A meta-model based approach to UML modelling and simulation

ARNIS KLEINS
Exigen Services DATI
17a Ganibu Dambis, LV-1045, Riga
LATVIA

YURI MERKURYEV
Department of Modelling and Simulation
Riga Technical University
1 Kalku Street, LV-1658, Riga
LATVIA

ARTIS TEILANS
Rezekne Higher Education Institution
115 Atbrivosanas aleja, LV-4600, Rezekne
LATVIA

MAXIM FILONIK
Lattelecom Technology
13 Skanstes, LV-1083, Riga,
LATVIA

Abstract: - This paper is devoted to a meta-model based approach to UML systems modelling and simulation. The approach allows creating a system model by operating with artefacts from the problem domain, followed by generation of a UML model. The discussed approach is illustrated by generating UML models, using Use Case and Activity diagrams of the UML language. As a novelty for UML modelling, especially for simulation purposes, the presented meta-model is enriched by a set of stochastic attributes of modelled activities. Such stochastic attributes are usable for further implementation of DEVS simulators.

Key-Words: -UML modelling, discrete event simulation, meta-model.

1. Introduction

Currently, using Unified Modelling Language is one of the most commonly explored approaches in system modelling. The authors’ experience acquired while working in software development industry shows that all over the world UML modelling is used to some extent in every medium and large scale project.

UML (Unified Modelling Language) belongs to the group of graphical modelling languages. Initially UML was built for information systems modelling to facilitate the development and maintenance processes. Nowadays the usage of UML is broadened. This language is used for building business models, which exceed the initial task of modelling of information systems. The
work at creation of the Unified Modelling Language started in 1996 and is still being continued. Any interested person can have a free access to the UML 2.0 specification on the Object Management Group’s web page [2, 3].

As regards system modelling, UML modelling is widely used at systems development or enhancement phases. This paper introduces a work, which tries to revive a known and approved Meta-model based technology [1] for systems modelling using UML.

A meta-model consists of entities and relationships. Entities represent elements of system’s model, and relationships represent relations between those model elements.

In case of UML, the meta-model based approach maps system model elements on to business domain elements of the modelled system, and, on the other hand a meta-model is mapped on to UML language elements. As a result of modelling, a set of model elements and relationships is created, from which system model graphical UML diagrams could be generated.

Especially for simulation purposes into presented metamodel are incorporated entities and relationships for defining stochastic simulation attributes of a modelled system.

2. UML modelling

UML modelling describes the system’s structure and behaviour. This language consists of

Fig. 1 Object types and relationships for UML modelling
graphical notations called diagrams and builds up an abstract model of a system. The UML standard is maintained by OMG (Object Management Group). In the beginning, UML was built for specification visualization and documentation of IT systems development. Nowadays usages of UML are not only limited to tasks of software engineering. UML is also used for business process modelling and for the development of systems which are not pure information systems.

Modelling with UML promotes model-driven technologies, such as Model Driven Development (MDD), Model Driven Engineering (MDE) and Model Driven Architecture (MDA). Supplementing graphical notations with terms such as class, component, generalization, aggregation and behaviour, helps save system designer’s time for system architectural tasks and design.

UML allows one to introduce new graphical elements and terms, specific for particular problem domain, and create domain specific profiles and stereotypes.

A UML model consists of a set of diagrams. A diagram is a partial representation of the model. A system model could be divided into three parts. The first part is a functional model, which reflects functionality of a system from the system user’s point of view. This kind of model is constructed using Use Case diagrams.

The second part of the UML model reflects the hierarchical structures of a system, using objects, attributes, operations and relations between objects. This part includes Class diagrams.

And finally, the dynamical model reflects internal behaviour of the system. A model of that kind is constructed using Activity, State, Sequence and Collaboration diagrams.

Fig. 2 Object types and relationships for simulation purposes
Models could be created using UML modelling tools. Nowadays, plenty of such software tools exist that support UML model building and differ mainly in price and popularity.

UML 2.0 consists of 13 types of diagrams. For better understanding, the diagrams could be ordered as a hierarchy [1].

A system model to be created with UML language should not necessarily contain all diagrams. For example, creating Information Customer

Fig. 3 Use Case diagrams stored in a meta model

![Use Case Diagrams](image-url)

Fig. 4 Example of stochastic output condition of an Activity

![Stochastic Output Condition](image-url)
System vision model or requirement specification, it is enough for system analyst to create Use Case and Activity diagrams. Use Case diagram answers a question – what a system does. Activity diagrams describe scenarios of every Use Case, i.e., Business processes. Due to that, of the set of UML diagrams, this work uses only Use case and Activity diagrams.

Activity diagrams are extended with stochastic attributes for event occurrences and branching probabilities.

3. Meta-model

In this chapter a Meta-model is presented. It consists of entities and relationships. As the research prototype, a tool is created which fills in entities and defines relations between them by using simple dialogs. The idea of this tool comes from [1].

In the early analysis stages, the structure of a target system may be largely unknown, resulting in frequent mistakes and changes to the system model. The main advantage of this approach is ability to do either autonomous modelling or use it as an initial source for graphical UML modelling tools. This is especially important during the development of UML models for large scale systems. It will organize all important facts from analysis stage according to the Meta-model (Fig. 2). The capture of objects can be performed in any sequence without regarding hierarchy. The object hierarchy and communication paths are entered later by defining relationships between objects. Let’s take a closer look to the Meta-model. An Actor might be a person, a company, computer system hardware or software. An Actor is a role of object outside of a system that interacts directly with it. An Actor could be a special case described by Is Special relationship or generalization described by Is General relationship. Use Case represents modelled system behaviour under various conditions as it responds to actor communicating the system to yield an observable result of value to an actor. Scenarios describe different sequences of system behaviour depending on the particular requests. An activity is a state of doing something. It could be real-world process or software routine. Activities are performed according to scenarios.

The Use case groups together those different scenarios. Activity could be preceded and/or followed by another activity. Activities could be forked to initiate parallel processing and then joined afterwards, or conditionally branched and merged. Activity could give control to another activity and also passing an Object. Activity could be atomic (action) or could be complex consisting of several another activities (actions) described in scenario. This is reflected in the model by Has Scenario / Is Scenario Of relationship. It is described by Is Received / Is Sent and Sends What / Receives What relationships. Use Case could consist of another Use Cases by using Include / Is Included relationship. If Use case is specialised case of another Use Case, it could be depicted by Extends / Is Extended relationship.

For Model to be used for DEVS in full extent it is necessary to compliment it with stochastic attributes. There have been Events and Conditions introduced into given 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 on 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. For this reason Output Conditions are used. Similarly it might be necessary to specify merging conditions which is done using Input Conditions.

Input and Output Conditions are defined by Condition expressions. These are logical expressions which use AND, OR and parentheses and occurrence probabilities. For example, for activity C to be started activities A and B must happen. Therefore Activity C Input Condition is A & B which represents JOIN. Another example, for activity Z to take place A and B or X must happen besides probability that X will take place is 20%. It can be described with expression A & B | X[20%] or with expression (A&B)[80%] | X. If after an activity branching takes place it also can be specified by Condition (Output Condition), which like Input Condition is described with logical expression. For example, to describe sequence of activities represented in Fig. 4 the
following expression can be used $A[10\%] \mid B[20\%] \mid (C \& (D[20\%] \mid F[80\%])) [70\%]$.

Usually model and its entities has several attributes such as queue length, resource capacity, different counters etc. They can be described by special Conditions which are connected to Scenario and Object with Has Attribute / Is Attribute Of relationship. For an activity Duration can be given with special Condition type which is represented with Has Duration / Is Duration Of relation.

5. Conclusions
In the paper, a solution for gathering system functional requirements with further possibility of generating the corresponding UML diagrams is discussed. UML diagrams are extended with stochastic attributes. This provides possibilities for further simulation of a developed model using DEVs simulators [1].

The further research will deal with studying how to gather the functional, architectural and non-functional requirements for information systems. During the future activities these requirements should be transformed into UML Class diagrams, Sequence and Collaboration diagrams or Component and Deployment diagrams. Another direction for further research is the creation of business related wizards for comfortable design of meta-model entities using business domain terminology.

References: