Available online at www.sciencedirect.com

ScienceDirect

Computer Standards & Interfaces 30 (2008) 315 – 322

COMPUTER STANDARDS & INTERFACES

www.elsevier.com/locate/csi

Advanced steps with standardized languages in the re-engineering process

Anna Medve

University of Pannonia (before named University of Veszprém), Faculty of Information Technology, Department of Information Systems, Veszprém, Hungary

Received 12 December 2006; received in revised form 20 August 2007; accepted 25 September 2007
Available online 12 October 2007

Abstract

Formal methods can be used at all stages of a software development project. In this paper we reflect on the roles of ITU-T standardized System Design Languages and their interplay in design processes of cooperative systems, highlighting the usability of User Requirements Notation (URN) standard to capture requirements with workflow-based re-engineering process of complex systems. In this paper we give our re-engineering experiences with the URN-part Use Case Maps (UCM) language capabilities and also the transformation processes of the UCM model elements to UML diagrams are presented. The new components can be very well documented and integrated into the existing system in a manner that even the stakeholders get involved in it.

Keywords: Re-engineering; URN; UCM; UML-2.0; ITU-T System Design Languages

1. Introduction and motivation

In a previous article of the journal Computer Standards&Interfaces 27(3)(2005), Adamis et al. [1] have already presented a survey of the most important standardized languages and notations that belong to Z series [2] of languages of the International Telecommunication Union. The Z series also contains several standards [4–12], which are commonly named as ITU-T System Design Languages. They are (please follow the list of abbreviations) ASN.1, DCL, MSC, ODL, SDL, TTCN, and URN composed by UCM and GRL, standards generally used in a system modeling and development processes of telecommunication systems. Adamis et al. [1] in this issue discuss the goals, the most important features and the basic syntactic and semantic rules of the languages and notations. Showing the purposes and the roles of these formal languages in the development process of telecommunication systems, the authors point out the specific features of the telecommunication systems that can be expressed by these different languages.

In this paper we supplement the presentation of the ITU-T System Design Languages familiarized in a previous article of the Computer Standards&Intefaces 27(3)(2005) publication. We present in Section 2 the scalability of roles of languages in their interplay in the design processes of cooperative systems, highlighting the usability of User Requirements Notation standard to capture requirements of complex systems with maps of workflow-based re-engineering processes.

Given in Fig. 1 the ITU-T multi-languages software process includes different tool-dependent and/or manually adapted language-mappings to serve as inputs for the combination of language pairs in test or deployment tasks. Several combination processes are given with transformations manually or automatically between languages on a basis of standard supplements that support transformations.

The ITU-T System Design Languages become in a class by itself to support the MDA/UML standard [13]. The Superstructure, that is the part of the language visible by the modeler, comprise some well-known and established ITU-T formal

Abbreviations: ASN.1, Abstract Syntax Notation One [3]; DCL, Deployment Constraint Languages [12]; FMT, Formal Method Technique; GRL, Goal-oriented Requirement Language [4]; ITU-T, International Telecommunication Union [2]; IDL, Interface Definition Language; MDA, Model Driven Architecture [13]; MSC, Message Sequence Charts [7]; OMG, Object-oriented Management Group [13]; SDL, Specification and Description Language [8]; TTCN-3, Testing and Test Control Notation [11]; UCM, Use Case Maps [6]; UML, Unified Modeling Language [13]; URN, User Requirement Notation [5]; ODL, Object definition languages [9]; SAP, Systems, Applications and Products in data processing.

E-mail address: medve@almos.vein.hu.

0920-5489/$ - see front matter © 2007 Elsevier B.V. All rights reserved.
doi:10.1016/j.csi.2007.09.004

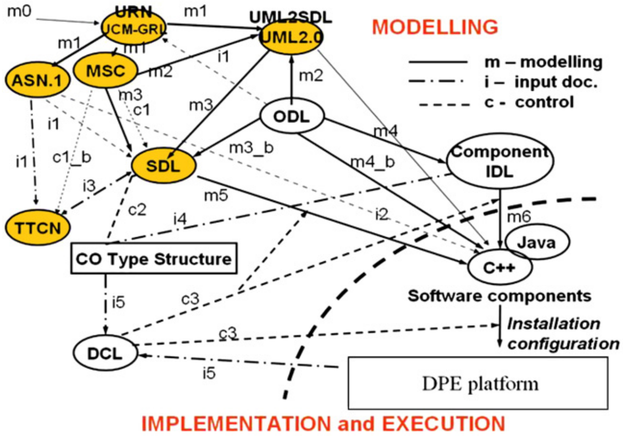

Fig. 1. The ITU-T System Design Languages roles in their interplay.

methods features like Message Sequence Charts (MSC) [7], Specification and Description Language (SDL) [8], Real-Time Object-Oriented Modeling (ROOM) [14] and others. Our investigation on ITU-T System Design Languages relates that some concepts and methods of research in communicating systems engineering are adaptable while introducing the IT supports within socio-technical context rules in the organizations [15].

Experiments to model business processes with these languages are lesser-known and its rely to use of User requirements Notations (URN) [5] and MSC languages in e-commerce by Weiss et al. in [16] presenting the URN capability to satisfying the goals of a Business Process Management (BPM) language [17]. We use the architectural viewpoints of *Use Case Maps (UCMs)* [6] the URN-part language for scenarios modeling according to Buhr in [18], and according to Amyot and Musbach in [19] the current UML features are presented as pieces of puzzle in the UML development process and UCM is presented as the missing piece. Gordijn in [20] relates on use URN to develop system architectures supporting the business value and process models. Medve in [21] presents the joint use of URN with UML contributing to express enterprise models that can co-evolve with enterprise engineering processes.

In Section 3 we present our results on the usability of User Requirements Notation (URN) standard to capture requirements for socio-technical systems by modeling with UCM the processes of workflow based on software's system. Our experiment focuses on the adaptability of URN to capture users' needs and to derive UML2.0 features to express software components related to the evolution of the ERP systems. Our experience in this area is in the real case study in order to document the evolution of a legacy system with UCM and UML [22].

Section 4 explains the transformation process of the UCM model elements to UML diagrams. Finally, a discussion on benefits of our experience and future plans is present.

2. ITU-T System Design Languages interplay in the design processes

The ITU-T System Design Languages are continuously developed in Z series [2] of ITU-T Recommendations. These languages are standardized formal and semiformal methods to support the entire cycle of software development and standards specification in telecommunication.

The leader role of the SDL (Specification and Description Language) [8] and UML2.0 (Unified Modeling Language) [13] cohesively generates the executable code. It is widely used and supported by commercial tools including automatic code generation, simulation and validation, it contain statements to model both the architectural and behavioral aspects of systems.

The URN (User Requirements Notations) is a first standardization effort for user requirements engineering. URN combines two complementary notations: the GRL (Goal-oriented Requirement Language) [4] as Non-Functional Requirement framework to capture business or system goals, alternative means of achieving goals and rationales for contributions and decisions and UCM (Use Case Maps) [6] as Functional Requirement framework for scenarios modeling. The URN is applicable within standards bodies, industry, and commercial organizations.

The MSC (Message Sequence Charts) [7] is very suitable for modeling temporal behavior of the system by describing sequences of messages that are exchanged between system components in order to achieve system functionalities, playing an important part in modeling and controlling of SDL models.

ASN.1 (Abstract Syntax Notation One) [3] defines sets of rules, which are suitable for converting any type of data into a bit stream suitable for transmission. The ASN.1 translations to IDL are the most recent results of ASN.1 Consortium for formal description of data types such as ASN.1.

The ITU-ODL () [9] offers necessary key concepts to design distributed telecommunication applications as Computational Objects (COs) communicating via multiple well defined interfaces. ODL is redefined in eODL (Extended Object Definition Languages) [10].

TTCN (Testing and Test Control Notation) [11] is designed for functional black box testing and to describe Abstract Test Suites (ATS) which are independent of a concrete test platform. Test software can be modeled and developed in exactly the same way as the functional system software.

DCL (Deployment Constraint Languages) [12] serves to describe the Distributed Processing Environment (DPE) Architecture for installing and configuring software components in environments of applications in telecommunication and information services.

In order to apply different techniques with their specific advantages in a combined approach for the design of distributed systems, the underlying concepts have to be integrated within a common concept space. The multi-language space of ITU-T System Design Languages have concepts on interrelations of processing the distributed systems by dividing software processes within co-operative autonomous languages to realize a controllable software development.

Given in Fig. 1 the ITU-T multi-languages software process includes different tool-dependent and/or manually adapted language-mappings (m0-m6) to serve as inputs (i1-i5) for the combination of language pairs of the Validation&Verification tasks (c1, c1_b) or of the deployment and configuration tasks (c2, c3) before and after code generation. It operates in the two main parts of the languages space: the Modeling part and the Implementation and Execution part of languages space.

There are several ways to combine languages to improve the quality of requirements engineering and in order to obtain the system architecture, as we can see in Fig. 1. The UML1.x model of system could be converted into SDL specification [23]. Data given in ASN.1 could be incorporated into SDL and TTNC specifications. We can check the SDL specification with MSC sequence diagrams. We can use SDL specification to generate the test cases for TTCN testing. We can apply UCMs and MSC and GRL to evaluate requirements. UCMs can be applied such as input in transformation to UML activity diagrams and sequence diagrams, as we present in Section 4. The UML 2.0 version integrates completive the MSC language for dynamic analysis purposes and SDL language for state machine implementation.

Several tools and methods that we can find at the SDL Forum Society [24] and at the ITU-T [2] are issued to support the joint use of combinations of ITU-T System Design Languages. The combinations of languages exploit their modeling or controlling force in the software processes and in the automation code generation. In the rest of the paper we present URN details on their advantages and the UCM-based re-engineering case. The introduction of the computer may change the organization's processes, and, as a result, the majority of re-engineering cases occur within the emergent and the consequential requirements, and open up new ways of working which generate new system requirements.

3. The UCM capabilities and the re-engineering case

UCM notations are very useful for descriptions of service functionality in requirements elicitation phase, where usually requirements suffer from heavy instabilities, whereas scenarios and potential component topologies (structures of functional and network entities) are volatile. UCMs fit well in approaches that intend to bridge the gap between requirements and an abstract system design, where a tentative distribution of system behavior over a structure is being introduced. Thus, they represent useful and powerful tools for the support of the thinking process and the evaluation of functional alternatives. It is essential and that UCM capabilities enable early reasoning on the basis of UCM model, about developing and implementing the components to the changed system's processes.

In re-engineering, [35] we change the design and implementation, perhaps to satisfy new requirements or perhaps to improve a system, say, to respond a customer feedback about inadequate performance. Once end-users have experience of a system, new requirements emerge for the different reasons of business priorities that may change with consequent changes of IT supports.

Our research results in [15,21], show the usefulness of using UCM maps to express the changed requirements. We deduct the needed system requirements by expressing the actual workflow processes of the systems. Consequently, we can involve the end-user in UCM specification to discover the changed working processes. Thus in bask-end an architect expresses system requirements specification and makes performance analysis [25]. We apply Buhr method, who in [26] suggests the use of UCMs to express architecture that should not be violated while re-engineering actual systems at the detailed level, i.e. the enduring requirements. If the original high-level design was well conceived, re-engineering may require only modifying the detailed level. The Buhr's methods being important in actually needed Model Driven Architecture (MDA) software processes steps. Amyot in [27] presents some combination of forward engineering and reverse engineering may be used to try to achieve a satisfactory re-engineering process.

Use Case Maps as part of the URN standard [5] to specify functional requirements. Use Case Maps (UCM) is the language used for the URN-FR is the notation [6]. UCM specifications serve as a technology-independent bridge between requirements and design when *combining behavior and structure in one view*, providing a path-centric view of the system. It supports to express all traditional objectives of information system for enterprise modeling and understanding as socio-technical systems with abstract components and constructs.

UCMs show related and interacting use-cases in *a map-like diagram* (see Fig. 2). *Paths* are wiggle lines that in an UCM diagram show the chain of events of a scenario. Scenarios may interact, for example, by one path triggering or disabling another.

Basic notations of UCMs (see Fig. 2) start points, responsibilities, end points and components. *Start points* are filled circles representing pre-conditions or triggering causes. *End points* are bars representing post-conditions or resulting effects. *Responsibilities* are crosses representing actions, tasks or functions to be performed. *Components* are boxes that represent, at the requirements level, abstract entities corresponding to actors, processes, objects, containers, agents, etc. (See also Fig. 3).

A lengthy path specification can be fitted into a root map by moving segments of it to child maps, using the *stub and plug-in* mechanism. The diamond symbol indicates a *stub* notion. Multiple levels of stubs and plug-ins can be used, structuring and integrating scenarios in various ways. Integration may

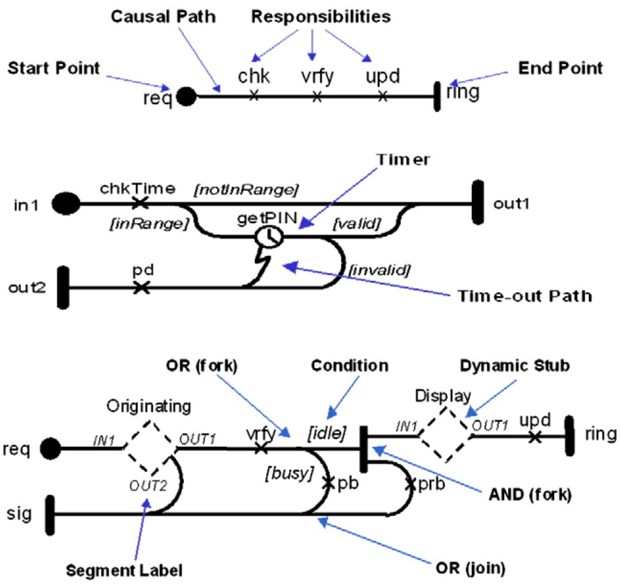

Fig. 2. Basic notations of an UCM path, stub and plug-in mechanisms, and control elements.

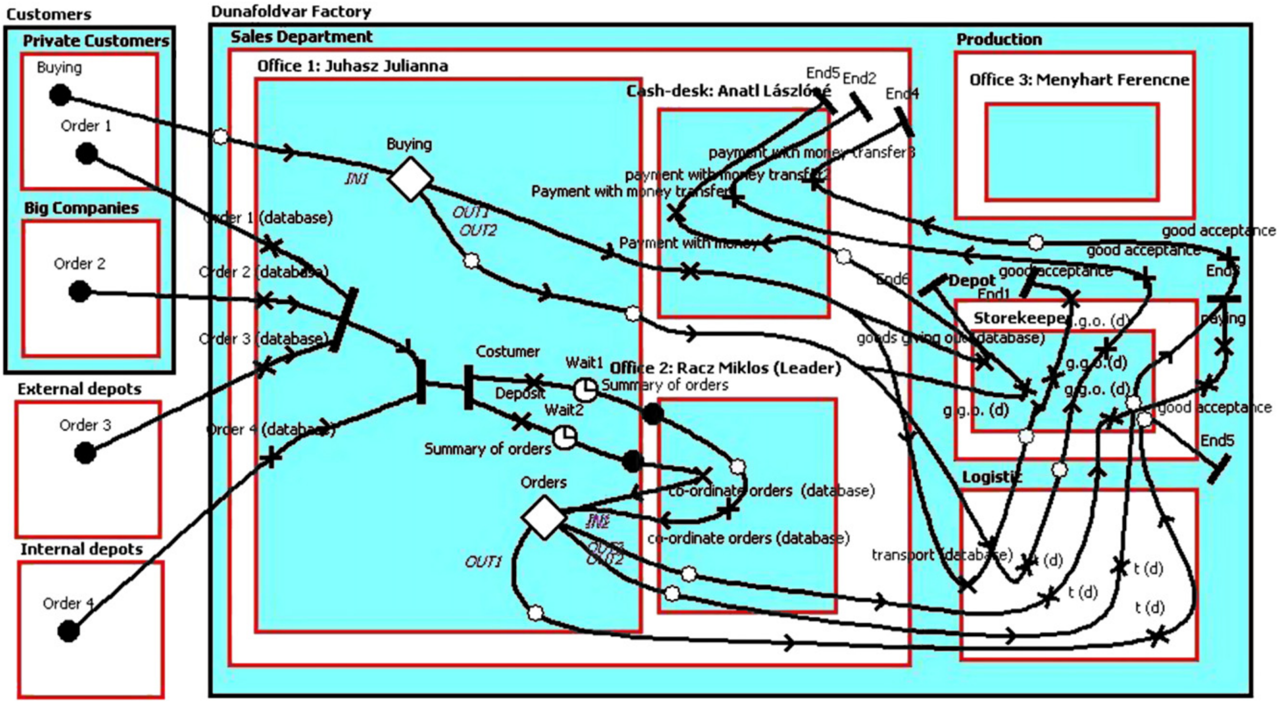

Fig. 3. The root-map of the Sales & Delivery UCM model for user's highlighted workflow processes.

involve or *static stubs*, which contain only one sub-map, or *dynamic stubs*, which may contain multiple sub-maps whose selection can be determined at run-time according to a *selection policy* based on numerous control elements OR-join, OR-fork, AND-join, AND-fork, timer, abort, failure point, and shared responsibilities.

Multiple paths through the same component and by one path triggering and another disabling show scenario interactions. For further details can be found in Amyot et al. [28], in Buhr [26] and [18], and in UCM Virtual Library [29].

The jUCMNav tool, [30] which supports the URN standard actually, is user-friendly with graphical editor under Java-based open-source Eclipse platform. The export-import possibilities in jUCMNav are various by generating the follows types:

- We can generate industry-standard XML files [31], which can be used as input to other tools that support XMI standard and can be imported into an existing model [32].
- The textual MSC files, by scenario definitions used for generating Message Sequence Charts in the Z.120 textual form [7]. The MSC files can be visualized using commercial and research tools [24] (e.g. Telelogic Tau 4.4).
- The intermediate XSLT files, by scenario definitions used for generating scenarios in an intermediate XSLT transformed into other representations such as MSC, UML sequence/collaboration/activity diagrams, TTCN test cases, etc. [30].
- The DXL scripts generated to run in DOORS requirements management system and update the DOORS database for Telelogic DOORS version 7 or above [33].
- The CSM files, by UCM models exported to CSM (Core Scenario Model), an intermediate format used by various tools for performance analysis [25].

The UCM-based re-engineering case study, which we give an overview of, the reengineering of Sales and Delivery processes within the software evolution of legacy system at the industrial unit Messer Hungary Dunaföldvár MbH (abbreviation on MH in the following). The software evolution was caused by SAP database inconsistencies, which result from novel business rules to satisfy orders within outsourcing transports services, and introduce changeable delivering services on-demand. Introducing IT support by two reports on intermediate order and integrating their usage in workflow processes have composed the solution.

There are growing the conflicts between the unique requirements of this work process versus the utilised off-the-shelf SAP software applications. The UCM-based requirements specification of work process dealt to performs the structure of work tasks and subtasks, because the whole flow and interdependencies of work enables the understanding, validation, and the prioritising business needs in appropriate scope of processes [22]. This UCM handled method results in specification of the views of the requesting organisation, which is the functional rather than the whole enterprise area. The requirements specifications for system integration will be included, which contains views on the enterprise area, in the system requirements specification UCM model.

To deduct and solve the emergent inconsistencies we have used UCM to map the actually workflow processes of sales sub-domain involving the users in mapping phase of requirements elicitation. At back-end an architect can observe the mapped processes and he has to set up the user requirements specification and the requirements analysis to perform it before the detailed design phase. After that, the decisions on changes of IT supports and workflow processes have been made. The complete UCM, UML1.3 and ABAP4 solution is given in Medve et al. [34].

To construct UCM scenarios the structural and organizational concepts of the *Sales & Delivery* domain will match the causal relationship concepts of UCM model. In the case of ordering, a product goes along the logistics chain via integrated processes. The sales department accounts for the sale of the manufactured goods, customer service, and the ongoing provision of stores. By means of a document chain, a Sales&Delivery workflow is established, which starts with the first contact with the customer, and it lasts until invoicing of the shipped goods, and passing of the data to Accounting. All phases and options appear on sales documents, enabling all the sales activities to be processed, the elements of the logistics chain to be identified, and real times recorded. Sales and marketing activities can be planned and controlled on the basis of these data. The main processes of system are represented in the UCM root-map; the main activities are detailed in the UCM sub-maps. The activities have allocated workflow-based which follow organizational components as *Customer, Sales, Depot and Logistic.*

Analyzing the main processes of this domain, we seek answers to the following questions: who is interacting with whom, and when is a certain job dealt with at a given location? Based on informal requirements of sales and distribution services acquired during interviews with employees, we have obtained the context view of the sales subsystem.

The scenario of the main processes starts with a triggering event on components *Private Customer, Big Companies,* External *Depot, Internal Depot*, and ends with post-conditions (bar labeled *End_1*, to *End_5*). These components are physically diverse customers who are presented to help the discussions with stakeholders. At the analysis phase these components will be modeled in one abstract component *Customers* and they get labeled conditions at the starting point indicated with a filled circle. We can observe two of stubs, the *Orders* and the *Buying* stubs*,* link to sub-maps *plug-ins*. Fig. 4 shows the *Buying* UCM sub-map.

In the process of directly purchase the BUYING in OFFICE_1 we can see at Fig. 4, who we distinguish old and news consumers in the steps of payment and the CHARGE BUYING sub-map process. The features that conform to the domain of business processes can easily be identified and specified in sub-UCM diagrams.

In the rest of paper we present the UCM model transformation into UML2.0 diagrams, as we can observe in Fig. 1 corresponding to the path (m0-m1-i1-m2) of the UCM-MSC-UML2.0 languages interplay. Depending on the tools, UCM or MSC will be the input document language.

4. UCM model transformations into UML2.0 diagrams

Our experiences in UCM model-based process on combination of UML, UCM, languages we illustrate using the diagram-based method to build UML2.0 model elements with Telelogic Tau G2 tools [33].

In UCM model transformation into UML2.0 diagrams, an important aspect is the fact that the UML modeler builds the diagrams without having any contact with the stakeholders respectively with the UCM modelers, being in touch only with the project manager. This facts follows the most important changes in UML2.0, namely decoupling of activity modeling concepts from state machines and the use of notations popular in the business modeling community. We build the model elements iteratively with focus on architecture and the business rules interoperability. We note that the UCM standard Z.152 [6] express the properties of path traversal mechanisms for UCM, which help to autotrain in translation in mind within the semantics of languages.

In the rest of the paper we present a modification issue with an emergent user requirement improvement. The entire solution is given in [36].

In order to translate UCM maps in UML diagrams we have oriented along to the root UCM and this responsibilities descriptions. At first, we start the analysis at one starting point. We have used the diagram-based method, realising *Use Case, Sequence* and *Class Diagrams* by their iterative improvements applying manual translation of the UCM models. During this preparation, when we realise a certain kind of control of the sequence diagram, possible imperfections of the builded diagram or mistakes in the processes' order can be identified. This is very important, because using the sequence diagram in this way increases the chance of continuing without a mistake. With this early discovering of mistakes, only the reconstruction of the diagram is needed. It leads to a significant time saving.

4.1. Building use case diagrams: they show use cases and actors, and their interrelationships

Use cases and relations have been derived from UCM scenarios and UCM Description Report. We go along one path, which originates in this point by creating a Use Case Diagram, and if we find a responsibility point, we create a use case. We have mapped the components in the UCM diagrams which have organisational or functional roles as actors. The semantics of "extend" and "include" relations in use case diagrams allow insertion of sub-scenarios into an existing scenario according to a selection policy based on numerous control elements (see Fig. 2) OR-join, OR-fork, AND-join, AND-fork, timer, abort, failure point, and shared responsibilities. In case a stub or a plug-in occurs, it is inevitable to make a decision on relation aspects of "include" or "extend" or a simple use case. The "generalise" relation allows systematic modification of scenarios with objectering from workflow-based instances of UCM components and responsibilities of business processes i.e. *External depots, Internal depots, or Payment modalities*, which will be generalised in *Depots* or *Payment* objects in type class or interface class (See Fig. 3).

4.2. Building class diagrams: they show a collection of static model elements such as classes

We have mapped the components in the UCM maps which have organizational or functional roles as active classes or interfaces into Class Diagrams. We create a new or complete existing class method when we are at a responsibility point. We

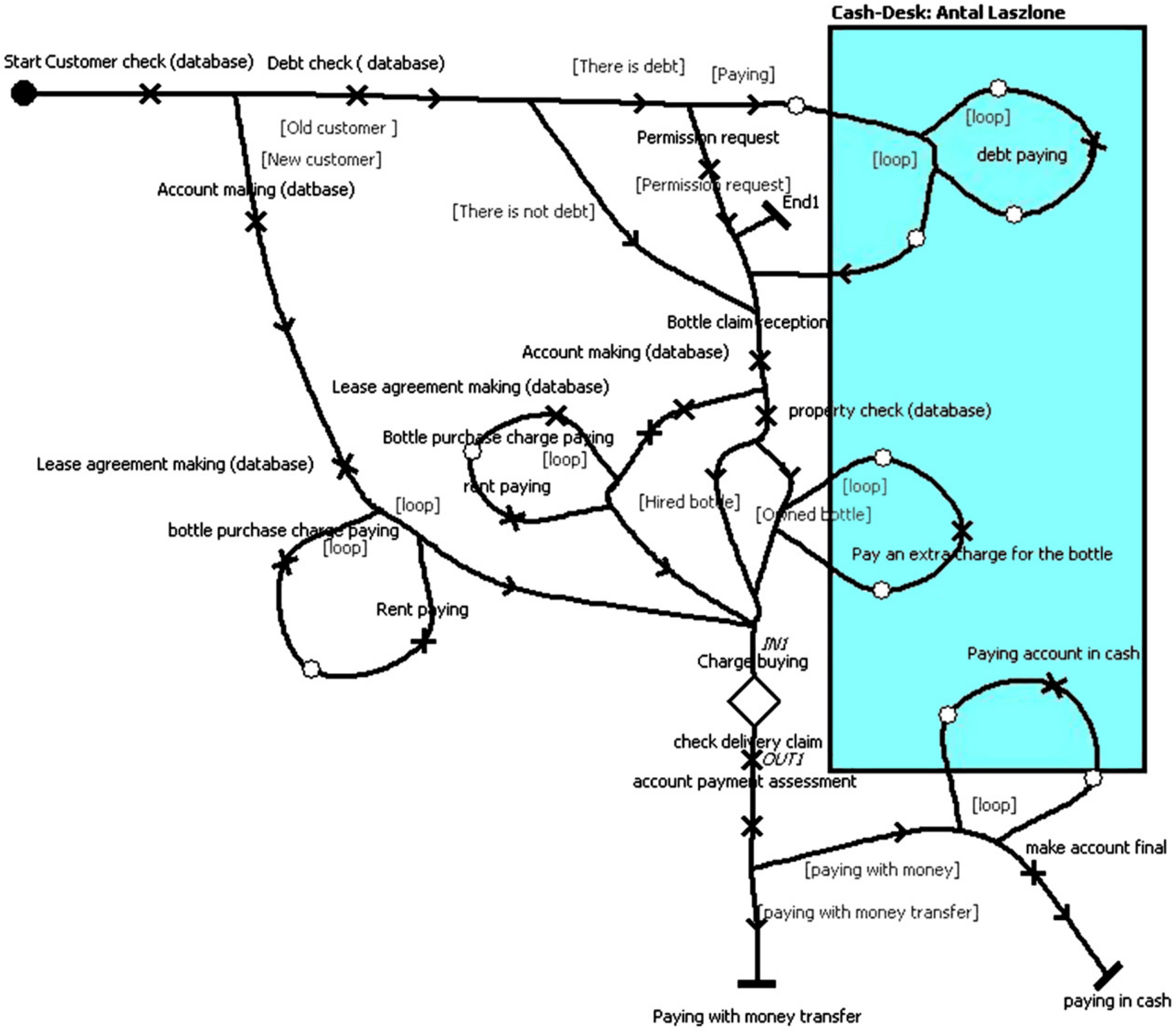

Fig. 4. The UCM sub-map of Buying plug-in.

have mapped the components representing resources of work processes as typed classes into Class Diagrams. When we map guarded ports of active classes, we highlight it with this static structure of behavior aspects of the communications among objects to accomplish goals.

4.3. Building sequence diagrams: they model the sequential logic, that is, the time ordering of messages between classifiers

UML sequences diagrams and MSC diagrams can be translated from UCM maps following transformation rules of Amyot et al. [28]. The components are mapped in instances, the start and end points of path in messages, the responsibilities in actions, the And-Forks in inline-statements, the timers in timers. Abstract messages are derived from an issue when there is a path through components.

We have synthesized our experiences to map abstract messages, as follows:

- We can use the visual design draw-drop advantages when we first complete the objects of class diagrams.
- We propose to derive the interacting instances in parallel with building active classes from components.
- Time sequenced responsibilities and their environment constraints are mapped in the mind;
- Communication flows statements of business-protocols related message sequences corresponding of control points are to be made;
- The messages and their attributes are to be supplemented with goals of business process interoperability.
- Qualitative analysis of process models must be made according [28] et [37].
- Selections policies of path ramifications to build inline operators are to be applied [23].
- The decisions on refactoring use cases given for obtaining documentation segments to highlighting sentences of DFT (design for test) and AO (aspect oriented) of views in the model must be made according to Medve in [38].

The obtained sequence diagrams show a flattened view of the scenarios, while the hierarchical structure of UCMs crowded with stubs and plug-ins disappears. We can model a system with

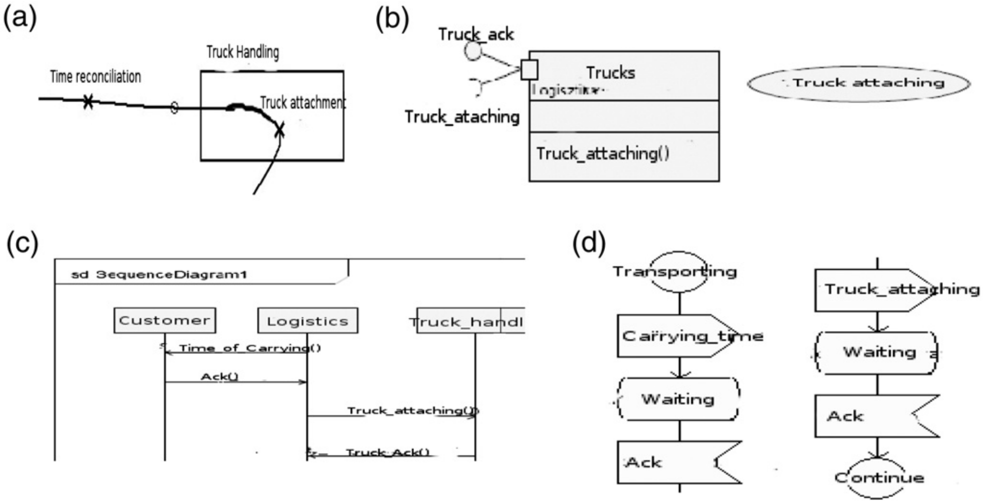

Fig. 5. The diagram-based steps to adding the new component to the UCM-based UML model.

several separate but related viewpoints with sequence diagrams of interacting objects.

4.4. Building activity diagrams: they depict high-level business processes, including data flow

UCM and UML Activity diagrams are complementary. Well-nested UCM maps depending on tool support can be translated into Activity diagrams [19]. Depending on tool support and its visual design forces, we can take advantage to construct Activity diagrams from UCM maps before building sequence diagrams, which are the benefits of controlling the coverage of UCM model before the transformation to reduce the number of iterations.

4.5. Manage changes and assets: an MDA process is applied with reuse

In the following we present a model modification issue with an emergent user requirement improvement. We will attach a truck protocol to Sales&Delivery business rules.

When the UML model is complete and a modification need appears, we recur to the UCM framework and follow an MDA software process. We reach this by creating a new component in the UCM diagram (see Fig. 5 a.) and the related path segment and responsibility for this modification. We have to create a new actor in the use case, and in the sequence diagram (Fig. 5 c.), and we have to create a new class for this component see at Fig. 5 b. In addition, we have to create new methods in the classes, and we have to modify the existing methods and create new state chart diagrams (Fig. 5 d.).

5. Conclusion

In this paper, we have given a survey of the ITU-T System Design familiarised in a previous article of the Computer Standards&Intefaces 27(3)(2005) publication, and a presentation of techniques in re-engineering processes.

In the presentation of languages (ASN.1, URN, MSC, SDL, eODL, TTCN, and UML) used for system modeling and development, we highlight the roles of languages in their interplay in the design processes. Their roles help to model software, data and business through a combination of multiple, open industry standards that are converging.

The UCM techniques are useful to help the mind processes and conversations between modelers and their audiences, often professional consultants, in the phase of elicitation of the business processes and rules. Besides, we can ensure that all members have a similar understanding of the legacy systems process' and a consistent message is transferred to the future documentation users excluding conflicting interpretations which may cause the failure of reengineering.

Our re-engineering experiences with UCM capabilities and also the diagram-based transformation processes of the UCM model elements into UML diagrams are presented. The new components can be very well documented and integrated into the existing system in a manner that even the stakeholders are involved in it. Our workflow-based requirements elicitation with UCM tool-based support of conversations between stakeholders and developers makes it easier to understand user's needs and to correlate with software evolution aspects.

We claim that the initial emotional impact of understanding, and eventually acceptance and positive experience lead the stakeholders to the use of UCM symbols (paper read-write form) in communication between project team and the employees. This contributes to the success of reengineering. We claim that the stakeholders do not need to understand the UCM or UML in detail. They must understand it well enough to discuss the results of the UML, and not model the diagrams.

In the future, we plan to exploit the experience gained during this study on a research of the UCM and GRL roles in the separated specification of domain specific properties and the communication's properties of interactions.

Acknowledgement

We are grateful to Prof. György Kozmann and to Prof. Katalin Tarnay, for their inspiring research work and for their comments on stimulating discussions. We are also grateful to Miklós Rácz, the Sales Delivery Department chief at Messer Hungary Dunaföldvár MbH, for his collaboration in requirements elicitations as far as the industrial case study is concerned.

At last but not least we are also grateful to Tamás Molnár, Ferenc Menyhárt respectively Sándor Pintér, all of them being students and members of the re-engineering project team.

References

[1] G. Adamis, R. Horváth, Z. Pap, K. Tarnay, Standardized languages for telecommunication systems, Computer Standards & Interfaces 27 (3) (2005) 191–205.

[2] ITU-T Recommendations Z series Languages and general software aspects for telecommunication systems, http://www.itu.int/rec/T-REC-Z/en.

[3] ITU-T – International Telecommunications Union Draft Recommendation Z.680 (02/03) ASN.1: Abstract Syntax Notations One, Geneva, Switzerland, 2003.

[4] ITU-T – International Telecommunications Union Draft Recommendation Z.151 (02/03) GRL: Goal-oriented Requirement Language, 2003.

[5] ITU-T – International Telecommunications Union Recommendation Z.150 (02/03) URN: User Requirements Notations – Language requirements and framework, 2003.

[6] ITU-T – International Telecommunications Union Draft Recommendation Z.152 (02/03) UCM: Use Case Map Notation. Geneva, Switzerland, 2003.

[7] ITU-T – International Telecommunication Union, Recommendation Z.120: Message Sequence Chart (MSC), Geneva, 1999.

[8] ITU-T – International Telecommunication Union, Recommendation Z.100: Formal description techniques (FDT)—Specification and Description Language (SDL), Geneva, 1999.

[9] International Telecommunication Union, Recommendation Z.130: Object Definition Language (ODL), Geneva, 1999.

[10] ITU-T – International Telecommunication Union, Extended Object Definition Language (eODL): Techniques for Distributed Software Component Development Conceptual Foundation, Notations and Technology Mappings, Geneva, 2003.

[11] ITU-T – International Telecommunication Union, Recommendation X.292: The Tree and Tabular Combined Notation (TTCN), Test and Test Control Notation – TTCN3, 2003.

[12] ITU-T – International Telecommunication Union, Recommendation Z.600: Information technology—open distributed processing—Deployment Configuration Language(DCL), draft, Geneva, 2003.

[13] OMG Unified Modelling Language: Superstructure, Version 2.0, OMG Final Adopted Specification ptc/03–08–02, January 10 (2003 from http://www.omg.org.

[14] B. Selic, G. Gullekson, P.T. Ward, Real-Time Object-Oriented Modeling, John Wiley & Sons, 1994.

[15] A. Medve, K. Szakolczay, G.Y. Kozmann, IT models for e-health application processes, Book chapter, in: M. Duplaga, K. Zielinski (Eds.), Overcoming the barriers to e_he@lth growth in Enlarged Europe, Health and Management Press, Kraków, 2005, pp. 9–28.

[16] M. Weiss, D. Amyot, Business process modeling with URN, International Journal of E-Business Research 1 (3) (2005) 63–90.

[17] Business Process Modeling (BPM): www.bpm.org, Business Process Rules: www.brcommunity.org.

[18] R.J.A. Buhr, Use case maps as architectural entities for complex systems, IEEE Transactions on Software Engineering 24 (12) (1998) 1131–1155.

[19] D. Amyot, G. Mussbach, On the extension of UML with use case maps concepts, The 3rd International Conference on the Unified Modelling Language, York, UK, October (2000).

[20] Jaap Gordijn, Hans de Bruin, Hans Akkermans, Scenario Methods for Viewpoint Integration in e-Business Requirements Engineering, HICS, 2001 Retrieved September 25 2003 from cs.vu.nl/~gordijn/h3411.pdf.

[21] A. Medve, Entreprise modeling with the joint use of user requirements notations and UML, in: P. Rittgen (Ed.), Enterprise Modeling and Computing with UML, Development Editor IDEA Group, Inc., Hershey, USA, 2006, pp. 46–69.

[22] A. Medve, Workflow-based re-engineering process of socio-technical systems, in: M. Raffai (Ed.), ISBIS'2007 Symposium on Business Information System, 9-10 of November 2007, Széchenyi István University, Győr, Hungary, Novadat, 2007, p. 36.

[23] A. Medve, Standardized formal languages for reliable model engineering, international scientific conference, MicroCAD'2005 International Scientific Conference, Applied Information Engineering,ITTC Miskolc, Hungary (2005) 315–321.

[24] SDL Forum Society: www.sdl-forum.org;

[25] D.B. Petriu, D. Amyot, M. Woodside, Scenario-Based Performance Engineering with UCMNav, SDL Forum, 2003 Retrieved January 10, 2005 from ftp://ftp.sce.carleton.ca/pub/cmw/sdl03-PetriuAmyotWoodside.pdf.

[26] R.J.A. Buhr: Use Case Maps for Object-Oriented Systems, Retrieved September 25 2003 from http://www.usecasemaps.org/pub/UCM_book95.pdf.

[27] Hamou-Lhadj, E. Braun, D. Amyot, T. Lethbridge, Recovering Behavioral Design Models from Execution Traces, 9th European Conference on Software Maintenance and Reengineering, CSMR, 2005, Retrieved September 2, 2006 from http://jucmnav.softwareengineering.ca/twiki/bin/view/UCM/VirLibCsmr05 (2005).

[28] D. Amyot, X. He, Y. He, D.Y. Cho, Generating Scenarios from Use Case Map Specifications, 2003 Retrieved January 10, 2004 from http://www.usecasemaps.org/pub/QSIC03.pdf.

[29] UCM Virtual Library: http://jucmnav.softwareengineering.ca/twiki/virtuallibrary.

[30] jUCMNav: http://jucmnav.softwareengineering.ca/twiki/bin/view/ProjetSEG/WebHome.

[31] W3 Consortium Extensible Markup Language (XML) 1.0. W3C Recommendation, 10 February 1998. http://www.w3.org/TR/REC-xml, (1998).

[32] D. Amyot and A. Miga: Use Case Maps Linear Form in XML, version 0.20, July 2000. http://www.UseCaseMaps.org/xml.

[33] Telelogic Tau G2: www.telelogic.com.

[34] F. Menyhárt, T. Molnár, A. Medve, Követelményelemzés-és specifikálás egy termékgyártó vállalat értékesítési folyamatának újratervezéséhez, (in Hungarian: Requirements engineering for Re-engineering of supply and delivery process and SAP integration), Technical Report, University of Pannonia, 2004.

[35] G. Mussbacher, D. Amyot, M. Weiss, Visualizing Aspect-Oriented Requirements Scenarios with Use Case Maps REV'2006, 2006 Retrieved September 25, 2006 from http://jucmnav.softwareengineering.ca/twiki/bin/view/UCM/VirLibRev06.

[36] S. Pintér, A. Medve, Termékgyártó vállalat értékesítési folyamatának UCM alapú részletes objektumterve UML2.0 és SDL formális nyelveken, (in Hungarian: UCM-based re-engineering of supply and delivery process with UML2.0 and SDL), Technical Report, University of Pannonia, 2006.

[37] A. Pataricza, A. Balogh, L. Gonczy, Verification and validation of nonfunctional aspects, in: P. Rittgen (Ed.), Enterprise Modeling and Computing with UML, Development Editor IDEA Group, Inc., Hershey, USA, 2006, pp. 261–303.

[38] A. Medve, MSC and the aspect-oriented paradigm in protocol engineering, MicroCAD'2003 International Scientific Conference, Applied Information Engineering, 6–7 March 2003, Innovation and Technology Transfer Centre Miskolc, Hungary, 2003, pp. 71–78.

Anna Medve is an assistant professor at the University of Pannonia. Her current research field is the application of formal methods supported with standards of software development in domain specific fields. Her research interests primarily concentrate on the field of requirements engineering and design for testability, patterns of conceptual frameworks. She is a member of the J. Neumann Society, the SDL Forum Society, the IEEE and the IEEE Computer Society.