Platform-Based Design for Clinical Information Systems

{J. Werner, J. L. Mathe, S. Duncavage} Student Member, IEEE,
B. Malin, Member, IEEE, A. Ledeczi Member, IEEE, J. N. Jirjis, J. Sztipanovits, Fellow, IEEE

Abstract—**Clinical Information Systems (CIS) have emerged as a new critical infrastructure that influence affordability and security of health care delivery. Complex and conflicting societal requirements, such as providing control for patients over their personal health information and requiring health organizations to assure the security and privacy of patient-specific information, create significant technical challenges for the design of CIS. This paper presents a novel approach that is based on the principles and tools of Model Integrated Computing (MIC), Platform-Based Design (PBD) and Service-Oriented Architectures (SOA). We present a domain-specific, graphical design environment and show how formal system specifications can be mapped to different Service-Oriented Architecture execution platforms through a set of standard languages, such as WSBPEL and XACML. The Model-Integrated Clinical Information Systems (MICIS) design environment includes a suite of domain-specific modeling languages capturing essential aspects of CIS design, model transformation tools that map the domain models onto the standard specification languages of SOA platforms and static model analysis tools checking the consistency and wellformedeness of the multiple-view models. The MICIS design tool is tested in modeling the MyHealth@Vanderbilt patient portal of the Vanderbilt University Medical Center.**

I. Introduction

MANY information-intensive industries have developed and deployed standards-based, secure IT infrastructures. In contrast, the healthcare industry remains, for the most part, dependent on paper records and fragmented, error-prone approaches to service delivery. Widely cited reports by the Institute of Medicine and National Research Council have documented weaknesses in information security related to healthcare [1], the costs and impact of medical errors (a substantial proportion of which involve a component of information mismanagement) [2], lack of a systems approach to complex, team-oriented interdisciplinary care [3], and the unrealized potential of using the Internet to improve the quality and availability of healthcare services [4]. There are many different Electronic Medical Record (EMR) systems and a relatively small, but growing, fraction of healthcare organizations have created patient portals that provide secure, personalized customer service via the Internet [20][27][28]. Nonetheless, patient portal technologies are not standardized, and most healthcare organizations rely upon an ad hoc integration of disparate tools and processes. The lack of standardization makes clinical information systems, which are large and decentralized, difficult to manage, secure and evolve.

In this paper, we introduce a formalized design approach to Clinical Information Systems (CIS). Our approach is based on standards-based design methodologies that have been successfully applied in other domains. Service-Oriented Architecture (SOA) [5] provides the framework to assemble loosely coupled software services that can be deployed on different platforms in complex distributed applications. Platform-Based Design (PBD) focuses on the creation of abstraction layers in the design flow and investigates the semantic properties of mappings across these layers [6]. Model Integrated Computing (MIC) is predicated on the notion that models play an essential role in every part of a system's life-cycle, which commences with specification, progresses through design, development, verification, integration, and concludes with upgrade and maintenance [7]. We believe that the combination of SOA, MIC, and PBD techniques can enable the design of complex CIS to ensure reliability, performance, privacy and security beyond what can be achieved by current ad hoc practices. These are prerequisites for the deployment of patient-centered clinical information management systems at a broad range of healthcare providers.

Our selected SOA context is the Web Services Business Process Execution Language, WSBPEL [8]. The standard defines a language for specifying business process behavior based on web services. It is complemented by a suite of coupled standards for web service modeling, web service interoperability, dynamic access control and security policy modeling.

A primary reason we selected SOA as the target platform is the dynamic environment it provides for policy-driven access control, privacy and security implementation. The SOA approach allows (1) the decomposition of business processes into inter- and intra-organizational workflows comprising local groups of services and (2) dynamic and

Manuscript received January 15, 2007. This work was supported in part by the Team for Research in Ubiquitous Secure Technologies (TRUST) NSF S&T Center.

J. Werner, J. L. Mathe, S. Duncavage, A. Ledeczi and J. Sztipanovits are at the Institute for Software Integrated Systems at Vanderbilt University. (phone: 615-343-8307; fax: 615-343-7440; e-mail: akos.ledeczi@ vanderbilt.edu).

B. Malin and J. Jirjis are with the Vanderbilt University Medical Center.

data dependent checking of permissions to execute web services. Related standards, such as XACML and XSD [9][10], provide low-level abstractions that are insufficient to describe the high-level privacy and security objectives of healthcare providers.

PBD is a well established methodology in the embedded systems and software community [6]. By applying this design methodology to CIS, we insert a standard-based SOA abstraction layer in the overall design flow. We selected this abstraction layer because it has several alternative implementations available for experimentation, including one from Oracle [11], IBM [12], and Microsoft [13] each, as well as open source solutions [14]. However, using PBD, we are not restricted to utilize exclusively this abstraction layer (developed for B2B applications) in CIS design. We develop a new layer of abstraction for CIS that is formally captured as a suite of domain specific modeling languages. The unique requirements and operational characteristics of health care delivery provide the underlying basis for these languages. We use the languages to transform ad hoc models of CIS onto a SOA architecture. The abstraction layer is supported by a suite of modeling, model transformation, model analysis and configuration tools that we build using components of the metaprogrammable Model Integrated Computing tool suite [17]. By introducing the most effective domain abstractions, we make models that are concise, understandable, and reusable.

By cooperating with the Vanderbilt University Medical Center, we were able to apply our methods to a real-life example: the MyHealth@Vanderbilt (MHAV) patient portal. The MHAV example was crucial in demonstrating the effectiveness of our design approach.

The rest of the paper is organized as follows. In Section II, we introduce the concept of MIC and the tools that we use for system modeling. This is followed by a description of the structure of the models. Section III describes the chosen methodology and the underlying technologies. Section IV introduces the CIS domain and presents our modeling environment, called Model-Integrated Clinical Information Systems (MICIS) that is used to model and evaluate the MHAV.

II. Model Integrated Computing

MIC employs domain-specific models to represent the system being designed [7]. These models are then used to automatically synthesize the applications and to generate inputs to analysis and simulation tools. This approach speeds up the design cycle, facilitates the evolution of the application, and helps system maintenance, which dramatically reduces costs throughout the lifecycle of the system [7].

The MIC tool suite [25] includes the Generic Modeling Environment (GME), a metaprogrammable tool for domain-specific modeling environments [15]. GME employs metamodels for specifying the abstract syntax and concrete syntax for domain-specific modeling languages [16]. Metamodels account for the concepts, relationships, model structuring constructs and wellformedness rules governing the construction of models. The metamodels implicitly define the family of all well-formed models that can be created using the resultant modeling environment. Using the metaprogramming capability of GME, metamodels are utilized to automatically configure GME for the domain. An interesting aspect of this approach is that GME itself is used to build the metamodels using a formalism based on the UML class diagram notation [18].

GME is used primarily for model-building. The models take the form of graphical, multi-aspect, attributed diagrams. The static semantics (or wellformedness rules) of a model are specified by Object Constraint Language (OCL) constraints [19] that are part of the metamodels. They are enforced by a built-in constraint manager during model building time. The dynamic semantics are applied by the model interpreters, i.e., by the process of translating the models to source code, configuration files, database schema and other artifacts that are called for by the application domain.

III. Methodology

Figure 1 illustrates the hourglass concept of platform-based design [6] as adopted for SOA applications.

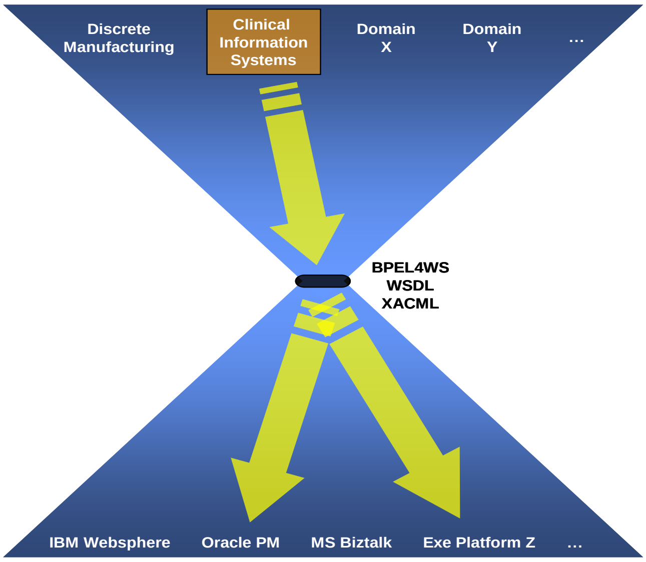


Fig. 1 - The hourglass shape representation of platform-based design for SOA applications.

At the top of the architecture are the different application domains where SOA is applicable. Each domain may use different suits of domain-specific modeling languages, reflecting which are most appropriate for the domain characteristics and domain-specific constraints. By translating the domain models onto BPEL4WS (the narrow waist of the hourglass), using the model transformation tool (GReAT) of the MIC tool suite [26], the underlying alternative implementations of SOA platforms for the BPEL4WS standard become applicable (the lower side of the hourglass). This radically simplifies the fast prototyping, integration and testing tasks.

Figure 2 depicts a more detailed view of the architecture of our design environment. At the heart of our approach, the domain-specific modeling language captures the system from multiple viewpoints.

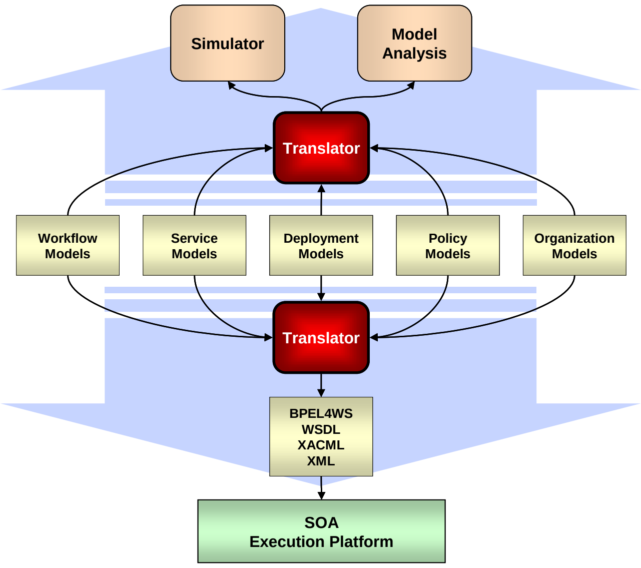


Fig. 2 - Detailed view of the architecture of the design environment.

A. MICIS Model Organization

Service models describe the available services including their interfaces. An important tool for the design and analysis is the characterization of *workflows* that takes place within the system as well as the *policies* that drive their execution. Workflows provide a representation of the manner by which data is accessed, handled, and shared. Failure to formally represent the daily business processes within the healthcare environment, and relate them to each other, makes it challenging to understand why a patient's medical record is accessed, or how interactions between patients and care providers are handled. When workflows are underspecified, or designed in an ad hoc manner, malformed policies can manifest that cause unanticipated consequences. For instance, seemingly routine and innocuous business processes can lead to egregious privacy compromises when taken in combination [21]. Thus, the development of formal workflow models serves as a starting point for developing and analyzing policy-driven operations that support privacy and security.

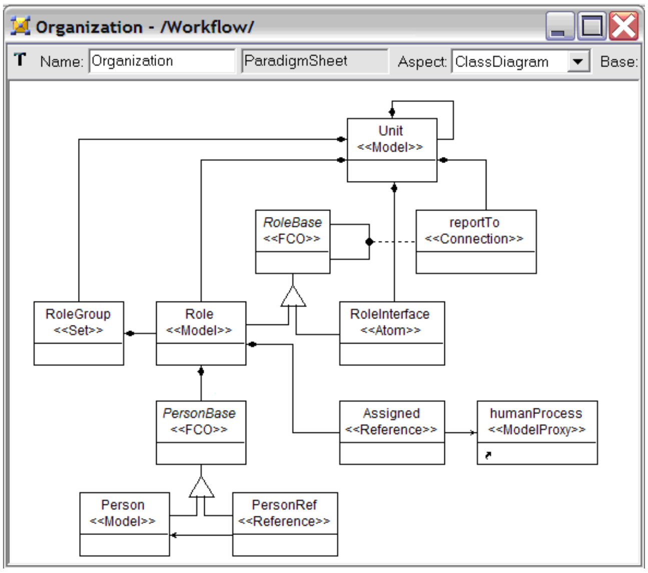
Deployment models capture the computational infrastructure of the enterprise, as well as the assignment of services to resources.

Organizational models are used to specify the architecture of the enterprise itself, such as the roles of different people. For example, these models are referred to by policies to facilitate role-based access control.

B. Language Design

A detailed description of the modeling language is beyond the scope of this paper; however, we present a brief description of the organization modeling. For all other modeling aspects, we only emphasize the different features the language provides.

Organizational models. Figure 3 presents the metamodel, the definition of the domain-specific language, for organizational models. The hierarchical structure of the enterprise is organized around *Units*. Units are defined recursively, such that each Unit can contain other units, in addition to *Roles*. *Persons* are defined inside of Roles, which means that a person can be bound to a given role. Due to the fact that a Person can be assigned multiple Roles, references to Persons, called *PersonRef*, can be defined. *RoleGroups* provide a means to classify different Roles according to multiple criteria. For instance, RoleGroups in a university environment could be drawn from the set {*Administrator*, *Faculty*, *Staff*, *Student*, etc.}. RoleInterfaces capture the interface of a given unit to the enterprise. They are used to represent how Units interact within the enterprise.


Organization - /Workflow/

Name: Organization ParadigmSheet Aspect: ClassDiagram Base:

Fig. 3 - Example metamodel in GME showing the language specifications for modeling organizations.

Finally, the interface to workflow models can be specified with the help of *Assigned* references to

HumanProcesses. The latter are part of workflows; specifically, they represent services that are assigned to humans (as opposed to machines). For example, if the system fails to provide an appointment for a patient, the task will be assigned to a Person in a certain Role.

Deployment models. The primary role of deployment models is to specify the organization of computer servers and their conjunctive networks. Deployment models interface with workflows in a similar manner to organizational models. Services are provided by (software) servers that are deployed on (hardware) servers that are a part of network models.

Service models. They capture the available services with special emphasis on their interfaces. The datatype modeling aspect of the defined interfaces of services allows the language to be strongly typed.

Workflow models. The workflow models can be thought of as a graphical equivalent of a simplified BPEL representation. The key elements of the workflow models are the service invocations that can be asynchronous or synchronous. Synchronization among service invocations can also be specified. The workflow aspect also includes the typical control structures such as switch, join, while, and catch. The inclusion of control structures in the language supports a modeler's definition of arbitrary workflow logic.

Data models. Variables and the implicitly handled messages are modeled as they travel through the workflow, transformed by assign operators, and amended by service invocations via input and/or output variables. Exception conditions are handled with catch operators that can redirect the flow to another part of the workflow or fork to start a new workflow path.

Policy statements are captured as a set of OCL expressions similarly to the approach in [22]. The main reason for this design choice is that GME has built-in native support for OCL in the form of a parser and expression evaluator. We are able to reuse this support for CIS design.

C. Model Testing and Execution

The domain-specific modeling language specifications are captured in the form of metamodels. Then, through an automatic generation process, the domain-specific design environment is synthesized and a new GME "instance" is created to model CIS.

The built-in constraint manager of GME is used for checking the models against constraint violations. While we plan to develop a suite of analysis tools for static model verification, the existing constraint checker already provides a powerful method to force modelers not to violate domain specific design rules.

The bottom half of Figure 2 shows the execution of the system. The domain-specific models, initially stored in GME, are automatically translated to the standard set of SOA languages: WSBPEL, WSDL and XACML. These can be executed on top of any SOA execution platforms.

IV. Example System: Patient Portal

The first step with MICIS is to develop models for MHAV, which is one of the more advanced healthcare sites providing a growing set of individualized services to more than 25,000 enrolled patients [27]. Current services include secure message-based communications between providers and patients that are automatically incorporated into the archival electronic medical record, the ability for patients to request appointments online, and view the current status of billing information. The MHAV site also offers clinical laboratory results to patients and other components of the electronic medical record that have previously been available only to physicians and other healthcare providers.

Our goal is to use MICIS models to analyze information flows and to explore security and privacy properties of the system. Below we describe an example that focuses on a single workflow of the patient portal, where the initiating actor who accesses the portal is the "patient".

Figure 4 presents a basic overview of part of an MHAV workflow model, including input descriptions, transformation rules, output descriptions in the form of flow control models and algorithms in the form of service invocations. The workflow is initiated by an implicitly modeled patient starting at the input: the *accessPortal*. The highlighted instance of the workflow represents one of the patient's selected execution traces in which the patient chooses to view her lab results.

This example represents an asynchronous execution of the following stages of the workflow. First, the patient must authenticate to the system. This step is conducted by the invoked *Portal:login* service. Second, the authenticated client is presented with the portal webpage, which is achieved through the *Portal:populatePage* service.

In the event that the patient authentication fails, the system will generate a fault and prevent the patient from further access to the system. We assume that the input provided by the patient at the *Portal:recUserInput* service is the selection of the *Portal:getLabResults* flow from all possible paths at that point in the workflow. Third, as a result of the patient's selection, the control is transferred to the service represented by the workflow captured inside of *Portal:getLabResults* (depicted in the dark frame on the top of Figure 5). It is important to recognize that the model in Figure 4 captures only the workflow aspect of the system. The data, organization, and deployment aspects are represented by other formal models (not shown).

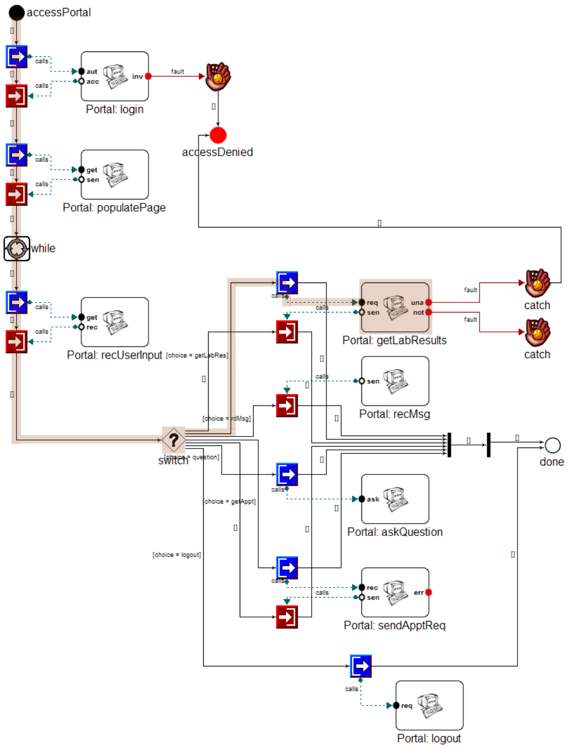


Fig 4. - Workflow aspect of Patient Portal model

Figure 5 presents two models; the 1) data and workflow description of the *Portal:getLabResults* and 2) organization aspect of the system. The top model presents the workflow for the *Portal:getLabResults* service; it contains both data and workflow objects. Providing medical data to patients is one of the most important functions of the patient portal, thus it is modeled with as much detail as possible. These detailed service descriptions allow the modeling of specific items, such as different faults, communicating processes, and forked execution.

This part of the workflow is initiated by the patient portal upon a certain user input: at the *switch* [on Fig.4] the *Portal:getLabResults* path must be selected by the logged in user. This selection initiates the (sub)workflow which as an initial step receives required data: the *patientCredentials* and the *patientIdentifier*. (Data componets are modeled in a different part of the system description not included in this example.) Received data is used in service invocations. For example, *patientCredentials* is passed to *STAR: authorizationService, and patientIdentifier* is passed to *STAR:provideLabResults.*

Invoked services can raise faults depending on the received inputs that can lead to forked execution. Fault handling occurs differently depending on the source: an *unauthorizedAccess* fault arises when the user fails to get authorized access to lab data by the service *STAR:authorizationService*. This fault is handled by denying access, logging its occurrence, and returning to invoking data flow. Fault *notFound* arises when the *Star:provideLabResults* service does not contain information for the given patient and is handled by notifying a database administrator, logging the fault occurrence, and returning to the invoking workflow.

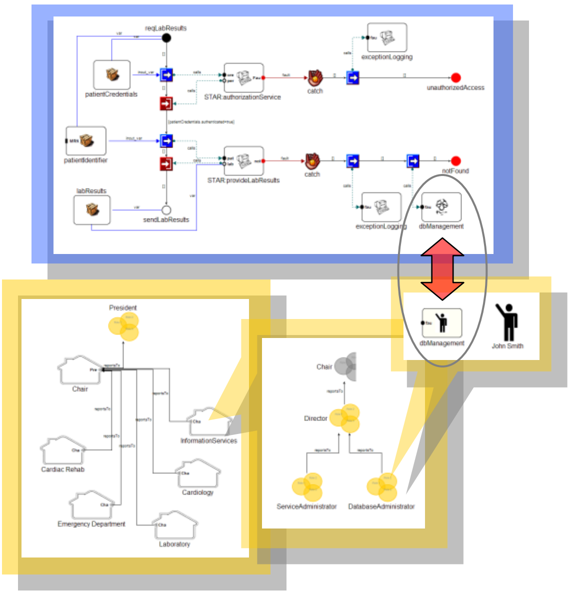


Fig. 5 - Cross-referencing between Organization and Workflow models

The models in the bottom of the figure (lighter frames) represent the organizational aspect of the system. The model on the left describes the hierarchical organization along with the communication paths between the Units (in this domain, the departments). The middle model represents a single Unit of the organization with a hierarchy of Roles. The rightmost model describes a single Role, which binds specific people with specific human services. In the presented example, the *dbManagement* service – invoked in the fault handler – is bound to *John Smith*, who is a *DatabaseAdministrator* in *InformationServices* Unit. Binding between Organization and Workflow models defines the ownership of the services and data provided by this services. Cross referencing data from Organization and Workflow models enable modeler to define permissions and responsibilities using Role-Based Access Control. Verification of the data confidentiality and integrity properties require referencing data from the Workflow and Deployment models. The model of the system is translated to Security Analysis Language [29] designed for verification of security properties of dataflow models. Analysis of our models have discovered multiple vulnerabilities in the MHAV patient portal and we believe that further analysis will be valuable for system developers.

V. Related Research

The MICIS approach is novel in that it provides an integrated and extensible tool suite for clinical information systems. The approach presented in [22] introduces dynamic authorization constraints for XACML using OCL constraints. In contrast, [23] proposes extensions to UML for security modeling. However, this approach is not best suited for the orchestration of web services. The research in [24] proposes a modeling process for medical Web services and presents a solution based on Biztalk [13]. The approach formally describes the problem; however, it is restricted to a single platform. Furthermore, the delivered solution lacks fine-grained security and privacy measures. In contrast, MICIS can be easily extended to include different policy mechanisms. Moreover, our solution introduces abstraction layers enabling platform-independence and providing the flexibility of incorporating additional information in the models.

VI. Conclusions

In this paper, we presented MICIS, a formal approach to the design of Clinical Information Systems that leverages sophisticated design methodologies. In contrast to previous approaches, MICIS integrates three synergistic technologies, Platform-Based Design, Service-Oriented Architectures, and Model Integrated Computing, which have proven track records in various information-dependent domains. Through the integration and application of mature system design tools associated with such technologies, we successfully developed a pilot model for a patient portal, a complex Clinical Information System that provides secure access to an Electronic Medical Records system for patients. The formal basis of our approach allows for the extension, reuse, and evolution of clinical information systems. MICIS evaluates the consequences of system changes without full redesign and implementation. The investigation in this paper addressed only a portion of patient portal functions, but given our initial success, we anticipate that our models will scale to all functions of patient portals and other Clinical Information Systems. Finally, since the healthcare-specific parts of the design environment are relatively small and compartmentalized, we believe that our approach is readily portable to other domains.

References

[1] P. Clayton, Ed, *For the Record: Protecting Electronic Health Information. Committee on Maintaining Privacy and Security in Health Care Applications of the National Information Infrastructure*, Washington, D.C: National Academy Press, 1997.

[2] L.T. Kohn, J.M. Corrigan, and M.S. Donaldson, Eds, *To Err is Human: Building a Safer Health System*, Washington DC,, National Academy Press, 2000.

[3] Committee on Quality of Health Care in America. Institute of Medicine, *Crossing the Quality Chasm: A New Health System for the 21st Century*, Washington, DC: National Academy Press, 2001.

[4] T. Shortliffe, Ed, *Networking Health: Prescriptions for the Internet*.. Washington, DC: National Academy Press, 1997.

[5] Organization for the Advancement of Structured Information Standards (OASIS), http://www.oasis-open.org/home/index.php

[6] A. Sangiovanni-Vincentelli, “Defining platform-based design,” EEDesign Exclusive Feature, March 16, 2002. Available at http://www.eedesign.com/features/exclusive/OEG20020204S0062

[7] G. Karsai, J. Sztipanovits, A. Ledeczi, and T. Bapty, “Model-integrated development of embedded software,” *Proceedings of the IEEE*, vol. 91, no. 1, pp. 145-164, Jan. 2003.

[8] OASIS Web Services Business Process Execution Language http://www.oasis-open.org/committees/tc_home.php?wg_abbrev= wsbpel

[9] XACML 2.0 Specification. http://www.oasis-open.org/committees/ tc_home.php?wg_abbrev=xacml

[10] XML Schema http://www.w3.org/XML/2005/xsd-versioning-use-cases/

[11] ORACLE Process Manager. http://www.oracle.com/technology /products/ias/bpel/index.html

[12] WebSphere Process Server, IBM. http://www-306.ibm.com/software/ integration/wps/

[13] Microsoft BizTalk Server. http://www.microsoft.com/biztalk/ default.mspx

[14] BPEL Engines. http://en.wikipedia.org/wiki/List_of_BPEL_engines

[15] A. Ledeczi, A. Bakay, M. Maroti., P. Volgyesi, G. Nordstrom, J. Sprinkle, and G. Karsai, “Composing domain-specific design environments,” *Computer*, vol. 34, no. 11, pp. 44-51, Nov 2001.

[16] G. Karsai, M. Maroti, A. Ledeczi, J. Gray, and J.Sztipanovits, “Composition and cloning in modeling and meta-modeling,” *IEEE Transactions on Control System Technology*, vol. 12, no. 2, pp. 263-278, March 2004

[17] K. Balasubramanian, A. Gokhale, G. Karsai, J. Sztipanovits, and S. Neema, “Developing applications using model-driven design environments,” *IEEE Computer*, vol. 39, no. 2, pp. 33-40, Feb 2006.

[18] J. Rumbaugh, I. Jacobson, G. Booch. *The Unified Modeling Language Reference Manual*, Addison-Wesley, 1998.

[19] J. B. Warmer and A. G. Kleppe, *The Object Constraint Language: Precise Modeling With UML*, Addison-Wesley, March 1999.

[20] D.B. Baker and D.R. Masys. “PCASSO: a design for secure communication of personal health information via the internet,” *International Journal of Medical Informatics*, vol. 54, no. 2, pp. 97-104, May 1999.

[21] B. Malin and L. Sweeney, “How not to protect genomic data privacy in a distributed network: using trail re-identification to evaluate and design anonymity protection systems,” *Journal of Biomedical Informatics*, vol. 37, no. 3, pp. 179-192, Feb 2004.

[22] M. Alam, R. Breu, M. Hafner, “Modeling permissions in a (U/X)ML world,” in *Proc. First International Conference on Availability, Reliability and Security*, pp. 685-692, April 2006.

[23] J. Juerjens, *Secure Systems Development with UML*, Springer-Verlag, 2004.

[24] R. Anzbock and S. Dustdar, “Modeling and Implementing Medical e-services,” *Lecture Notes in Computer Science*, Springer-Verlag, 2004.

[25] K. Balasubramanian, A. Gokhale, G. Karsai, J. Sztipanovits, and S. Neema, “Developing applications using model-driven design environments,” *IEEE Computer*, vol. 33, no. 2, pp. 33-40, Feb 2006.

[26] G. Karsai, A. Agarwal., F. Shi, and J. Sprinkle, “On the use of graph transformation in the formal specification of model interpreters,” *Journal of Universal Computer Science*, vol. 9, no. 11, pp. 1296-1321, Nov 2003.

[27] https://www.myhealthatvanderbilt.com/

[28] J.J. Cimino, V.L. Patel, and A.W. Kushniruk, “The patient clinical information system (PatCIS): technical solutions for and experience with giving patients access to their electronic medical records,” *International Journal of Medical Informatics*, vol. 68, no. 1-3, pp. 113-127, Dec 2002.

[29] M. Eby, J. Werner, G. Karsai, A. Ledeczi. “Integrating Security Modeling into Embedded System Design.” *International Conference and Workshop on the Engineering of Computer Based Systems*, IEEE, March, 2007