UML, topologia konfiguracji

Introduction

The issue of building reliable specialized computer systems (SCS) data processing at different levels of sensitivity is particularly topical, especially in regard to the uses of SCS in government, military or financial institutions. The problem of processing information with different levels of sensitivity has been intensively studied since the early 70s of the twentieth century [1,2,3]. Formal basics of the so-called multilevel security (MLS) are presented in the work of Bella-LaPaduli (B-LP) [2]. The computer system of multilevel security (MLS) is necessary to define the so-called allowance of users to work with classified information, as required by official duties carried out taking into account the principle of the "need to know basis" and classifying information, because of its level of sensitivity. The most commonly implemented model is the B-LP or its very similar modifications (Biba [4], Clark-Wilson [5], etc.). According to this model, granting subjects access to adequate resources is achieved by mandatory access control (MAC), which means that any subject (or process) and system resource (file data, communication channel, etc.) is attributed a security context. In order to determine the powers in MAC systems, labels are constructed with the security context, in particular the pair: <label sensitivity, information category>. On the set of labels of subjects and objects the partial order relationship is determined, and comparing the security level of a newly requested object to that of every object to which the subject currently has access is required [3]. An immutable rule must be imposed on the subjects and resources (objects), such as [3] for read accesses, the current security level had to be greater or equal to (or to “dominate”) that of the new object, for altering accesses, the current security level had to be less than or equal to (or be dominated by) that of the new object. It should be noted, that to implement in systems and networks a set of such rules (i.e. to build a dependable system solely based on an operating system with multilevel information security) is extremely difficult and expensive. This is mainly due to difficulties in building a reliable reference monitor and the difficulty of ensuring that the system will not "leak" of sensitive information due to the possible existence of so-called covert channels in the operating system [6].

One of the possible approaches to the development non-distributed a MLS computer system involves the use of virtualization technology and building software that acts as the manager of virtual machines [7,8]. For this approach to be trustworthy requires both the use of strictly virtualizable hardware, and a trustworthy monitor mechanism for separating the activities of the virtual machines. Such software should allow for the simultaneous launch of several instances of special operating systems acting as virtual machines on a single computer (workstation, server) dedicated for data processing of various clauses of sensitivity (e.g., non-classified, restricted), or to process data for which the separation is needed. This approach became fully possible thanks to modern Intel and AMD processors hardware support for virtualization solutions and COTS-type developed virtualization software package. Currently widely used x86 architecture extensions are such that support hardware virtualization [7,9] such as: Intel Virtualization Technology, in particular VTx, VTD - x86 and VTi - for Intel IA-64 (Itanium) and AMD Virtualization (AMD - V) for 64-bit x86 processors from AMD. These technologies also allow (in addition to virtual machine emulation hardware support) to build a trusted environment in which separate virtual machines (which are separate security domains) exist in separate hardware partitions. Implementation of such an MLS system project requires the integration of the available virtualization technology (software and hardware), application of formal methods for both ensuring and monitoring the confidentiality and integrity of data processing and user authentication techniques. The natural way of building such systems becomes a component approach, which assumes the use of prepared (available) hardware and software components, in particular, available open-source COTS virtualization packages as Xen [10] or KVM [11]).

The article proposes an innovative approach to designing a secure MLS-type SCS systems, which main objective was to provide tools to verify the confidentiality and integrity of data being processed on the basis of models, as well as a test of resistance of the whole system built on various types of attacks at the stage of its development.

The MLS-type SCS developed method of production, which we will define as the MDmls (Model Driven Multilevel Security) method, organizes the MLS-type SCS manufacturing process and derives from the MDA concept (Model Driven Architecture) and MDD (Model Driven Development) [12,13]. A similar approach to building secure software is shown in [14], but it does not include multilevel security issues. The article presents the basic assumptions of the method, but focuses mainly on describing the steps of the modeling process. This means that by using the concept of an executable model (xUML) [23]

implementation processes as those that are always associated with a specific platform (in accordance to the MDA) in the manufacturing cycle are carried out as implementation actions - independently for each PSM. Similarly, depending on the specificity of the system being built, usually it will be necessary to use a few domain languages (DSL), in the place of the universal language (UML). The integration of security models with models of systems described in UML enables their simulation, which allows identifying security problems of the SCS MLS-type software at the stage of modeling. The presentation of the proposed method shows the construction process of the metamodel, and profiles in compliance with UML version 2.2, and the domain-specific modeling (DSM) process [24, 25].

Method Origin

The method was developed for the project's research and development called "Secure Workstation for Special Applications (SWSA - Project funded by the Polish Ministry of Science and Higher Education under grant OR00014011 ; Polish project title: “Bezpieczna stacja do zastosowań specjalnych”), developed by a consortium led by the Military University of Technology in Warsaw. The aim of the project is to develop a trusted environment for processing of sensitive (or even classified) information from different security domains (either data from separate administrative domains or data with different classification levels; in both cases data that should be processed separately) on the same physical machine. This goal should be achieved through application of virtualization technology to provide different virtual machines with either Linux or Windows systems for each security level. SWSA system will be used by multiple users with different privileges.

In the SWSA architecture some elements can be distinguished such as a trusted system platform (TSP) – a SWSA component, including an operating system kernel and a virtual machines monitor as well as a executable special versions of operating systems (virtual machines). The general scheme of the SWSA architecture is shown in Figure 1. The virtual machines manager (VMM) is a key component of the SWSA, that is why we will begin by presenting its essential components and defining the role of VMM in the SWSA structure.

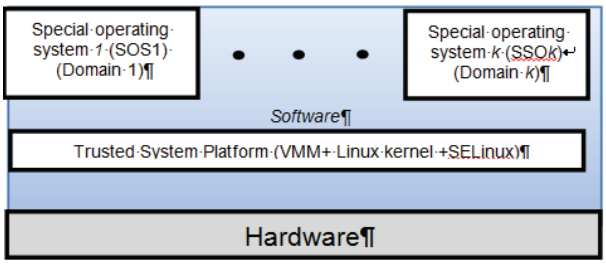


Fig. 1. Simplified SWSA architecture scheme

The key role of the VMM is to run virtual machines in accordance with defined rules of security policy (using hardware support) and their switching, with the possibility to ensure the separation of resources. The essential elements of the SWSA architecture consists of n virtual machines (VM) and the secure system platform (TSP), which we will describe in the form of the following formula: $\mathrm{SWSA} := \mathrm{TSP} + \{\mathrm{MV_i}\},\ i \in \{1,..,n\}$. TSP form the Linux operating system kernel, its extension as a component (SELinux - solution is incorporated into the Linux kernel via Linux Security Module (LSM) framework) and virtual machines manager (VMM), $\mathrm{TSP} := \mathrm{VMM} + \mathrm{Linux\ kernel} + \mathrm{SELinux}$. TSP allows you to run and supervise activities of specialized operating systems (SOSi), $i \in \{1,..,n\}$ and user programs operating in their environment {PU}, which constitutes a virtual machine $\mathrm{VM} := \mathrm{SOS} + \{\mathrm{PU}\}_p$.

Due to the ownership of the hardware design SWSA it was assumed that the proposed VMM software should allow launching a few (of several possible) instances of special versions of operating systems ($\mathrm{VM_i}$) on a single computer providing access control, cryptographic protection, and strict control of data flow. For example (Fig. 2), TSP supervises the operation of n virtual machines, from which two operate simultaneously ($\mathrm{MV_1}$ and $\mathrm{MV_2}$) and in accordance, $\mathrm{PU_{1.1}}$ is active on $\mathrm{MV_1}$ (which is described as $\mathrm{VM_2} \xrightarrow{\mathrm{Run}} \mathrm{PU_{1.1}}$) and $\mathrm{PU_{2.2}}$ is active on $\mathrm{MV_2}$ (i.e. $\mathrm{VM_2} \xrightarrow{\mathrm{Run}} \mathrm{PU_{2.2}}$). VMM manages access to both virtual machines and the hardware resources (physical and virtual).

The project also assumes that working within a virtual machine (VM) the instance of a special version of the operating system (SSO) is a separate security domain $\mathrm{MV_i[SSO_i]} := \mathrm{DB_i}$, and each of the domains allows data processing, qualified to different security levels. Fig. 2 shows the two security domains, and each of them is linked to one virtual machine.

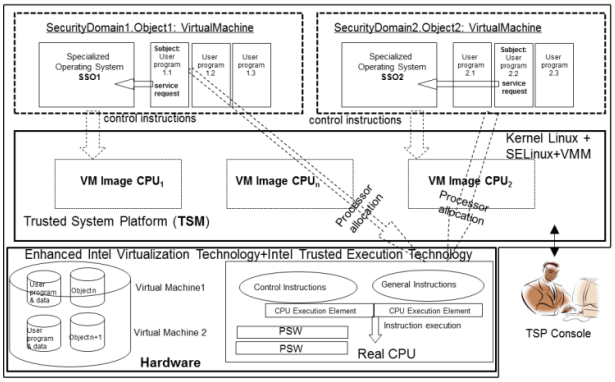


Fig. 2. Components of the TSP architecture

The main task of TSP is to control access, including user authentication by using different methods available and to prevent user access to virtual systems (special versions of SSOi), which does not have appropriate permissions.

The use of VMM-level hardware components is assumed, which enable security technology hardware support and hardware support virtualization, including Trusted Execution Technology TXT (Fig. 4) that allows the separation of distinct partitions of isolated execution environments, and including the TPM (Trusted Platform Module) component allows to create and securely store encrypted keys. A very important piece of hardware, from the viewpoint of the project, is the 5520 chipset, which supports features such as: Intel Quick Path Interconnect (QPI), Enhanced Intel Virtualization Technology (VT-d, VT-c), Intel TXT, Intel Matrix Storage Technology.

Method Scope

Since the project for a Secure Workstation for Special Applications (SWSA) refers to a complex computer system, its implementation requires the definition of solutions in the areas of hardware and software.

Support for Hardware Design Processes

The hardware project, due to the component based approach (using out-of-the-box components), will be limited

to determining the configuration, which describes the characteristics of the solution. The proposed SWSA hardware architecture is based on a machine with the Intel Westmere architecture, two processors Xeon E5630 model, each of which contains the four processors cores. The flowchart of the system based on Xeon 5600 series processors is shown in Figure 3 In particular, the connection between the processors and the chipset (I/O Controller) is noticeable, implemented using QPI (QuickPath Interconnect). In this scope of design process it is proposed to use the extended UML language with so-called topological models [15,16,17]. The creation of these models is supported by the CASE environment - Rational Software Architect (RSA).

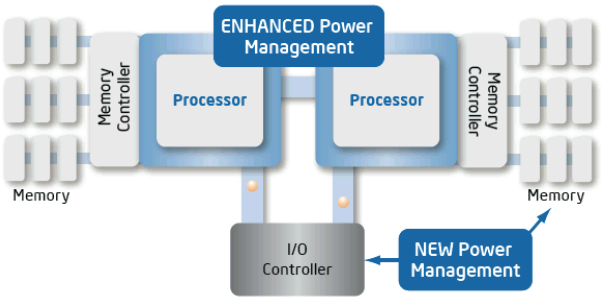


Fig. 3. Xeon 5600 processor series-based system flowchart

A very important consideration for the selection of this environment is to support virtualization mechanisms, in particular, Xen technology (supporting mechanisms for the safe switching between operating systems processes and management of virtualized resources.) It is worth noting that the topology models, which use the Xen virtualization package looks the same as those that use VMWare, except that Xen is represented by a "virtualization Xen Hypervisor" as a physical unit of the operating system, and not a separate configuration component. Let us note that equipping the RSA with an additional component of the Rational Software Architect Extension for Deployment Planning will allow for the direct generation of configuration files for target physical configurations, and additionally to read configuration parameters and create topological models based on the operating SWSA (its actual physical configuration).

Support for Software Design Processes

The software design problem boils down to building the trusted system platform (TSP), which includes: operating system kernel and virtual machines manager (VMM) and running special versions of operating systems. It is worth noting that both the operating system kernel, as well as special versions of operating systems in terms of the carried out project are ready components, and the project will be limited only to the specifications of their interfaces and configuration descriptions. Interfaces would be described in UML and the configurations on topological diagrams. In this scope, it is recommended to use CASE tools - Rational Software Architect (RSA).

The essential complexity of software design is thus reduced to the construction of VMM virtual machines manager, which will be responsible for its own implementation of the Xen hypervisor virtualization component, but it is worth noting that the work includes the implementation of its own unique solutions in this scope, in particular, it provides multilevel security policies. Due to the complexity of this part of the project a new designing method was proposed.

MDmIS (Model Driven Multilevel Security) Method

An important element of our proposal is the use of methods that allow (a formal) verification of the confidentiality and integrity of processed data with different levels of secrecy. The proposed approach of creating the SWSA is based on the specialization of the methodologies based on MDA (Model Driven Architecture) [12] by integrating elements of security models (secureUML, UMLsec) [26] with processing models expressed in UML.

Method Assumptions

In the process of creating the BPS the elaborations approach was rejected to the MDA, for the benefit of a new translations approach, assuming the building of an extended specification of the developed PIM models for the use of language semantics of ALF actions (currently UAL) with the possibility of running and debugging these models in the fUML environment.

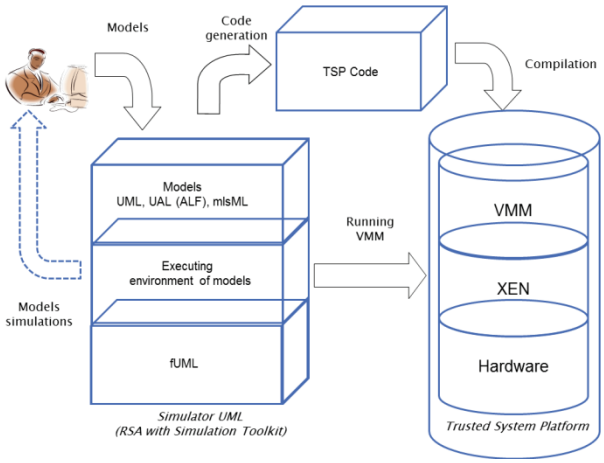


Fig. 4. Model of the TSP software development environment in accordance with MDmIS

At this stage, the IBM RSA ver. 8.0 extended environment was used with the Rational Software Architect Simulation Toolkit, with support for UML Action Language (UAL), which provides a subset of the specifications described in OMG with technologies fUML and ALF (Fig. 4). Because the ALF specification was accepted recently in February of this year, the authors predict that IBM will provide support for this standardized form of ALF action language semantics instead of UAL.

Even though in the selected CASE environment (IBM, RSA) there are no corporate transformations[1] for developed domain languages (DSL) security description (including SecurityUML or UMLsec) we assume that for research purposes, the use of the translationist approach for building systems in accordance with the MDA concept will provide us with satisfactory results, due to:

- availability of full support for the creation of domain languages in accordance with MDA[2];
- possibility of simulating developed models as a tool for verifying the integrity and confidentiality of data in the BPS environment. possibility of describing the SWSA configuration and its resources (both physical and

[1] Transformations provided by the manufacturer (IBM).

[2] Fulfilling this requirement is an important element of the developed MDmIS method and defined for its needs the mlsML language.

virtualized) and simulations of their behavior through the use of topological models (Fig. 2)[3];
- enabling the creation of formal models of secure systems, i.e. adding resources of expression extending UML and OCL in the form of language semantics actions (currently UML Action Language (UAL));
- enabling the use of the component approach to design and build the TSP architecture, which with MMW elements comprise ready components: SE Linux kernel module and the XEN hypervisor (Fig. 4) [15].

Method Processes

Due to the research nature of the conducted project, the simplified process of building the SWSA was divided into three activities: modeling, simulation (as a test model) and implementation (Fig. 5).

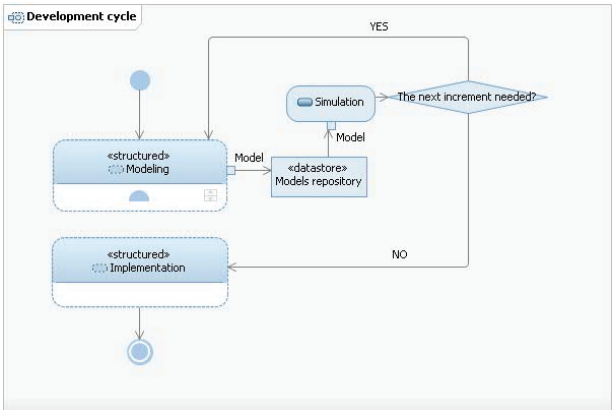


Fig. 5. SWSA development process model

From the perspective of project management, the transition to the implementation stage takes place after completing modeling, which we build the next release of "tested models" in an incremental and agile way. Implementation, however, is carried out but only after delivering the final version of the system model, and considerably makes up the result of automatic transformation of models into system code and descriptions of the required configuration. Therefore, the method assumes that all developed, in accordance with MDA, models are combined with transformations: speed manual or automatic (model-to-model [M2M], model-to-text [M2T]). The concept of the proposed transformation approach can be described with the following dependence:

$$\mathbf{CIM}[RM] \xrightarrow{\text{Manual}} \mathbf{PIM}[DM] \xrightarrow{\text{M2M}} \mathbf{PSM}[\{SM\}] \xrightarrow{\text{M2T}} \mathbf{IM}[\{Code\}],$$

where: RM - Requirement Model, DM - Domain Model, $\{SM\}$ – set of System Models, $\{Code\}$ – set of Implementation Models.

Domain Languages - Defined and Used in the MDmls Method

In the proposed method activities related to domain-specific modeling (DSM) have a key significance. Their course is implemented in the following steps: a sketch of the domain-specific language (DSL), design and implementation of domain language, creation of a UML profile for the domain language, the creation of pictographic representation of the UML profile, the development of model transformations.

[3] In this example, the password definition requirement was set, which failure to comply is signaled by a warning for the virtual machines (Guest1, Guest2).

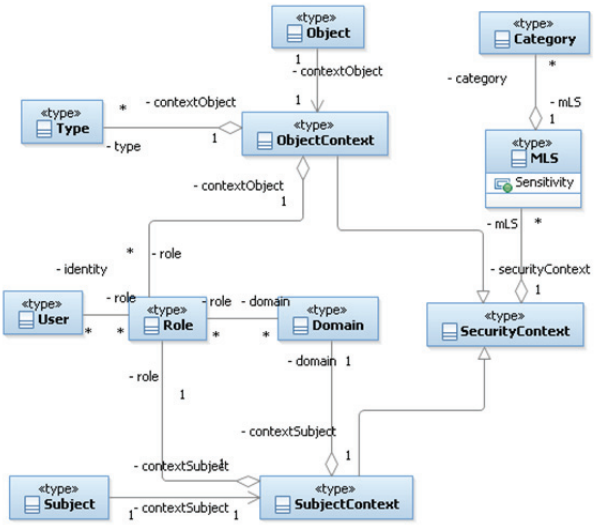


Fig. 6. DM4MAC domain model constitutes a base for defining the mlsML4MAC profile

These actions will allow the creation of precise views of the modeled system domains, providing each view with an autonomous DSL languages. During modeling, i.e., sketching, designing and implementing the domain language, i.e. during the creation of the metamodel the abstract syntax should be defined, which in turn will serve to describe the profile. Modeled by the built domain language constitutes the computer system in which the processes of processed data does not violate the multilevel security policies. However, in general, we should assume that the specialized domain language will streamline realizing tasks associated with modeling in the given domain of applications, not only facilitating the system design processes, but also communication between domain specialists building it, and applying model transformation technologies will enable maximizing the range of automation of code generation and configuration of the system being built (minimizing the costs associated with the system transmission to various specific target platforms). The proposed method also assumes the possibility of using other available domain languages such as SysML, SoaML, etc.

The creation of new Domain Specific Languages requires the metamodel created earlier, because only on its basis can the profile be specified. For the definition and representation of the metamodel the Meta-Object Facility standard (MOF) will be used promoted by the OMG organization. The built UML profiles (mlsML4MAC, mlsML4RBAC), within the proposed method, are designed to meet the following objectives:
- Integrate DSL with UML – UML notation is the base for the created profile, and knowledge contained in the profile is based on a metamodel also UML,
- Facilitate reuse – the elaborated domain languages will constitute a library, which you can always refer to by indicating the appropriate domain.

Because the built language was based on the concepts of MDA and MDD, therefore its primary purpose was to quickly reach the skeleton code of the implementing the product. DSL was built using predefined models (requirements, domains), which entailed a requirement to build an abstract syntax, identification of concepts, well-shaped detailing the roles, operations and limitations, as well as validation and testing of models.

The metamodel is a general solution, while the profile is its clarification. Metadata can describe any area of the

system and can include additional information. The three main elements of the metadata modeling are: class, association and package, the ones provided by MOF are similar to the elements of the UML standard, they are characterized only by a greater degree of simplification. Classes – may have attributes and operations on two levels: object and class, have the ability of multi-inheritance from other classes. Associations – make it possible to binary connect an instance of classes. Important elements defined in MOF are the limitations and data types. Constraints are used to connect the semantic restrictions, for their recording any language can be used. Data types allow the use of non-object types as attributes or parameters.

Method Stages

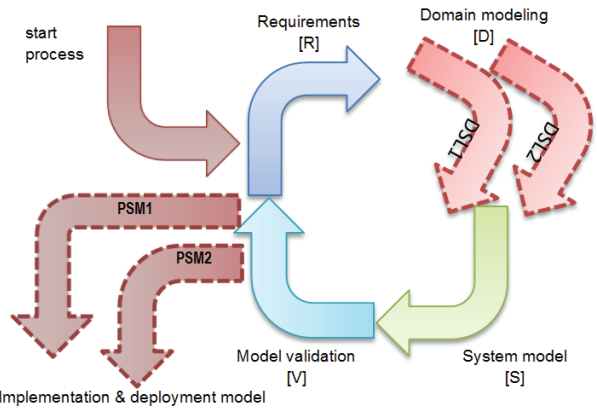
It is proposed to adopt the manufacturing process in accordance with RUP, i.e.: inception phase, elaboration phase, construction phase and transition phase.

Inception phase – as a result of this stage is the terminology domains of the system built are determined based on the requirements modeling process leading to the full identification of domain problem. The main result of this process is the metamodel and a sketch of the domain language (actually a set of DSL languages). In the proposed process, domains must be obligatorily specified for each project, but their model can be obtained from a library of models (based on previously developed DSLs). Based on the analysis of the requirements model objects and their attributes will be separated in the domain model.

Elaboration phase – the elaboration phase leads to the development of a verified domain model and its implementation in the form of profiles, supplemented with pictograms of defined stereotypes as well as rules, and restrictions of built DSL languages implemented by profiles. It is necessary to clarify the requirements gained at the inception stage, so it is possible to precisely define rules and restrictions of domain language models stored in UML, OCL, and UAL, which will allow their verification in simulation processes (both metamodels and models). At the elaboration stage, it is also necessary to also prepare the transformation models. The main result of the development stage is therefore a domain model, which will be treated as a conceptual model (metamodel), describing the subject domain, their attributes, relationships and constraints. Based on the metamodel a specialized domain language is being built, consisting of a few (for the needs of SWSA - two) UML profiles. The profile is treated as a set of stereotypes that define the importance of design entities associated with the domain problem.

Construction phase – the construction stage focuses on building models of the system, i.e. the use of verified domain models for storing system models in them. The construction of the system will usually be conducted in a few iterations, separately for each predetermined area of the system, in the corresponding domain language (as many iterations as we have areas of the system to model). Each developed system model is then validated during the model simulation process (including not only the newly defined rules and restrictions of the system model, but predefined rules and restrictions defined for the metamodel). This process ends once the developed models are integrated, eliminating the overlapping solutions stored in different DSLs and then we execute the developed DSL2UML transformations. We end the construction process with a validation of developed UML models (resulting from the transformation), which if necessary can be supplemented by non-modeled elements of the built system domain languages. It is worth noting that this step may be omitted, if for the given DSL there are available transformations, DSL(n)2PSM(i).

Transition phase – this stage leads to the creation of the implemented system as a result of the automatic transformation of models to platform specific models (PSM), then their appropriate packaging and partitioning into elements of the technical structure described by verified topological models.



Phase	Inception		Elaboration			Construction							Transition	
Iteration	1		2			3				4			5	6
Discipline	R	D	R	D	S	R	D	S	V	D	S	V	I	I

Fig. 7. MDmls disciplines during the SWSA development cycle

Summary

Basic elements of the MDmls method intended for designing specialized MLS-type systems are presented (in outline) in the article, in particular, the objectives and scope of the method, its basic processes, domain languages used, stages and method environment are discussed. The description of methods on how to validate the design solutions, specification of threat scenarios and simulation research on models in the RSA environment is beyond the scope of this article and will be the subject of another publication.

Thanks to the integration of security models with the MLS-type SCS models (described in UML), in the proposed method of designing MDmls, the possibility of simulating models is obtained, which allows verification of many security problems of the designed system at the modeling stage. Using the model topology also allows the (physical) study of separating data processing processes belonging to different domains of security, which is one of the elements of verifying MLS-type systems. Additionally, the topology model enables defining the SWSA configuration, and then validate the rules concerning the exhaustion of station physical resources (resulting from its current configuration).

REFERENCES

[1] Anderson J. P.: *Computer Security Technology Planning Study*. Vol. II ESD-TR-73-51. Electronic System Division. Air Force System Command. Hansom Field, Bedford. MA, 01730. 1973

[2] Bell D. E., La Padula L. J.: *Secure Computer System: Unified Exposition and Multics Interpretation*, ESD-TR-75-306, Bedford, MA: ESD/AFSC, Hanscom AFB (http://csrc.nist.gov/publications/history/bell76.pdf).

[3] Bell D. E.: *Looking Back at the Bell-La Padula Model*. Reston VA, 20191, 2005

[4] Biba K.J.: Integrity Consideration for Secure Computer System, MTR-3153, 1975

[5] Clark D., Wilson D.R.: A Comparison of Commercial and Military Computer Security Policies. Proc. IEEE Symposium on Research in Security and Privacy, 1987, 184–194.

[6] Smith R.: Cost profile of a highly assured, secure operating system, ACM Transactions on Information and System Security, Vol. 4, No. 1, 2001, 72–101
[7] Robin, J.S., Irvine, C.E.: Analysis of the Intel Pentium's Ability to Support a Secure Virtual Machine Monitor. In: Proceedings of the 9th USENIX Security Symposium, Denver, Colorado, USA, 2000
[8] Irvine C. E., Levin T. E., Nguyen T. D. and others, Overview of a High Assurance Architecture for Distributed Multilevel Security, Proceedings of the 2004 IEEE Systems, Man and Cybernetics Information Assurance Workshop, West Point, NY, 2004
[9] Methods and Applications of System Virtualization using Intel® Virtualization Technology(Intel® VT), Intel® Technology Journal, Vol. 13, Issue 01, March 2009
[10] Barham P., Dragovic B., Fraser K. and others: Xen and the Art of Virtualization, University of Cambridge Computer Laboratory, CGF Brussels, 2004
[11] Kivity A., Kamay Y., Laor D. and others, KVM: the Linux Virtual Machine Monitor, Proceedings of the Linux Symposium, Ottawa, Ontario, 2007, 225–230
[12] Frankel D. S.: Model Driven Architecture: Applying MDA to Enterprise Computing. John Wiley & Sons, 2003
[13] Dąbrowski W., Stasiak A., Wolski M.: „Modelowanie systemów informatycznych w języku UML 2.1", PWN, Warszawa, 2007
[14] Lodderstedt T., Basin D. A., Doser J.: SecureUML: A UML-based modeling language for model-driven security. In: 5th International Conference 2002, LNCS, vol. 2460, 2002, 426–441
[15] Planning deployment with the topology editor, IBM Tutorial, 2008
[16] Modeling deployment topologies, IBM Tutorial, 2008
[17] Narinder M.: Anatomy of a topology model in Rational Software Architect Version 7.5: Part 1: Deployment modeling, IBM, 2008
[18] Narinder Makin: Anatomy of a topology model used in IBM Rational Software Architect Version 7.5: Part 2: Advanced concepts, IBM, 2008
[19] Willard S.: "General topology", Courier Dover Publications, 2004
[20] Li N., Mitchell J.C.: RT: A Role-Based Trust Management Framework. In: 3rd DARPA Information Survivability Conference and Exposition (DISCEX III), 2003, 201–212
[21] King S.T., Chen P.M., Wang Y., Verbowski C., Wang H.J., Lorch J.R.: SubVirt: Implementing Malware with Virtual Machines. In: IEEE Symp. on Security and Privacy, 2006
[22] Ferrie P.: Attacks on Virtual Machine Emulators. Association of Anti Virus Asia Researchers Conference, Auckland, New Zealand, 2006
[23] Mellor S., Balcer M.: Executable UML: A Foundation for Model-Driven Architecture, wyd. Addison-Wesley Professional, 2002
[24] Fowler, M. i Parsons, R.:„Domain Specific Languages". Addison Wesley, 2010
[25] Fowler, M.: „Patterns of Enterprise Application Architecture". Addison Wesley, 2002
[26] Jürjens J.: "Secure Systems Development with UML", Springer, 2010

***Authors**: dr inż. Zbigniew Zieliński, Military University of Technology, ul. S. Kaliskiego 2, 00-908 Warszawa, E-mail: zzielinski@wat.edu.pl, dr inż. Andrzej Stasiak, Military University of Technology, ul. S. Kaliskiego 2, 00-908 Warszawa, E-mail: stasiak@ita.wat.edu.pl, dr inż. Włodzimierz Dąbrowski, Instytut Sterowania i Elektroniki Przemysłowej, Politechnika Warszawska ul. Koszykowa 75, 00-662 Warszawa, E-mail: w.dabrowski@ee.pw.edu.pl; and Polish-Japanese Institute of Information Technology, ul. Koszykowa 86, 00-662 Warszaw*