Business Process Management with Unified Modeling Language

CĂLIN-ADRIAN COMES
Petru Maior University
Department of Financial-Accounting
Nicolae Iorga 1, 540088, Târgu-Mureş
ROMANIA

NICOLAE GHIŞOIU
Babeş-Bolyai University
Department of IT Applied to Economics
Mihail Kogălniceanu, 400084, Cluj-Napoca
ROMANIA

Abstract: The evolution of Enterprise Information Systems(EIS) has been shift from data-centric information systems to process-centric information systems in last decade. Business Process Management(BPM), extension of Workflow Management(WFM) contains methods, techniques and tools to support the design, enactment, management and analysis of business process. Petri Nets(PN) are used like standard for BPM and WFM technology. Our approach is to define a methodology for analyze, design, implement, maintain EIS only with Computer Aided Software Engineering(CASE) tools, according to Model Driven Architecture(MDA) from Platform Independent Models(PIMs) to Platform Specific Models(PSMs). In our approaches we proposed an extension of Object Constraint Language(OCL), Complete Constraint Language(CCL) in idea to use the Activity Diagram(AD) from Unified Modeling Language(UML) as formal basis for BPM and WFM technology, respectively Entity Relationship-Stored Procedures(ER-SP), an extension of ER(Entity Relationship) model for conceptual, syntactic and semantic modeling of relational databases in idea to create a link between external to physical models.

Key–Words: Enterprise Information Systems, Business Process Management, Workflow Management, Petri Nets, Computer Aided Software Engineering, Model Driven Architecture, Object Constraint Language, Complete Constraint Language, Activity Diagram, Unified Modeling Language, Entity Relationship-Stored Procedure

1 Introduction

The EIS application helps companies to manage the diversity of national and international trade. With over three decades of industry experience, EIS enables any kind of organization to streamline and automate the import and export activities according the regulatory compliances, mitigating the financial risk regarding global transactions, giving a holistically approach to manage global activities over heterogeneous landscapes.

OCL was formally developed as a business language with roots in Syntropy, second-generation object-oriented analysis and software design method developed at Object Designers Limited in the UK during the early 1990s, used to describe UML models, mapping only Class Diagrams and Object Diagrams.

In this paper we try to extend the Constraint Language syntax to the rest of UML diagrams: Use-Case Diagrams, Sequence Diagrams, Collaboration Diagrams, State Transition Diagrams and Activity Diagrams in idea to manage the Business Process with this OCL extension.

2 State-of-the-Art

Petri Nets flavors [1, 2, 3, 5, 6, 10, 12, 13, 14, 15, 16, 17, 18] are used for modeling and analysis of Workflow Management and Business Process Management in the workflow engines of SAP[1] and Oracle.[2] Oracle and SAP are using Business Process Execution Language(BPEL) integrated in their Application Servers.

3 Related Research

Business Process Management and Semantic Interoperability Research Program with their specific projects ATHENA and DIP are main direction at SAP Research. With ATHENA (**A**dvanced **T**echnologies for Interoperability of **H**eterogeneous **E**nterprise **N**etworks and their **A**pplications) the SAP Research team has already developed a prototype for modeling and executing cross-organizational business processes, a draft ontology defining core business concepts, and a prototype of an adaptive and extensible infrastructure for Web-services connectivity to the project. In Figure 1 is illustrate the Model to Model

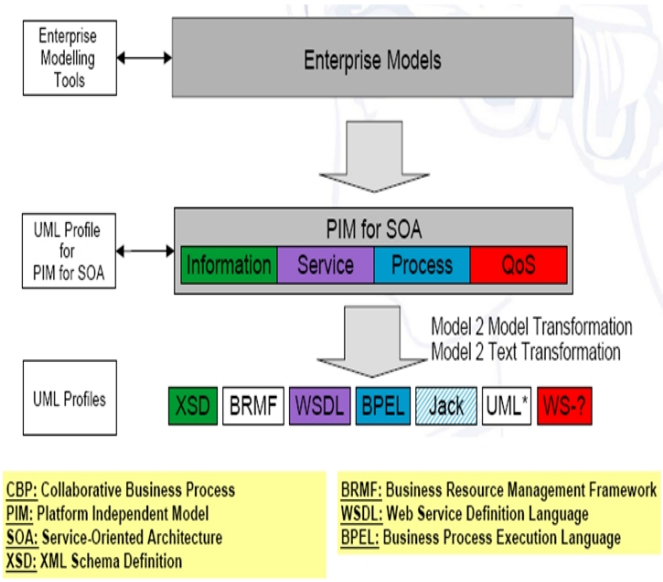

Figure 1: ATHENA Model-driven and Adaptable Interoperability Framework and Infrastructure, source [4].

Transformation and the Model to Text Transformation [4].
ATHENA builds upon the following vision statement:

"By 2010, enterprises will be able to seamlessly interoperate with others."

Data, **I**nformation, and **P**rocess Integration with Semantic Web Services (DIP) has the goal to provide a common framework for sharing and using data across applications, enterprises, and community boundaries.

4 Research Questions

We elaborate and describe two research questions:

1. (RQ1) The proposal **Complete Constraint Language** and UML contains enough methods, techniques and tools to support the design, enactment, management and analysis of Business Process according to Model Driven Architecture(MDA) from Platform Independent Models(PIMs) to Platform Specific Models(PSMs)?
2. (RQ2) The proposal **Entity Relationship-Stored Procedures(ER-SP)**, an extension of ER(Entity Relationship) model for conceptual, syntactic and semantic modeling of relational databases could create a link between conceptual to physical models?

With (RQ1) we try to extend the ATHENA project with Complete Constraint Language in idea to transform the Platform Independent Model level into Platform Specific Model e.g. Java EE from SUN[3] or .NET Framework from Microsoft[4].
With (RQ2) we try to extend the DIP project with Entity Relationship-Stored Procedures(ER-SP) like Platform Independent Model for Data into Platform Specific Model RDBMS.

5 Motivation Scenario

We know that only from Class Diagram it is possible to generate skeleton code for applications programming, but the real blue print for a project is entire with constraints from all UML diagrams type. My proposal research extends the description language OCL [21] with CCL for all UML diagrams with the purpose of giving permission for checking the syntax and the semantics for all of them. The UML diagrams must be in relationship giving together the blueprint for OOA/OOD models with collaboration and interaction [22]. The ATHENA[4] project manage the transformation between PIM for SOA(Information, Service, Process, and QoS) and XML flavors(XSD, BRM, WSDL, BPEL, et.c.). Our approach is to manage the transformation between PIM(UML with OCL and CCL and ER-SPL) and PSM (Java EE or .NET Framework)

6 Use-Case Constraint Language syntax

We formally define the abstract syntax of Use-Case Constraint Language with the following components: a set of actors, a set of use-cases, a set of associations with role names and multiplicities, a generalization hierarchy over actors and use-cases.

6.1 Actors

Definition 1(ACTOR) The set of actors is a finite set of names:

$$ACTOR \subseteq N \tag{1}$$

Each *actor* $\in$ *ACTOR* induces *actor type's* $t_{actor} \in T_{ACTOR}$. T_{ACTOR} is a set of type names. All actors have a distinct name; in particular, an actor name may not be used again to define another actor with a

[3]Sun Microsystems, Inc. All Rights Reserved
[4]Microsoft Corp. All Rights Reserved

different type.

$\forall actor_1, actor_2 \in T_{ACTOR}$:$(actor : t_{actor} \rightarrow actor_1, actor : t_{actor} \rightarrow actor_2) \Longrightarrow actor_1 = actor_2$

Actors with the same name may, however, appear in different use-cases that are not related by generalization.

6.2 Use-cases

Definition 2(USE-CASE) The set of use-cases is a finite set of names:

$$\textit{USE-CASE} \subseteq N \tag{2}$$

Each *use-case* $\in$ *USE-CASE* induces *use-case type's* $t_{use-case} \in T_{USE-CASE}$. $T_{USE-CASE}$ is a set of type names. All use-cases have a distinct name; in particular, a use case may not be used again to define another use-case with a different type.

$\forall use-case_1, use-case_2 \in T_{USE-CASE}$:$(use-case : t_{use-case} \rightarrow use-case_1, use-case : t_{use-case} \rightarrow use-case_2) \Longrightarrow use-case_1 = use-case_2$

6.3 Associations

Associations for actors and use-cases describe the relationship between actors and use-cases. Generally, actors and use-cases may participate in any number of associations, and associations may connect two or more actors and/or use-cases.

Definition 3(ACTOR ASSOCIATIONS) The set of actors associations is a finite set of names:

$$\textit{ACTOR\_ASSOC} \subseteq N \tag{3}$$

$$assoc : \begin{cases} ACTOR\_ASSOC \rightarrow ACTOR^+ \\ as \mapsto \langle actor_1, \ldots, actor_n \rangle, n \geq 2 \end{cases}$$

Definition 4(USE-CASE ASSOCIATIONS) The set of use-case associations is a finite set of names:

$$\textit{USE-CASE\_ASSOC} \subseteq N \tag{4}$$

$$assoc : \begin{cases} USE-CASE\_ASSOC \rightarrow USE-CASE^+ \\ as \mapsto \langle use-case_1, \ldots, use-case_n \rangle, n \geq 2 \end{cases}$$

6.4 Role names

Actors and use-case appear more than once in a associations, each time playing a different role. A role name of a actor and use-case are used to determine the navigation.

Definition 5 (ACTOR ROLE NAMES)Let actor_assoc $\in$ ACTOR_ASSOC be an actor association *assoc(as)*=$\langle actor_1, \ldots, actor_n \rangle$. Role is defined by function:

$$roles : \begin{cases} ACTOR\_ASSOC \rightarrow N^+ \\ actor\_assoc \mapsto \langle r_1, \ldots, r_n \rangle, n \geq 2, \\ \forall i, j \in \overline{1,n} : i \neq j \Rightarrow r_i \neq r_j. \end{cases} \tag{5}$$

Definition 6 (USE-CASE ROLE NAMES)Let use-case_assoc $\in$ USE-CASE_ASSOC be an use-case association *assoc(as)*=$\langle use-case_1, \ldots, use-case_n \rangle$. Role is defined by function:

$$roles : \begin{cases} USE-CASE\_ASSOC \rightarrow N^+ \\ use-case\_assoc \mapsto \langle r_1, \ldots, r_n \rangle, n \geq 2, \\ \forall i, j \in \overline{1,n} : i \neq j \Rightarrow r_i \neq r_j. \end{cases} \tag{6}$$

6.5 Generalization

A generalization is a taxonomic relationship between two actors/use-cases. This relationship specializes a general actor/use-case into more specific actor/use-cases.

Definition 7(GENERALIZATION HIERARCHY FOR ACTORS) A generalization hierarchy for actors $\prec$ is a partial order on the set of *actors* ACTOR.

(7)

Pairs in $\prec$ describe generalization relationship between two actors. For actors $actor_1$, $actor_2 \in$ ACTOR with $actor_1 \prec actor_2$ the actor $actor_1$ is **child** actor and $actor_2$ is a **parent** actor.

Definition 8 (GENERALIZATION HIERARCHY FOR USE-CASES) A generalization hierarchy for use-cases $\prec$ is a partial order on the set of *use-cases* USECASE. Pairs in $\prec$ describe generalization relationship between two use-cases. For use-cases $use-cases_1$, $use-cases_2 \in$ USE-CASE with $use-cases_1 \prec use-cases_2$ the usecases $use-cases_1$ is **child** use-cases and $use-cases_2$ is a **parent** use-cases.

(8)

Definition 9(ABSTRACT SYNTAX FOR USE-CASE CONSTRAINT LANGUAGE) The syntax for use-case model is a structure defined by:

$$M_1 = (ACTOR, USE-CASE, assoc, roles, m, \prec) \tag{9}$$

Where components are defined in (1), (2), (3), (4), (5), (6), (7), (8) and m represents the multiplicity.

7 Abstract Syntax for Sequence Constraints Language

Definition 10(ABSTRACT SYNTAX FOR SEQUENCE CONSTRAINT LANGUAGE) The Use-Case Diagram could generate the Sequence Diagram; the difference is that in Sequence Diagram the role names are indexed. The index is the "red line" for flow messages. Use-case could be defined by word **what**, respective Sequence Diagram by word **how**.

$$M_2 = (ACTOR, CLASS, assoc, roles, m, index) \tag{10}$$

where m represents the multiplicity.

8 Abstract Syntax for State Constraints Language

Definition 11(ABSTRACT SYNTAX FOR SEQUENCE CONSTRAINT LANGUAGE) The syntax for state model is a structure defined by:

$$M_3 = (i, STATE, SUBSYS, DECISION, transit, f) \tag{11}$$

9 Abstract Syntax for Collaboration Constraints Language

Definition 12(ABSTRACT SYNTAX FOR COLLABORATION CONSTRAINT LANGUAGE) The syntax for collaboration model is a structure similar to Sequence Constraint Language, but points the dynamic collaboration between components defined by:

$$M_4 = (ACTOR, CLASS, assoc, roles, index) \tag{12}$$

10 Abstract Syntax for Activity Constraints Language

Definition 13(ABSTRACT SYNTAX FOR ACTIVITY CONSTRAINT LANGUAGE) Activity Diagrams are an amalgamation of number of techniques: state modeling, work flow modeling and petri-nets, defined by:

$$M_5 = (i, STATE, SUBSYS, DECISION, transit, f) \tag{13}$$

11 Entity Relationship-Stored Procedure

ER-SP (Entity Relationship - Stored Procedures) model try to be an extension to ER (Entity Relationship) model for conceptual, syntactic and semantic modeling of RDBMS in idea to create a link between external models to physical models for RDBMS. ER-SPL (Entity Relationship - Stored Procedures Language) tries to be independent by RDBMS vendors in idea to support their own SQL(DDL, DML, DCL) dialects. The model proposes a new entity type called Stored Procedures type and the language behind him in idea to be platform independent in relation with SQL dialects and their Procedural Languages. The concept of stored procedures type is abstract, helps in syntactic and semantic modeling, and is required in physical implementation. A stored procedures entity type along with his language captures the syntactic and semantics of an RDBMS schema in his dynamical evolution.

The ER[8] Model serves as the foundation of many systems analysis and design methodologies, repository systems, and CASE tools from commercial software vendors to free source world. Any important article, book from respectable publishing houses where the modeling word is a key word the ER Model or ER Model flavors are just in place.

During the time because there is no standard for ER Model there are a lot confusions regarding the definition of the entity.

Bernhard Thalheim in[23] cite more than twelve different approach for defining the entity notion with the following remark "The confusion is almost since most of the database and software engineering books do not define at all the notion of entity "

11.1 ER Model

ER Model has been redefined because of his popularity with ER Metamodels like EER (Extended Entity Relationship) [11], CSL (Conceptual Schema

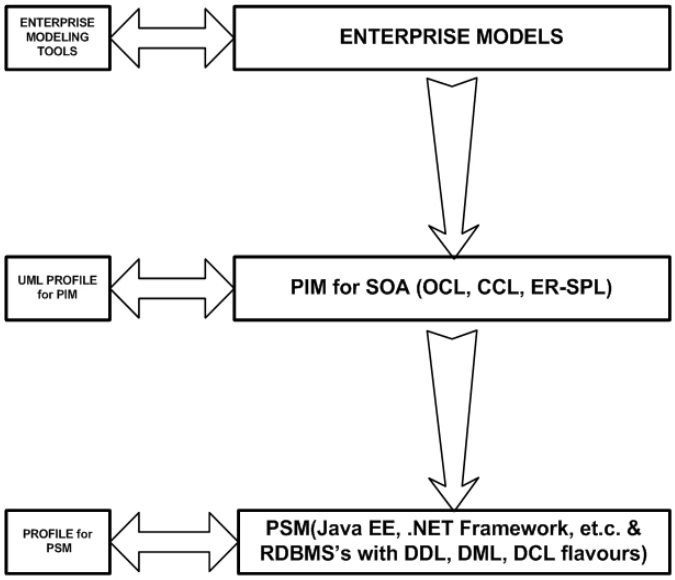

Figure 2: A PIM to PSM Interoperability Framework with Complete Constraint Language.

Language)[7], HERM (Higher-Order Entity Relationship Model)[24] [4], and REMORA [9].

12 Contributions

Our approaches underline the importance of Complete Constraint Language in the transformation of Business Process according to Model Driven Architecture(MDA) from Platform Independent Models(PIMs) to Platform Specific Models(PSMs).

13 Conclusions

In this paper we propose the abstract syntax for Complete Constraint Language, platform independent language for UML 2.0. Because most of business application development process do not respect all milestones for their methodologies when some upgrades are made, the team involved is focused **only on coding**, reanalyzing and redesign is neglected. Complete Constraint Language for UML 2.0 diagrams has been proposed to fill this gap.

14 Future work

In the future research we study the concrete syntax and semantic for Complete Constraint Language with support for validation and we proposed the Data Constraint Language (DCL), an extension description language for ER(Entity Relationship) Model on conceptual level respective ER-SP(Entity Relationship - Stored Procedure) for Database Diagrams on physical level.

Acknowledgments: The author would like to acknowledge the financial support by "Petru Maior" University of Târgu-Mureş, through the Doctoral Grant no. 1783/2006.

References:

[1] Wil M.P. van der Aalst, Business Process Management Demystified: A Tutorial on Models, Systems and Standards for Workflow Management, ACPN 2003, LNCS 3098, Springer Verlag GmbH, pages 1-65, 2004

[2] Wil M.P. van der Aalst and K.M. van Hee, Workflow Management: Models, Methods and Systems, MIT press, Cambridge, MA, 2002

[3] Wil M.P. van der Aalst, The Application on Petri Nets to Workflow Management, The Journal of Circuits, Systems and Computers, 8(1), pages:21-66, 1988

[4] ATHENA Consortium, ATHENA Results Overview, October 2005,

[5] K. Barkaoui, J.M. Couvreur, and C. Dutheillet, Application and Theory on Petri Nets 1995, LNCS 935, Springer Verlag GmbH, pages:25-44, 1995

[6] J. Billington, Application and Theory on Petri Nets 2003, LNCS 2679, Springer Verlag GmbH, pages:483-506, 2003

[7] B. Breutmann, E. F. Falkenberg, and R. Mauer, CSL: A Language for Defining Conceptual Schema, Amsterdam, North Holland, 1979

[8] P. Pin-Shan Chen, The Entity-Relationship Model: Toward a Unified View of Data, ACM Trans. Database Systems, Volume 1, Number 1,1976, pp 9-36, ACM Press, New York, NY, USA

[9] R., Colette, An Information System Methodology supported by an Expert Design Tool, Elsevier Science Publishers,1988

[10] J. Desel, and J. Esparza, Free Choice Petri Nets, volume 40 of Cambridge Tracts in Theoretical Computer Science, Cambridge University Press, Cambridge, UK, 1955

[11] David W. Embley and Tok Wang Ling, Synergistic Database Design with an Extended Entity-Relationship Model, Proceedings of the Eight

International Conference on Entity-Relationship Approach to Database Design and Querying, North-Holland, pages:111-128, 1990

[12] R. Eshuis, and J. Dehnert, Reactive Petri Nets for Workflow Modeling, Application and Theory on Petri Nets 2003, LNCS 2679, Springer Verlag GmbH, pages:295-314, 2003

[13] J. Esparza, Synthesis rules for Petri Nets and how they can lead to new results, Proceedings of CONCUR 1990, LNCS 458, Springer Verlag GmbH, pages:182-198, 1990

[14] J. Esparza and M. Silva, Circuits, Handles, Bridges, and Nets, Advances in Petri Nets 1990, LNCS 483 Springer Verlag GmbH, pages:210-242, 1990

[15] K. van Hee, N Sidorova and M. Voohoeve, Soundness and Separability of Workflow Nets in the Stepwise Refinement Approach, Application and Theory on Petri Nets 2003, LNCS 2679 Springer Verlag GmbH, pages:335-354, 2003

[16] A. W. Holt, Coordination Technology and Petry Nets, Advances in Petri Nets 1985, LNCS 222, Springer Verlag GmbH, pages:278-296, 1985

[17] K. Jensen, Colored Petri Nets, Advances in Petri Nets 1990, LNCS 483, Springer Verlag GmbH, pages:342-416, 1990

[18] K. Jensen, Colored Petri Nets. Basic Concepts, EACTS monographs on Theoretical Computer Science, Springer Verlag GmbH, 1997

[19] OMG, Object Management Group, Unified Modeling Language (UML) Specification: Infrastructure, version 2.0 (ptc/03-09-15)

[20] OMG, Object Management Group, Unified Modeling Language (UML) Specification: Superstructure, version 2.0, Revised Final Adopted Specification (ptc/04-10-02)

[21] OMG, Object Management Group, Object Constraint Language (OCL), OMG Final Adopted Specification (ptc/03-10-04)

[22] James Rumbaugh, Ivar Jacobson, Grady Booch, The Unified Modeling Language Reference Manual, Addison-Wesley (1999)

[23] B. Thalheim, Conceptual Modeling: Current Issues and Future Directions, The Strength of ER Modeling, LNCS 1565, Springer Verlag GmbH, pages: 227-242, 1999

[24] B. Thalheim, Herm: Putting theory into practice. IDS-92, IFIP-Workshop on Data-base Intellectualization, Control Systems and Machines, 1992, 5/6, pp 85-93, Kaliningrad