UDK 004.43:004.45
Original scientific paper

ROLE OF UML SEQUENCE DIAGRAM CONSTRUCTS IN OBJECT LIFECYCLE CONCEPT

Miroslav Grgec, Robert Mužar
University of Zagreb, Faculty of organization and informatics, Varaždin, Croatia
{migrgec, robert.muzar}@foi.hr

Abstract: *When modeling systems and using UML concepts, a real system can be viewed in several ways. The RUP (Rational Unified Process) defines the "4 + 1 view": 1. Logical view (class diagram (CD), object diagram (OD), sequence diagram (SD), collaboration diagram (COD), state chart diagram (SCD), activity diagram (AD)), 2.Process view (use case diagram, CD, OD, SD, COD, SCD, AD), 3. Development view (package diagram, component diagram), 4. Physical view (deployment diagram), and 5. Use case view (use case diagram, OD, SD, COD, SCD, AD) which combines the four mentioned above. With sequence diagram constructs we are describing object behavior in scope of one use case and their interaction. Each object in system goes through a so called lifecycle (create, supplement object with data, use object, decommission object). The concept of the object lifecycle is used to understand and formalize the behavior of objects from creation to deletion. With help of sequence diagram concepts our paper will describe the way of interaction modeling between objects through lifeline of each of them, and their importance in software development.*

Keywords: *UML (Unified Modeling Language), sequence diagram, object, object lifecycle, lifeline, software, model, code, RUP (Rational Unified Process), software development.*

1. INTRODUCTION

UML sequence diagrams are used to represent or model the flow of messages, events and actions between objects or components of a system. Sequence diagrams are used primarily to design, document and validate the architecture, interfaces and logic of the system by describing the sequence of actions that need to be performed to complete a task or a scenario. They are useful design tools because they provide a dynamic view of the system behavior which can be difficult to extract from static diagrams of specifications.

By RUP and “4+1 view” of system (logical view, process view, development view, physical view and use case view), sequence diagrams are used in analysis and design, business modeling and testing phase. Regarding to mentioned phases, sequence diagrams could be used to explore the logic of a complex operation, function or procedure. In this paper we will describe the role of sequence diagram constructs in the concept of object lifecycle.

2. BASIC NOTIONS, CONSTRUCTS AND USAGE OF UML SEQUENCE DIAGRAM

A sequence diagram is a dynamic diagram that shows what happens during the time. In this diagram, all the details of the operations are specified, and the messages that each object involved in the operation sends to the others is detailed together with the time instant at which it happens. The diagram is built with time that progresses from up to down, and the objects are ordered from left to right according to the time instant in which they appear in the sequence of messages.

A sequence diagram is a time-based representation on messages in the system because it shows time line of events that happen in the system. Sequence diagram shows what happens when a particular flow through a use case or activity diagram is executed. It also shows set of collaborating objects, in addition to showing the messages that pass between them.

2.1. OBJECT

Object is a discrete entity with a well-defined boundary and identity that encapsulates state and behavior; an instance of a class. By Martin's and Odell's definition, "An object is anything to which concept[1] applies. It is an instance of a concept." [3]. An object is an instance of a class[2], which describes the set of possible objects that can exist. As an instance of a class, object contains attributes and its values, and also it contains methods that can be used for manipulation with attribute values. An object can be viewed from two related perspectives: as an entity at a particular point in time with a specific value and as a holder of identity that has different values over time. Both views can coexist in a model, but not in the same object or class. The first view is appropriate to a snapshot, which represents a system at a point in time. An object in a snapshot has values for each of its attributes. An object is attached to a collection of links that connect it to other objects.

In UML object is depicted with rectangle inside which there can be shown only its name or its name and class name whose instance it is (Figure 1). By the object name, inside rectangle, there can be specified all or only specific object attributes. If object attributes are depicted, than rectangle is horizontally divided into two sections. Upper section contains object name, and lower section contains depicted attributes. To distinguish object notation from class notation, name of object is underlined and written in lowercase. Object in UML can be named on three different ways (Figure 1):

- object – named object of some unnamed class,
- object:class – named object and named class,
- :class – unnamed object with named class.

[1] A concept is an idea or notion that we apply to the things, or objects, in our awareness [3].

[2] Class is an implementation of a concept or type. It is the construct most commonly used to define abstract data types in object oriented programming languages [3]. Each class has a name and contains attributes that describe certain class, and it contains methods that manipulates with attributes.

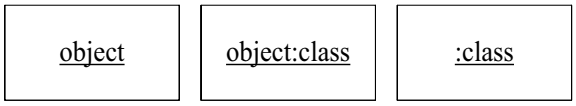

Figure 1. Object representation in UML

In sequence diagram objects are depicted with rectangle and vertical line extension below the object. This line is called lifeline.

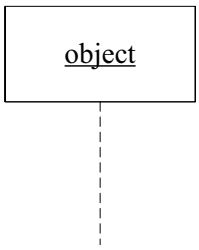

Figure 2. Object representation in UML Sequence Diagram

2.2. LIFELINE

Lifeline is a role in an interaction that represents a participant over a period of time and, by extension, the participant itself. In other words, it represents the existence of the participant over the time. In a sequence diagram, it is shown as a vertical dashed line, parallel to the time axis, with a head symbol showing its name and type [5], [6]. Vertical line illustrates the life span of an object within the context of a single interaction.

2.3. EXECUTION SPECIFICATION

Execution specification is the specification of the execution of an activity, operation, or other behavior within an interaction. An execution represents the period during which an object performs a behavior either directly or through a subordinate behavior. It models both the duration of the execution in time and the control relationship between the execution and its invokers [5], [6], [7]. Focus of control is another name of an execution specification.

On a sequence diagram execution specification is drawn as a tall, thin rectangle, the top of witch is aligned with its initiation event and whose bottom is aligned with its completion event. Execution specification can be nested by stacking another execution specification slightly to the right of its parent (Figure 2).

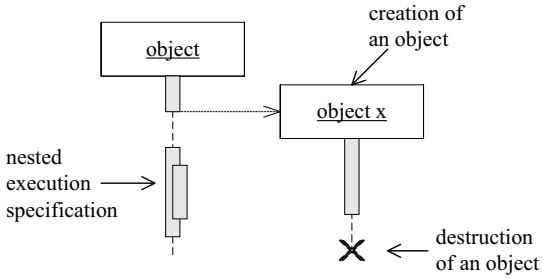

Figure 2. Object execution specification, creation and destruction of an object

2.4. OBJECT COMMUNICATION

Objects communicate with each other by exchanging messages.

Message is the conveyance of information from one role to another as part of an interaction within a context. At the instance level, message is a communication from one object to another. A message may be a signal or the call of an operation. The sending and the receipt of a message are occurrence specifications [5], [6], [7].

On a sequence diagram message is drawn as a solid arrow from one lifeline to another, indicating that one object has requested some work from another object. Object's request may represent method call, Web service message, socket message or any other mechanism that object can use for contacting another. Message is described using following syntax:

[guard][iteration] sequence_number : return_variable := operation_name (argument list)* ([4], [5])

This is complex syntax. Usually, in process of sequence diagram modeling, messages are depicted with solid arrow and operation name with arguments above an arrow (Figure 3).

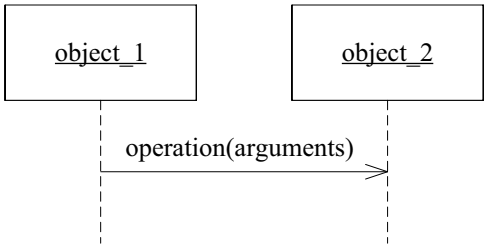

Figure 3. Sequence diagram communication

2.5. CREATING AND DESTROYING OBJECT

Sequence diagrams can show creation and destruction of an object.

Creation of an object is the result of an action that instantiates the object. When an object exists prior to the beginning of an interaction, the object icon is placed at the top of the sequence diagram. If the object is created during the execution of the interaction, the object icon appears at a relative vertical position within the diagram coinciding with the

point in time when it is created (Figure 2). Message that initiates creation of on an object during the execution of the interaction is drawn as a dashed line with open headed arrow.

Destruction of an object is the elimination of an object and the reclaiming of its resources. As shown on Figure 2, on a sequence diagram destruction of an object is shown by large "X" at the end on the lifeline of the object. The destruction can be further indicated by drawing the lifeline of some object to the point at which it was destroyed [2].

2.6. OBJECT ACTIVATION AND SUSPENSION

When a message is sent to an object it invokes a behavior on that object and the object is activated. Object activation is depicted as a top of execution specification rectangle on the lifeline. The typical start of execution specification is at the point that a message hits the lifeline, placing a demand on the resources of the object. Object suspension is depicted at the end of execution specification, and usually occurs when either the behavior is completed or control is returned to the requesting object (Figure 4).

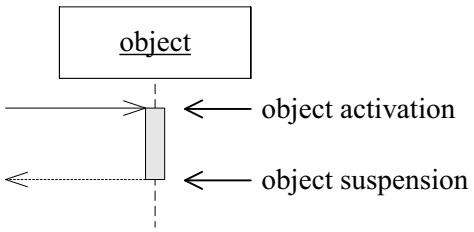

Figure 4. Object activation and suspension

2.7. INTERACTION FRAMES

Sequence diagram is mainly used as tool for depicting object interaction, but not as tool for modeling control logic. When there is need for depicting control logic (structured control operators) on sequence diagram, interaction frames[3] (Figure 5) are used.

[3] A frame provides a portable context for a diagram. Once enclosed in a frame, a diagram may be nested inside another frame/diagram. By nesting a frame inside of another frame, the enclosing diagram effectively reuses the entire enclosed diagram inside the new context [4].

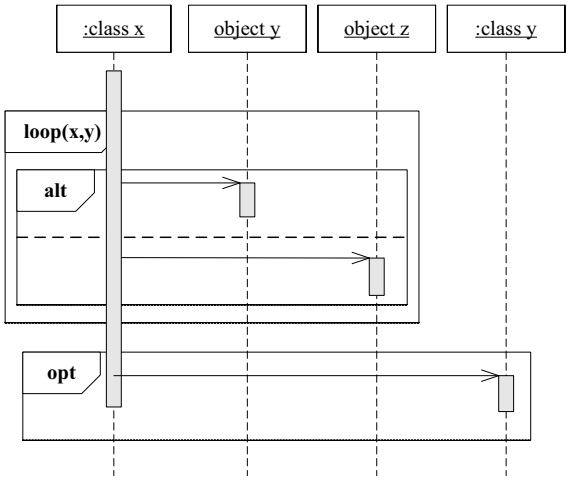

Figure 5. Interaction frames

Interaction frames consist of some region of sequence diagram that is divided into one or more fragments. Each frame consists of an operator and each fragment may have a guard[4]. In Table 1 are listed common operators for interaction frames.

Table 1. Common operators for interaction frames [2]

Operator	Meaning
alt	Alternative choice among multiple fragments. Only the one whose condition is true will execute.
opt	Optional fragment that will execute only if the supplied condition is true.
par	Multiple parallel fragments that execute in same time.
loop(x,y)	Fragment in loop frame may execute multiple times and guard indicates the basis of iteration.
region	Critical region in fragment that cannot be interleaved with any other interaction. Only one thread can execute at once.
neg	Negative. Fragment shows an invalid interaction.
ref	Reference that refers to an interaction defined on another sequence diagram. The frame is drawn to cover the lifelines involved in interaction. Parameters and return values can be defined.

[4] Guard condition - A condition that must be satisfied to enable an associated transition to fire [5], [6], [7].

Operator	Meaning
sd	Sequence diagram. Used to surround an entire sequence diagram.

2.8. USE OF SEQUENCE DIAGRAMS

Sequence diagrams are typically used to model usage scenarios, logic of methods and logic of services.

A usage scenario is a description of a potential way some system is used. The logic of a usage scenario may be part of use case, entire pass through a use case[5], or pass through the logic contained in several use cases [1].

Sequence diagrams could be used to explore the logic of a complex operation, function or procedure.

Service is effectively a high-level method, often one that can be invoked by wide variety of clients. This includes Web services as well as business transactions implemented by variety of technologies [1].

3. OBJECT LIFECYCLE REPRESENTATION WITH UML SEQUENCE DIAGRAM

Objects in a system are not only some compartments that contain data combined with some functions to manipulate with the data. Each object in a system has a life. Attributes values of an object are hidden inside the object, what means that object has a memory. Before it can be used, object must be created and filled with attributes values. That is the beginning of its lifecycle. During its lifecycle, object's attributes values are consumed and object is changing its state. Finally, when object isn't needed anymore it has to be removed. That is the end of its lifecycle.

In UML objects lifecycle is depicted primarily with a state machine diagrams[6]. State machine diagrams can be constructed by checking sequence diagrams to see which events are important to the more dynamic objects in a system. One can see whether a sequence diagram shows one object sending an event to another, and whether a second object, stimulated by that incoming event, have to make a transition from its present state to another state.

3.1. STATE INVARIANT

Sometimes the response of an object to a message depends on the condition or state of the object at the time that the message is received. UML 2.0 adds constraint notation to the lifeline to support modeling of the required state. State describes the condition of the object in terms of the values of its attributes at a point in time.

UML offers another one construct on a sequence diagram, state invariant (Figure 6). State invariant is an assertion that a given constraint must be true when a certain state is

[5] Use case – is a specification of sequences of actions, including variant sequences and error sequences, that a system, subsystem or class can perform by interacting with outside objects to provide a service of value[5], [6], [7].

[6] State machine diagram shows a state machine, including simple states, transition, and nested composite states. State machine is a behavior that specifies the sequence of states an object goes through during its lifetime in response to events, together with its responses to those events [5], [6], [7]. State in UML is depicted by rectangle with rounded corners.

active. If it is not true, the model is an error. A state invariant may also be placed on a lifeline within an interaction. The interval between occurrence specifications is equivalent to a state. A state invariant on a lifeline may be shown by superimposing the text of constraint (in curly braces) over the lifeline or by placing a state symbol containing the name of a state on the lifeline. State invariant may be specified before an interaction to define the conditions under which the set of messages may be passed. A state invariant may also be used to following a series of messages to define the required condition of an object after completion of the series [4], [5], [6], [7].

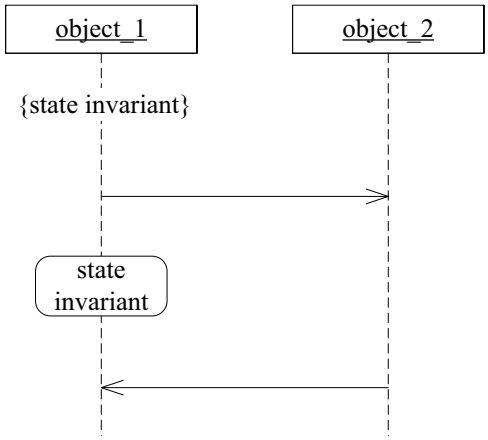

Figure 6. State invariants on lifeline

3.2. PROPOSITION FOR DEPICTING STATES ON A LIFELINE

We consider depicting states by rectangle with rounded corners on a lifeline of an object to be large and not practical. If there are many possible states of an object and all are depicted on a lifeline with a state symbol, sequence diagram can be vast and confusing.

If there is an object on sequence diagram that has a lot of events going into them, it is a candidate for modeling its lifecycle with state machine diagram, because it has the most state transitions. We propose to depict such states by rounded ovals on lifeline as it is depicted on Figure 7. Such ovals consume less space than rectangles with rounded corner and are not confusing.

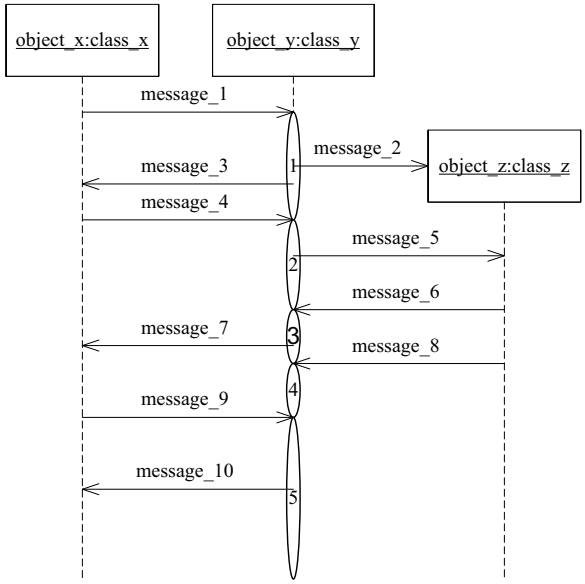

Figure 7. Oval depiction of states on lifeline of an object

Because of the simplicity states depicted in that way can be numbered.

Figure 7 shows one general example of a sequence diagram with depicted states with oval symbol. Points where object object_y as an instance of a class *class_y* receives an event is point of state change. That is the start of an oval representation of a state. Oval is stretched to the end point where object receives another event. The end point can be a start point for a new state or point in time where object is destructed.

CASE (Computer Aided Software Engineering) tools that support UML can be upgraded so that from lifeline state machine diagram can be generated with all states that are depicted with ovals.

3.3. EXAMPLE OF OVAL DEPICTION OF STATES ON A LIFELINE OF AN OBJECT

Figure 8 and Figure 9 illustrates a scenario interaction between John (an instance of certificate clerk), certificate, and contact and certificate type. On both of those figures we are interested in lifecycle of an unnamed object *:certificate*. On Figure 8 state of an unnamed object is depicted by rectangle with rounded corners and on Figure 9 state of an unnamed object is depicted by oval. Sequence diagram scenario is described as follows:

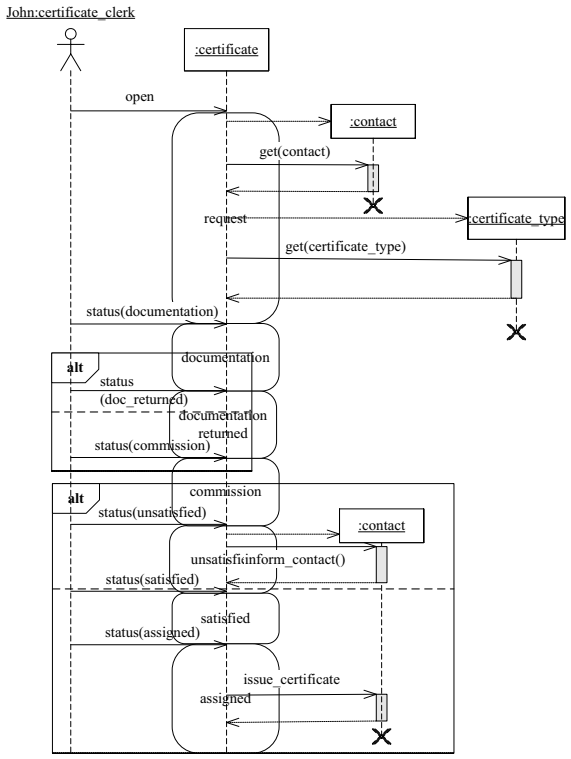

Figure 8. Example of sequence diagram with depiction of states on lifeline of an object as it is defined by UML state concept

John work as a certificate clerk. Certificates are assigned to particular persons (contacts) that acquired criteria for testing natural gas installations. John opens new request for a certificate and assigns person (contact) and certificate type to it (status = request). Contact delivers documentation required for certificate issuing that will be checked (status = documentation). If documentation doesn't satisfies certain conditions, it is returned to contact (status = documentation returned). If documentation satisfies certain conditions, contact must pass an examination in front of board of commissioners (status = commission). In front of board of commissioners contact can pass an examination (status = satisfied) or can not (status = unsatisfied). If contact hasn't passed an exam, in certain time contact can send new request for a certificate. To contact, that passed an exam in front of board of commissioners, certificate is issued (status = assigned). Certificate has duration of few years and after that he or she have to go through scenario again.

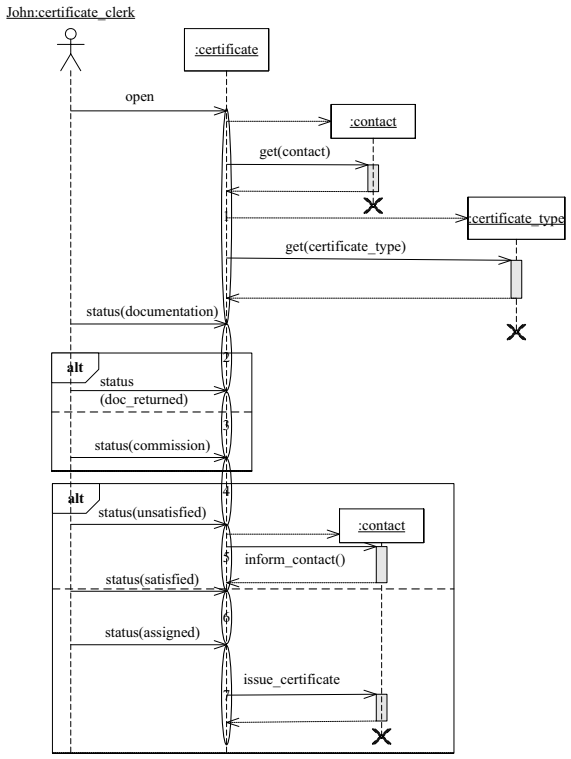

Figure 9. Example of sequence diagram with oval depiction of states on lifeline of an object

States on Figure 9 are depicted with oval and such sequence diagram is more readable than sequence diagram with states on Figure 8 depicted by rectangle with rounded corners, and lifecycle of an unnamed object *:certificate* can be easily followed.

4. CONCLUSION

Every object in the system is depicted by attributes and their function is depicted by methods. The object in the system goes through certain states which are results of methods influencing the attributes of the object. Every influence or possible event causes the change in the state of the object. The change in the state of the object from its creation to its destruction represents its lifeline. UML defines the concepts of the sequence diagrams, which basic concepts are chosen and described in this paper, and thus makes it possible to represent these changes in the state of the object in the system graphically. Activation of the object in the system starts when the message is sent to the object. The receipt of the messages starts the change in the state of the object in the different phases of its lifecycle. The concepts of SD enable the modeling of these interactions between the objects (the concept of the lifeline). Because the existing concept has proven to be impractical, the authors of the article suggest a new oval shaped concept to represent the lifecycle of the object on the lifeline (Figure 8).

REFERENCES

[1] Ambler, S. W. (2004): *Object Primer, Third Edition*, Cambridge University Press, Cambridge

[2] Fowler, M. (2003): *UML Distilled, A Brief Guide to the Standard Object Modeling Language*,3rd, Addison Wesley, Boston

[3] Martin, J.; Odell, J. J. (1998): *Object-oriented Methods – A Foundation, UML Edition*, Prentice Hall PTR, New Jersey

[4] Pender, T. (2003): *UML Bible*, John Wiley and Sons, Boston

[5] Rumbaugh, J.; Jacobson, I.; Booch, G. (2004): *The Unified Modeling Language Reference Manual, Second Edition*, Addison Wesley, Boston

[6] Rumbaugh, J.; Jacobson, I.; Booch, G. (2005): *The Unified Modeling Language User Guide 2nd Edition,* Addison Wesley, Boston

[7] *UML 2 Superstructure Specification*, http://www.omg.org/cgi-bin/doc?ptc/05-07-04(14.04.2006)

Received: 11 March 2007

Accepted: 29 October 2007