PROPERTY TRANSACTIONS IN SLOVENIA

UML in Use Case Modelling

Modelling is a well-proven and widely accepted engineering technique for controlling complex reality. Unified Modeling Language (UML) is a general-purpose aid for graphical modelling. The author presents a use case driven approach for real-estate transaction in Slovenia.

Short Introduction to UML
By Mathias Lemmens, editor, GIM International

When one wants to convert a process (dynamic system) taking place within a certain domain - such as registration of a transfer act in a land administration system - into a software system, one may approach the process from several directions.
UML (Unified Modeling Language) has been developed to describe the different views on a do-main process in graphical notations in the form of diagrams. No less then nine types of modelling diagrams are distinguished: use case, class (package), object, sequence, collaboration, statechart, activity, component, and deployment. Use cases aim at obtaining system requirements from a userâ€™s view. The *use case diagram* is a collection of use cases, users of the system (actors) and their messages. Use cases are represented by ovals, actors by stick figures and communications by lines that link actors to use cases. The *class diagram* describes the static structure of a system by displaying classes and the relationships among them. It is the backbone of UML and is essentially the communication language for designers. Classes are represented by rectangles divided into three parts: class name, attributes and operations. Lines represent relations between classes. Related classes can be grouped into packages so that overview is kept. A *Package diagram* organises elements of a system into related groups to minimise dependencies between packages. Packages are represented as folders (rectangles with small tabs at the top). The package name is on the tab or inside the rectangle. Dotted arrows represent dependencies.

Objects are the basic elements of a system; they interact by sending each other messages. *Object diagrams* describe the static structure of a system at a certain instant. Class and object diagrams describe the static part of the process, interaction diagrams are dynamic: they describe how objects work together over time. A *sequence diagram* is an interaction diagram, which specifies which messages are sent when. These are arranged according to time: time progresses downward along the vertical axis. The objects involved are listed in rectangles along the horizontal axis at the top, from left to right according to their place in the message sequence. The lifeline is the time that an object exists and is represented by a vertical dotted line. Arrows represent messages. The length of the activation bar represents the duration of execution of the messages. In Figure 6, seller, buyer, sale contract and notary are all classes, which start sending messages at the same time, but the first message sent by the buyer takes longer than the first message of the others. The object rectangles are labelled either by class name, object name or both; class names are preceded by colons (:) *Collaboration diagrams* are also interaction diagrams: they communicate the same information as sequence diagrams; how-ever, the focus is now on roles of objects rather then on the dispatch times of messages. Since time is not represented, the messages are numbered to denote sending order.

Objects may be in different states at differing times, the *state depending* on current activity or condition. A statechart diagram shows the possible states and the transactions that cause a change in state. States are represented by rounded rectangles while transactions are arrows connecting state. Events or conditions that trigger transitions are written beside the arrows. While a statechart diagram describes an object undergoing a process, an activity diagram focuses on the flow of activities involved in a single process. The *activity diagram*, basically a general-purpose flowchart, shows how these activities depend upon one another.
Component diagrams are the software analogues of *class diagram*; they show the types of software components, their interfaces and dependencies. *Deployment diagrams* represent the physical configurations of software and hardware, including nodes, links and dependencies.

Property Transfer
The transfer of a parcel from seller to buyer has to follow a sequence of procedures, which can be described as a system with static structure and dynamic interactive behaviour. UML captures such systems by modelling them as a collection of classified objects collaborating to perform services that ultimately benefit the user of the system. The static structure defines important classes, their properties and relationships. The dynamic behaviour defines the states and modifications of objects over time and the interaction amongst them required for accomplishment of a service. A use case defines such a service for an actor or specific user. The popular starting point is use case modelling that drives the whole development process. The main issue is to develop a suitable method tailored to conceptual modelling of real-estate transaction problems. The case considered here is the simplified sale of a whole parcel, within a Slovenian situation. Our approach is use case driven and we focus on procedural aspects. Figure 1 shows the general context as a UML class diagram; class-oriented development is described in the literature as Core Cadastral Domain Model (CCDM).

Problem Domain

Problem domain modelling concerns selection of objects, generalisation and aggregation of objects into classes and definition of properties (attributes) and relations of objects and classes. Prior to this stage, available data and expertise has to be gathered on property transaction modelling, similar approaches, existing transaction processing systems and detailed user requirements (proficiency acquisition). The resulting abstract model is based on selected semantics, formalism and terminology. Classes, their properties and relations are defined in UML class diagrams. The best sources of domain classes are likely to be high-level problem statements, user requirements and expertsâ€™ knowledge of the problem domain. These domain models also serve as a glossary of terms (ontology of the domain) used during the whole modelling process. The next step is to review the lists of candidate classes and eliminate items that fall outside the scope of the problem domain. During the refining stage, relations among classes should be added. Such relations include associations, generalisations and various dependencies.

The Use Case

A use-case is a sequence of activities performed by an actor to achieve a goal or a service. An actor is a user or an entity such as database or other system outside the problem domain. The relationship between actors and use cases is m to n. This approach aims at capturing important user requirements and possible scenarios. Use case model creation is an iterative and incremental process of writing, drawing and refining text and UML diagrams. The resulting model is an exter-nal view which determines all further stages, including definition of the static data model (class diagrams). This is because use case modelling represents the conceptual centre of the methodology. A complete and unambiguous use case describes one aspect of use without presuming specific design or implementation. The sidebar shows a simplified use case description of the main scenario for the transaction of a whole parcel. Easements are left out because they are usually resolved by special contract. The level of detail in a use case diagram can be adapted to the required level. Here we limit diagrams to an overview level.

Shared Responsibilities

For important use cases, the processing sequence should be outlined on activity diagrams. Activity diagrams enable better understanding of comprehensive use cases. We derived general and detailed activity diagrams to show the sequence and distribution of responsibility for our case, in which the order of activities is generally sequential and defined by laws and other regulations. Figure 5 shows a generalised activity diagram for our case; excluded is the checking procedure for accordance with public regulation pre-emption, mortgagees and easements.

Interacting Objects

Collaboration specifies how a set of connected objects interacts. Interactions consist of sequences of messages sent between objects, modelled by UML sequence diagrams. The dimension of the vertical axis is time, which proceeds from top to bottom; the horizontal axis shows objects or classes and their roles in the collaboration. One sequence diagram depicts the basic and alternate courses within the particular use case, anticipating how such a process will be accomplished over time. The vertical lifeline of an object and its role is shown as a dashed line. When an object is active, its lifeline may be drawn as a double line on top. Messages are shown as arrows from the lifeline of one object to that of another, and are arranged in time sequence down the diagram. An arrow can represent a sequential (normal synchronous) message, an asynchronous (one way) mes-sage, a temporal event or a returning message. The length of the activation rectangle can be used to reflect the focus of control of each object. The flow and progression of messages within the sequence diagram in fact represents the flow of control among the participating objects. Figure 6 shows a partial example for Slovenia. The pre-emption and encumbrance (mortgages and easements) are not included.

Review and Testing

The outcome of the procedural analysis should match well with use case requirements. Verifying and assuring that the â€˜howâ€™ on a sequence diagram matches the â€˜whatâ€™ of the use case is the aim of critical design review: each sentence of use case description and messages across the sequence
diagram should fit. It must always be obvious which object is in control and for how long. Testing of real-estate transaction cases is finally performed by simulating detailed use cases step by step, and by comparing diagrams.

Concluding Remarks

Elaboration of certain detailed UML diagrams can be a demanding task. The availability of powerful and adaptable modelling tools is important. It is crucial that the sequence of steps be co-ordinated and harmonised.

Acknowledgement
This article results from research and development within the framework of the COST G9 action entitled â€˜Modeling real property transactionsâ€™ (www.i4.auc.dk/costg9).

Further Reading

- Bennett, S., Skelton, J., Lunn, K., 2005, Schaumâ€™s Outline Series: UML (2nd Edition), McGraw-Hill.
- Blaha. M., Rumbaugh, J., 2004, Object-Oriented Modeling and Design with UML, (2nd Edition), Prentince Hall.
- Lemmen, C., van Oosterom, P., Zevenbergen, J., Quak, W., van der Molen P., 2005, Further Progress in the Development of the Core Cadastral Domain Model, FIG Working Week 2005 and GSDI-8, Cairo, Eg.
- Rosenberg, D., Kendall, S., 2001, Applying Use Case Driven Object Modeling with UML: An Annotated e-Commerce Example. Addison-Wesley Object Tech-nology Series.
- Rumbaugh, J., Booch, G., Jacobson, I., 2005, The Unified Modeling Language Reference Manual, (2nd Edition), Addison-Wesley Object Technology Series.