Domain Specific Language for Securities Settlement Systems

Ojars Krasts
University of Latvia,
Riga, Latvia
ojars.krasts@gmail.com

Arnis Kleins
Riga Technical University,
Riga, Latvia
arnis.kleins@rtu.lv

Artis Teilans
Riga Technical University,
Riga, Latvia
artis.teilans@rtu.lv

Abstract — **Actual problems during design, implementation and maintenance of securities settlement systems software are achieving complementarity of several different, connected, asynchronously communicating settlement systems and verification of this complementarity. The aim of this paper is to create domain specific language for modeling of settlement systems and their interactions. Then use models to calculate settlement systems behavior. Specific of settlement systems requires that they perform accordingly to business rules in any situation. This makes use of model checking a very desirable step in development process of settlement systems. Defining a domain specific language and creating editor supporting it is a first step to enable use of model checking techniques. Created models also can be used as input for other analysis methods and tools, for example, basis path testing, simulation and as base for deriving test cases**

Keywords - modeling; validatioan; verification

I. SETTLEMENT SYSTEM MODELING NEEDS

Securities settlement systems typically operate in connection to other financial institutions settlement systems. De-facto standard in this industry is that communication is implemented as asynchronous exchanges of messages, for example, using SWIFT (Society for Worldwide Interbank Financial Telecommunication) network. Regardless of that normal processing path of each involved settlement system may be quite straightforward; the number of various exception cases can be quite large – communications failures, timeouts, lack of funds, human errors, even misbehavior of connected systems.

High importance requirements to securities settlement systems and settlement systems in general are correctness of systems behavior in any systems state, even in exception situations. Behavior of settlement system should be defined for any system state and exception. So the verification and testing are actual tasks during both the development of new systems and reorganization of existing systems.

In many cases availability of interconnected settlement systems for testing is quite limited - they usually belongs to different financial institutions, possibly in different countries and time zones, and are implemented on different platforms. So there are both administrative and technical difficulties for real systems testing. Cooperatative testing usually is arranged only on last stages of implementation and others institutions systems are not available during design and development phases. However, the behavior of settlement systems is precisely defined in financial institution, or their supervising institution in country, rules and regulations. This allows making models of these connected systems, including the system under development or reorganization. Together these models can be used for various kinds of analysis, verification and test cases design starting with initial development phases of settlement systems, allowing spotting any problems early.

II. MODELING LANGUAGE DIAGRAMS AND SYMBOLS

Before choosing diagram types most adequate for settlement systems modeling, we need to choose what mathematical model or abstraction is closer to them. The first that comes in mind, based on experience in development of such systems are finite state machines and their state change diagrams. Each instance of financial transaction really goes through various states – for example: deal entered, matching completed, funds locked, waiting for confirmations, completed. These state changes are caused by incoming messages, like input symbols for finite state machines, and messages needs co come in certain order.

However there are differences from finite state machines

A. Reading the input messages

Sender of message is important in many cases for settlement systems. State change may be different for different senders. That would be like having multiple inputs for finite state machine, each with its own sequence of input symbols.

B. Asynchronous nature of communication

Messages can arrive in order different from order they are processed. Although messages are placed in queue upon arrival, they can be picked up from queue out of order. This is typical for confirmations processing, when settlement process can look up incoming queue for what it needs

The presented activity has been supported by the European Social Fund within the project "Support for Doctoral Studies at University of Latvia".

C. *Timeouts and time events.*

In settlement systems there can be time events with absolute, astronomical, time and also events with time relative to some time value within an instance of particular financial transaction process. The absolute time events however are typically are not tied to exact clock time, but to significant event in system, which is global, external to particular transaction processing, common for all transactions. Examples are 'start end of day processing', 'sending instructions to central bank deadline'. Relative time events example is 'confirmations deadline', which may be at 16:00 day before settlement date of particular transaction. Time events typically cause exceptions in normal processing flow.

D. *Straightforward state change sequences.*

The main processing flow is usually linear, forward going sequence of states with possibly some alternate symmetric paths for buyer and seller and some forward going jumps over optional sequences. Important here is that backward going jumps and loops are extremely rare - processing do not enter the same state twice. For implementations using inputs to state matrix that means that this state change table will be almost empty for modeling just normal processing flow.

E. *Groups of states.*

Reactions, state changes, to some of incoming messages tends to be common for entire zones in state flow diagrams. Typically this is for time events and exception processing, but reaction to some messages, like cancellation messages, also can be common for groups of states. For example cancelation can be done the same way for all states until any funds are locked or moved – for all data collecting and seller-buyer matching states.

III. Existing Known Diagram Types

Unified Modeling Language (UML) diagrams [1], in particular state change diagrams and activity diagrams with swimlines, are the well known.

However above mentioned settlement systems specifics could be better represented in domain specific diagrams. Even if the symbols visually are the same, the modified, tuned for specifics of settlement systems, semantics of these symbols will make UML-like diagrams more precise.

The UML diagrams are not the only ones which can be used for settlement systems modeling. There already has been modeling tools family GRADE [2] in pre-UML times. One of the latest GRADE versions was used for Latvian Central Depository and Latvia State Funded Pension System modeling. Favorite diagram used in financial systems modeling was 'Business process diagram' – very similar to UML activity diagram with swimlines:

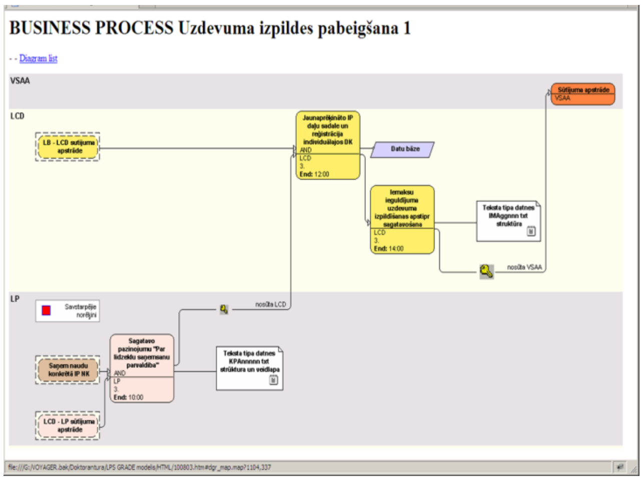
BUSINESS PROCESS Uzdevuma izpildes pabeigšana 1
- - Diagram list
VSAA
LCD
LP
Sūtījuma apstrāde
VSAA
LB - LCD sutījuma apstrāde
Jaunaprēķināto IP daļu sadale un reģistrācija individuālajos DK
AND
LCD
3.
End: 12:00
Datu bāze
Iemaksu ieguldījuma uzdevuma izpildīšanas apstipr sagatavošana
LCD
3.
End: 14:00
Teksta tipa datnes IMAggnnn txt struktūra
nosūta VSAA
Savstarpējie norēķini
nosūta LCD
Saņem naudu konkrētā IP NK
Sagatavo paziņojumu "Par līdzekļu saņemšanu pārvaldībā"
AND
LP
3.
End: 10:00
Teksta tipa datnes KPAnnnnn txt struktūra un veidlapa
LCD - LP sūtījuma apstrāde
file:///G:/VOYAGER.bak/Doktorantura/LPS GRADE modelis/HTML/100803.htm#dgr_map.map?1104,337

Figure 1. GRADE Business Process diagram.

Such kind of diagram presents well interaction between systems in swimlanes, but does not reflect asynchronity of communications. Also process in each swimline is represented as procedural logic rather than state change diagram, which is more adequate to reality.

One domain specific graphic modeling language which combines both state change and procedural logic is Bilingva [3], developed in University of Latvia:

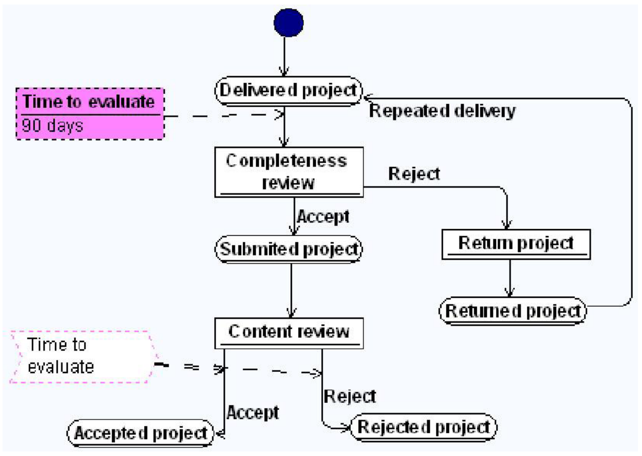

Figure 2. Bilingva diagram

The rounded symbols represent states, rectangular – processing. Some notion for time out also is introduced. However these diagrams are not showing interprocess communications - these are different domain specific, for document processing systems.

Combining these two approaches – the UML-like activity diagrams, showing interactions between systems, and Bilingva, representing both states and processing logic, produces modeling language for securities settlement systems domain.

IV. Modeling Tools Development

The modeling language alone can be used for diagramming, communication with domain business analysts and useable for specification and documentation and specification of settlement systems. The development of domain specific language for settlement systems is ongoing process. But the real value of domain specific modeling language opens up when there is tools support. At first the diagram editors are needed, which can save models, diagrams, as computer files. This enables further development of model analysis tools, with what the problems, mentioned in section 1, of verifying settlement systems correctness can be solved.

Diagram editors development is using metamodel approach. This approach has proven itself as very flexible, especially in cases of research work, when modeling language constantly evolves. This has been used for years by authors [4][5], in some extent even since times when GRADE family of tools [2] was developed.

The underlying technologies, however, change with the time. In the current development are used Microsoft Visual Studio 2010. This version can use Microsoft Visualization and Modeling SDK (VMSDK), which is also based on metamodel approach, and with it diagram editors can be developed very quickly.

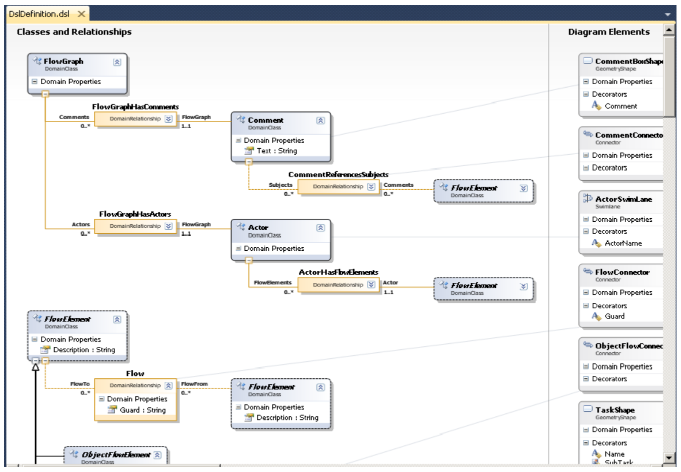

Figure 3. Domain specific language definition using metamodel

This Implemented modeling tool prototype is functioning inside Microsoft Visual Studio Shell. It could be distributed either with Microsoft Visual Studio Shell, or as Microsoft Visual Studio Add-In. The diagrams can be saved in XML files, thus opening access to them from analysis tools.

V. ANALYSIS OF MODELS

Having diagram editors which produces XML model files allows access to model contents for analysis tools and development of such tools.

A. Model Checking

The first tools which we aim to integrate are model checking tools, either existing, like DiVinE[10], for example, or created specifically for settlement systems domain.

Existing tools usually have their own language for specifying the model, but this is a language like programming language, not the graphic settlement systems domain specific language, used by model developers and by business analysts. So to integrate with existing tools is necessary to develop the translator from domain specific language to tools language. The specification of the properties to check is also needed to be done in the tool specific language. This would be valuable addition for inclusion in domain specific language, so it will be possible to specify rules to check using settlement systems domain specific terms and involve business analysts in this process, as in process of development of the model itself.

The situation, when input for model checker is high level domain specific model implies that there is already some level of abstraction. That relieves a task for model checker, making state explosion problem less explicit. As from authors experience in the settlement systems domain, the number of states in each of interconnected processes is below 20.

The organization of model as set of diagrams, each covering certain part of settlement system, can also be considered as expression of modularization, also aiding in reducing complexity of model checker task.

The use of automated translator from domain specific high level graphical language to model checker input is important here, as it eliminates human errors in this transformation.

B. Other analysis methods – Basis Path Analysis

Next analysis tools, we aim to integrate with settlement systems domain models, are for Basis path analysis [6]. Basis path testing is a white-box testing technique, and having models for both system being developed and for connected systems enables us to use it. The comprehensive model analysis, performed by basis path method and possibility to automatically identify test cases brings us closely to goal of settlement system verification.

Basis path analysis has already defined procedures for analyzing model elements with semantics like programming language statements. For domain specific parts, introduced in section 2.1, we need to define how to handle them.

Exception handling and timeouts are producing the most number of control flow paths. If exception or timeout handling block is used about group of elements in model, this means exception condition or time checking after each model element in group, and branching to exception handler. But this checking is essential for settlement systems verification completeness.

C. Other analysis methods – related works

Work already has been done to integrate two other analysis tools with domain specific models:

- Simulation, which required addition of frequency, distribution and probability data to diagram symbols [4][7]. There have been Events and Conditions introduced into domain specific language Meta-model. Events are used to describe situations when activity can be started on the occurrence of specific event. For example, Timer events can occur at definite time or periodically. Activity can be started on event raised by another activity. To simulate forking, branching, joining and merging of activities it might be necessary to add additional conditions, for example branching distribution.

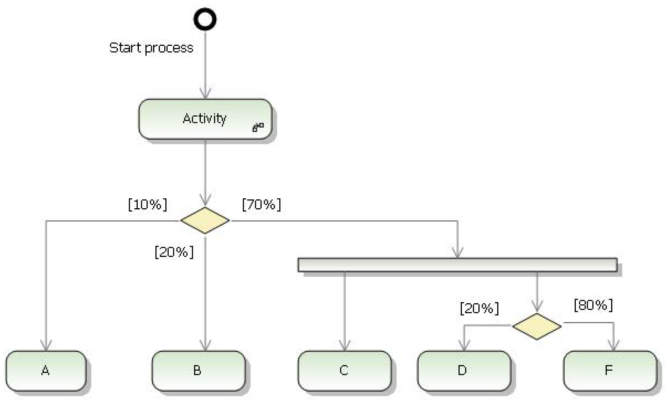

Figure 4. Symbols indroduced for branching simulation

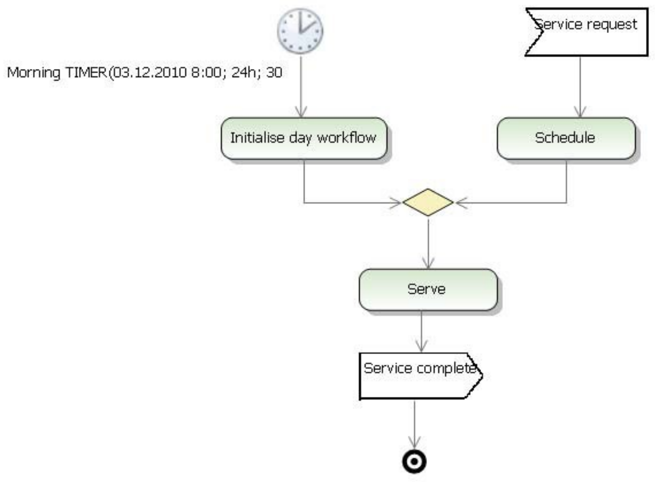

Figure 5. Symbols indroduced for events simulation

• Risk analysis [7][8], integrating special symbols from CobiT [9]. Diagrams using these symbols are considered to be used by system analysts and architects, not by automated analysis tools. These are included for completeness, so model can cover the risk analysis aspect too, which is closely related to model verification.

D. Future work

The work in [7] already combines both simulation and risk analysis into domain specific language. Adding this to settlement systems specific domain fits here well, as both these themes are important aspects in settlement systems. This would produce quite comprehensive tool set for settlement systems domain.

ACKNOWLEDGMENT

The presented activity has been supported by the European Social Fund within the project "Support for Doctoral Studies at University of Latvia".

REFERENCES

[1] UML Superstructure Specification v2, Object Management Group, 2004.

[2] M. Treimanis, O. Krasts, I. Nazartchik. GRADE – GRAPES Development Environment. University of Latvia, Riga, 1992. 109 pages.

[3] J. Ceriņa-Bērziņa, J. Bičevskis, Ģ. Karnītis: Information systems development based on visual Domain Specific Language BiLingva. 4th IFIP TC2 Central and East European Conference on Software Engineering Techniques (CEE-SET 2009), Krakow, Poland, October 12-14, 2009.

[4] A. Kleins, A.Teilans, Y. Merkuryev, O.Krasts. A Metamodel Based Approach for UML Notated Domain Specific Modelling Language. 2011 UKSim 13th International Conference on Modelling and Simulation

[5] Artis Teilans, Arnis Kleins, Uldis Sukovskis, Yuri Merkuryev, Ivars Meirans. 2008. A metamodel based approach to UML modelling. Proceedings of EUROSIM/UKSIM 10th International Conference on Computer Modelling & Simulation. Emmanuel College Cambridge, UK, April 1-3.

[6] Roger S Pressman, R.S. Pressman and Associates. Software Engineering: A Practitioner's Approach, 6/e. ISBN:0072853182, McGraw-Hill, 2005, p. 425

[7] Artis Teilans, Arnis Kleins, Ojars Krasts, Andrejs Romanovs, Yuri Merkuryev, Pjotrs Dorogovs, DOMAIN SPECIFIC SIMULATION LANGUAGE FOR IT RISK ASSESSMENT. Proceedings of 25th European Conference on Modelling and Simulation, 7-10 June 2011, Krakow, Poland.

[8] Artis Teilans, Andrejs Romanovs, Yuri Merkuryev, Arnis Kleins, Pjotrs Dorogovs, Ojars Krasts. Functional modelling of IT risk assessment support system. Proceedings of the 16th International Scientific Conference "Economics and Management-2011 (ICEM-2011)". 27-29 April 2011, Brno, Czech Republic.

[9] CobiT 4.1. (2007). IT Governance Institute.

[10] J. Barnat and L. Brim and M. Ceska and P. Rockai: DiVinE: Parallel Distributed Model Checker (Tool paper); Parallel and Distributed Methods in Verification and High Performance Computational Systems Biology (HiBi/PDMC 2010), IEEE, 2010, 4-7.