JOURNAL OF OBJECT TECHNOLOGY

Online at http://www.jot.fm. Published by ETH Zurich, Chair of Software Engineering ©**JOT, 2005**

Vol. 4, No. 6 Special Issue: Use Case Modeling at UML-2004, Aug 2005

Use Case from the ODP Viewpoint

Joaquin Miller, Lovelace Computing Company

Abstract

Discussion of use cases and UML often focuses on the UML use case diagram. Use case diagrams are contrasted (usually unfavorably) with text use cases. But specification of use cases with UML is not limited to the use case diagram. In fact UML 2 includes a variety of expressive techniques for specifying a use case. This article looks at UML use cases from the ODP viewpoint, and describes UML use case specification techniques in ODP terms.

1 INTRODUCTION

The Reference Model of Open Distributed Processing (RM-ODP) [ODP] provides a language for interpreting the concepts of a modeling language, for building models and for explaining the notions of conformance to a model and of conformance testing.

Because of the great practical value of the Reference Model in specification of software, and for thinking about languages for the specification of software, it can be helpful to use ODP concepts to understand other modeling concepts.

In this short contribution I offer one way to interpret the use case of UML 2 [UML2] using ODP concepts. My hope is that this will provide a useful framework for thinking about the concerns of other articles in this issue.

2 THE QUESTION AND ITS ANSWER

The question is:

In a specification of a particular system, what is or should be the relationship between the ODP concepts and the UML use case concepts.

The answer is:

A particular use case of a certain system is a part of the community contract of a community of a certain type. Let's call communities of that type 'use case communities.' In that use case community, that system is represented as a single object. The roles in that community are of two kinds, one or more actor roles and a single subject role. The object

Cite this article as follows: Joaquin Miller: "Use Case from the ODP Viewpoint", in *Journal of Object Technology*, vol. 4, no. 6, Special Issue: Use Case Modeling at UML-2004, pp. 109 -118
http://www.jot.fm/issues/issue_2005_08/article9

representing the system fulfills the subject role. Objects in the environment of the system fulfill the actor roles. The objective of that community is the observable result of value of that use case (the value to one or more of the actors of that use case).

This explains "the key concepts associated with use cases … actors, use cases, and … subject." [16.1]

> *References of this form are to sections of the UML 2.0 Superstructure Specification. See reference [UML2]. This is a reference to section 16.1 of the specification.*
>
> *As in the preceeding quotation, when it suits my didactic purpose, I will take the liberty of deleting portions of text quoted from the UML 2 specification.*

The relationships between the use cases in the specification of a system (specialization, inclusion, and extension), are discussed by Genilloud and Frank in another article in this issue. [Genilloud05] They also discuss use case types, use case templates, and use case actions (i.e. use case occurrences, use case executions, instances of use case types, or instantiations of use case templates), so I'll adopt a relaxed attitude to these important distinctions.

3 ODP

The Reference Model of Open Distributed computing provides, in Part 2, an analytical framework and concepts for specification of systems. In Part 3, the reference model provides five viewpoints and corresponding specification languages.

The enterprise viewpoint is "a viewpoint on a … system and its environment that focuses on the purpose, scope and policies for that system. [3-4.1.1.1]

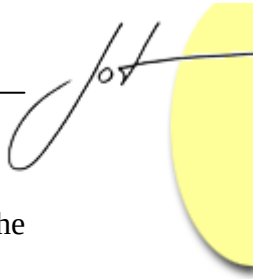
> *References of this form are to clauses of the RM-ODP standard. This reference is to subclause 4.1.1.1 of Part 3, X.903 | IS 10746-3. References of this form that begin with '2-' are to Part 2, X.902 | IS 10746-2, those beginning with '11-' are to X.911 | IS 15414. See reference [ODP].*

"The enterprise language uses concepts taken from [Part 2], and introduces refinements of those concepts, prescriptive rules and additional viewpoint-specific concepts…"

Enterprise language concepts important for use cases are object, action, behavior, role, contract, community, and objective or purpose.

> *Another important enterprise language concept is policy. It is not essential for this discussion.*

A community is "a configuration of objects formed to meet an objective." [3-5.1.1] "In an enterprise specification, an ODP system and the environment in which it operates are represented as a community. At some level of description the ODP system is represented as an enterprise object in the community." [3-5.2]



"The objectives [i.e., purpose] and scope of the ODP system are defined in terms of the roles it fulfils within the community of which it is part…" [3-5.2]

The definition of role in Part 2 is problematic. [2-9.14] For our purpose here, it will be useful to think of a role as a model element used to specify an action or behavior without identifying a particular participant in that behavior. That is, a role serves as a placeholder, used when the specifier of a community does not wish to be more specific at that place in the specification. [3C UML] In the operation of a system, a specific object fulfills that role. Likewise, roles are used to specify a community template, and objects are bound to (some of) the roles when a community is instantiated using that template.

Thus, a community in which the system is represented as a single object may be specified as comprised of that object and roles for objects in the environment of that object. Let's call these 'environment roles.' These roles specify the behavior of the objects in the environment of that system. This enables specification of the community without identifying specific objects in the environment. For uniformity, the behavior of the system may also be specified as a role or roles, the role(s) of the system.

The scope of the system is "the behaviour that system is expected to exhibit." [11-6.1.1] The objective of the system is its "practical advantage or intended effect…" [11-6.2.1]

The specification of a community includes a community contract. "The objective of the community is expressed in a contract that specifies how the objective can be met. This contract: states the objective for which the community exists, governs the structure, the behaviour and the policies of the community, constrains the behaviour of the members of the community, [and] states the rules for the assignment of enterprise objects to roles." [11-7.3.1]

4 UML

A UML 2 "use case is the specification of a set of actions performed by a system, which yield an observable result that is…of value for one or more actors..." [16.3.6]

Use cases "are used to capture the requirements of a system, that is, what a system is supposed to do." [16.1]

"The subject [of a use case] is the system under consideration to which the use cases apply." [16.1] "Use cases need not be attached to any specific subject..." that is, a UML use case can be used to describe the behavior of more than one system.

A UML 2 actor is a collection of roles.[16.3.1] One role per use case of that actor.

The specification says that a UML 2 "actor specifies a role." If intended, this is a change from UML 1. In UML 1, "each actor defines a coherent set of roles." [UML1.5] This is not noted as a change in the UML 2 specification; no rationale for this change is given. Since the UML 2 specification tells us that "the term 'role' is used

informally" in the discussion of actor, I take the liberty of interpreting UML 2 actor and this use of 'role' as I explain in this paper.

A use case can have several actors. In a use case, the system also fulfills a role, UML calls this role, 'subject.' The purpose of the system in the role, subject, corresponds to the purpose of the use case.

A use case diagram does four things. First, it provides a list of use cases. Second, it specifies the actors of each of those use cases. Third, a use case diagram may also show specification relationships between the use cases. Finally, it may identify a subject for the use cases. We are concerned with the second purpose. To be precise about that, I need to write: A use case diagram specifies, for each of the use cases, the actors with roles in that use case.

A UML use case is a behaviored classifier. Whatever else that means, it means that a use case may have UML behaviors. "The behavior of a use case can be described by a specification … such as interactions, activities, and state machines, or by pre-conditions and post-conditions as well as by natural language text … It may also be described indirectly through a collaboration … These descriptions can be combined." [16.3.6]

Textual use cases are specifically authorized by UML 2, and they can be used exclusively, or combined with other techniques of behavior specification, such as the widely used interactions. The UML action language provides the capability to completely specify a behavior of any of the UML behavior kinds.

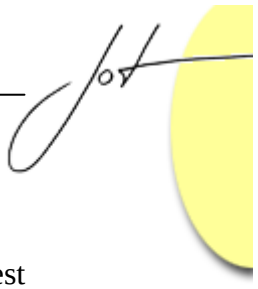
When use cases are used systematically, the set of use cases of a system can be a partition of the interactions of the system with its environment. Then those use cases completely specify the externally observable behavior of that system.

To complete the picture, we will need one more element of UML, collaboration. "A collaboration describes a structure of collaborating elements (roles), each performing a specialized function, which collectively accomplish some desired functionality." [9.3.3] "A collaboration use represents the application of the pattern described by a collaboration to a specific situation involving specific classes or instances playing the roles of the collaboration." [9.3.4]

I don't grok the way the UML 2 text specifies collaboration, so I can only offer a naive discussion here.

The UML 2 specification tells us that a use case "…may also be described indirectly through a collaboration that uses the use case and its actors as the classifiers that type its parts." [16.3.6] I'll stick my neck out, and say that a use case may be specified using a collaboration with the subject of that use case and the roles of the actors in the use case as the roles of the collaboration.

"A collaboration in UML 2.0 is a kind of [behaviored] classifier, and can have any kind of behavioral descriptions associated." [9.3.3] That is, a collaboration declares the roles, and the behavior can be described by a specification such as an interaction, activity, or state machine; thus, when a collaboration is used, the behavior of the use case is "described indirectly."



Role bindings are used to assign participants to a collaboration use.

I feel that representation of a use case as a collaboration along with a behavior best serves the objectives of a model: The roles of the collaboration are the actor and subject roles of the use case. A role binding serves to identify the subject. The use case diagram can then be used to list the use cases and specify their specification relationships.

Now we have all the elements we need to use UML to explain UML use case in ODP terms.

5 USE CASE IN ODP IN UML

With this background, I'll expand on the answer given at the beginning.

A particular use case of a particular system is a part of the community contract of a community of a certain type. I call a community of that type a 'use case community.'

That use case community is modeled as a UML collaboration. Let's call a collaboration of that type a 'use case collaboration.'

The roles in that collaboration are of two kinds, one or more actor roles and a single subject role.

For that collaboration, that system is represented as a single UML object. That object is bound to the subject role by a role binding. UML actors "type" the actor roles.

The objective of that community is the observable result of value of that use case (the value to one or more of the actors of that use case).

The contract of that community is specified in this way:

- the objective in a text note,
- the structure by the connectors and role bindings of the collaboration,
- the behavior by one or more of text, pre- and postconditions, or behaviors,
- the rules for the assignment of objects to roles by types and constraints,
- the other rules by multiplicities, types, and constraints.

The answer given at the beginning said that a particular use case of a certain system is a part of the community contract of a use case community. I wrote 'a part' to allow for readings of the UML specification that treat some of the contract as not being specified in the use case. On the other hand, if we read the specification as permitting us to include in the specification of the use case (as behavior(s), constraints, types, role bindings, and so on) all of the community contract, then that use case includes the entire contract of that use case community.

A UML enterprise viewpoint specification of that particular system includes a collaboration which is a composition [2-9.1b] of all the use case collaborations that include that system in the role, subject. Let's call that the 'combined collaboration' of the system. The way in which those use cases are combined is:

- the role of the system in that combined collaboration is the composition of the subject roles in each of those use case collaborations;
- the other roles in that combined collaboration are actor roles; each is a composition of actor roles of those use case collaborations;
- every actor role in those use case collaborations is included in at least one of the actor roles in that combined collaboration;
- the behavior of each actor in that combined collaboration is the composition of the behaviors of the roles of that actor in those use case collaborations;
- the contract of that combined collaboration is the conjunction (or other composition) of the contracts of those use case collaborations.

The specification of the subject roles and the actors roles may be the complete enterprise specification of the behavior of the system; if so, we can say that, from the enterprise viewpoint, the behavior of the system is completely specified by its use cases.

6 FURTHER WORK

Though I wrote that the behavior of an actor is the composition of the behaviors of the roles of that actor, I have not specified the way in which those behaviors are combined. [2-9.1b]

Readers familiar with the ODP enterprise language will have noticed that I have not attempted to discuss all the capabilities of the enterprise language that might be applied to use cases.

Readers familiar with the UML 2 specification of use case will have noticed that I've made no effort to tease out and attempt to exactify [Bunge03] each of the several theories of use case that are found when reading the UML 2 specification. Instead I've chosen texts from the UML specification that are consistent with a theory of use case that fits with the Reference Model of ODP. That suits my purpose, which is to explicate UML 2 use case from the ODP viewpoint.

Once the bugs in this first attempt are found and removed, it remains to exactify it. This can be done, for example, using the techniques of Part 4 of the Reference Model. [ODP4] Much of the work can be done using OCL [OCL] and the MOF (or UML profile) mechanism, by specializing UseCase, Collaboration, and so on.

REFERENCES

[ODP] ISO/IEC JTC1/SC21, *Open Distributed Processing—Reference Model: Foundations, Architecture, and Enterprise Language*, ITU-T Recommendations X.902, X.903, and X.911 | ISO/IEC 10746-2, 10746-3 and 15414. See http://www.joaquin.net/ODP.

[UML2] UML 2 Superstructure Specification http://ww.omg.org/cgi-bin/doc?ptc/2004-10-02

[Genilloud05] Guy Genilloud and William F. Frank, *Use Case Concepts from an RM-ODP Perspective*, in Journal of Object Technology, vol. 4, no. 6, Special Issue: Use Case Modeling at UML-2004 August 2005, http://www.jot.fm/issues/issue_2005_07/article9

[3C UML] *3C UML 2 submission*, 2002. http://ww.joaquin.net/cuml/3C-UML--2.14.pdf.

[UML1.5] *UML 1.5 Approved Specification*, 2003. http://ww.omg.org/cgi-bin/doc?formal/03-03-01

[Bunge03] Mario Bunge, *Philosophical Dictionary*, Amherst NY: Prometheus Books, 2003. ISBN 1-59102-037-9

[ODP4] ISO/IEC JTC1/SC21, *Open Distributed Processing—Reference Model: Architectural semantics*, ITU-T Recommendation X.904 | ISO/IEC 10746-4

[OCL] OCL Final Adpoted Specification, 2003. http://www.omg.org/cgi-bin/apps/doc?ptc/03-10-14.pdf

About the author

Joaquin Miller is a software architect. He contributed to UML 1 and UML 2 and is project editor of X.911. E-Mail: joaquin at acm dot org