2 CREATING AND WRITING TEXT USE CASE SCENARIOS
Practical 2

Table of Contents

1 The aim of the laboratory work is

Writing use case scenarios for an individual study project. (Написание сценариев вариантов использования для индивидуального учебного проекта.)

2. Script concept (понятие сценария)

It should be noted that one of the requirements of the UML language is the self-sufficiency of diagrams for the presentation of information about the models of designed systems. However, most developers and experts agree that the visual means of the UML language are clearly not enough to take into account in the use case diagrams the particularities of the functional behavior of a complex system. To this end, it is recommended to supplement this type of diagrams with text scenarios that refine or detail the sequence of actions performed by the system when executing its use cases.

A scenario is a specific sequence of actions that describes the actions of actors and the behavior of the modeled system in the form of plain text.

In the context of the UML language, the script is used to further illustrate the interaction of actors and use cases. Various ways of presenting or writing such scenarios are proposed. One of these templates is discussed below and can be recommended to students for use in the initial stages of conceptual modeling (Table 2.1).

Table 4.1. Template for writing a script of a separate use case

Main section	Section "Typical course of events"	Section "Exceptions"	Section "Notes"
Name of use case	Typical course of events leading to successful	Exception number	Notes No. 1

использования	execution of use-case	1	
Actors		Exception number 2	Notes No. 2
Purpose		...	...
Short description			
Type			
Reference to other use-case		Exception number N	Notes No. N

When writing use case scenarios, it is important to understand that the text of the scenario should complement or clarify the use case diagram, but not completely replace it. Otherwise, the advantages of visual representation of models will be lost.

Building a diagram of use cases is the very first stage of the process of object-oriented analysis and design, the purpose of which is to present a set of functional requirements for the behavior of the designed system. The specification of requirements for the designed system in the form of a diagram of use cases and additional scenarios can be an independent model, which in UML has been called the model of use cases and has its own special standard name or stereotype <<useCaseModel>>.

3 Example scripts of the use-case diagram

Example of writing scripts for an ATM management system.

To illustrate the features of the specification of functional requirements in the use case diagram, consider a model of a conventional ATM. The process of functioning of this system is well known to credit card holders, therefore, does not require additional description. The peculiarity of the considered ATM is that there is no possibility of transferring funds from one account to another. The system in question has two actors, one of whom is an ATM client, and the other is the "Bank", which carries out the relevant transactions. Each of these actors interacts with an ATM. Although the main actor "Client", since it is he who initiates the functionality of the ATM.

The main functional requirements for an ATM are to provide the customer with the possibility of withdrawing cash on a credit card and receiving a certificate of account status. It is these functional requirements that are specified by individual use cases, which serve as key elements of the corresponding conceptual model. Since it is necessary to authenticate a credit card to fulfill these use cases, they both apply to the credit card PIN code additional service. As follows from the essence of the functional requirements put forward to the ATM, this service can act as a separate use case of the chart being developed and be connected to the first two variants by the inclusion relation. The corresponding use case diagram may include only the two actors indicated and three use cases (Fig. 2.1).

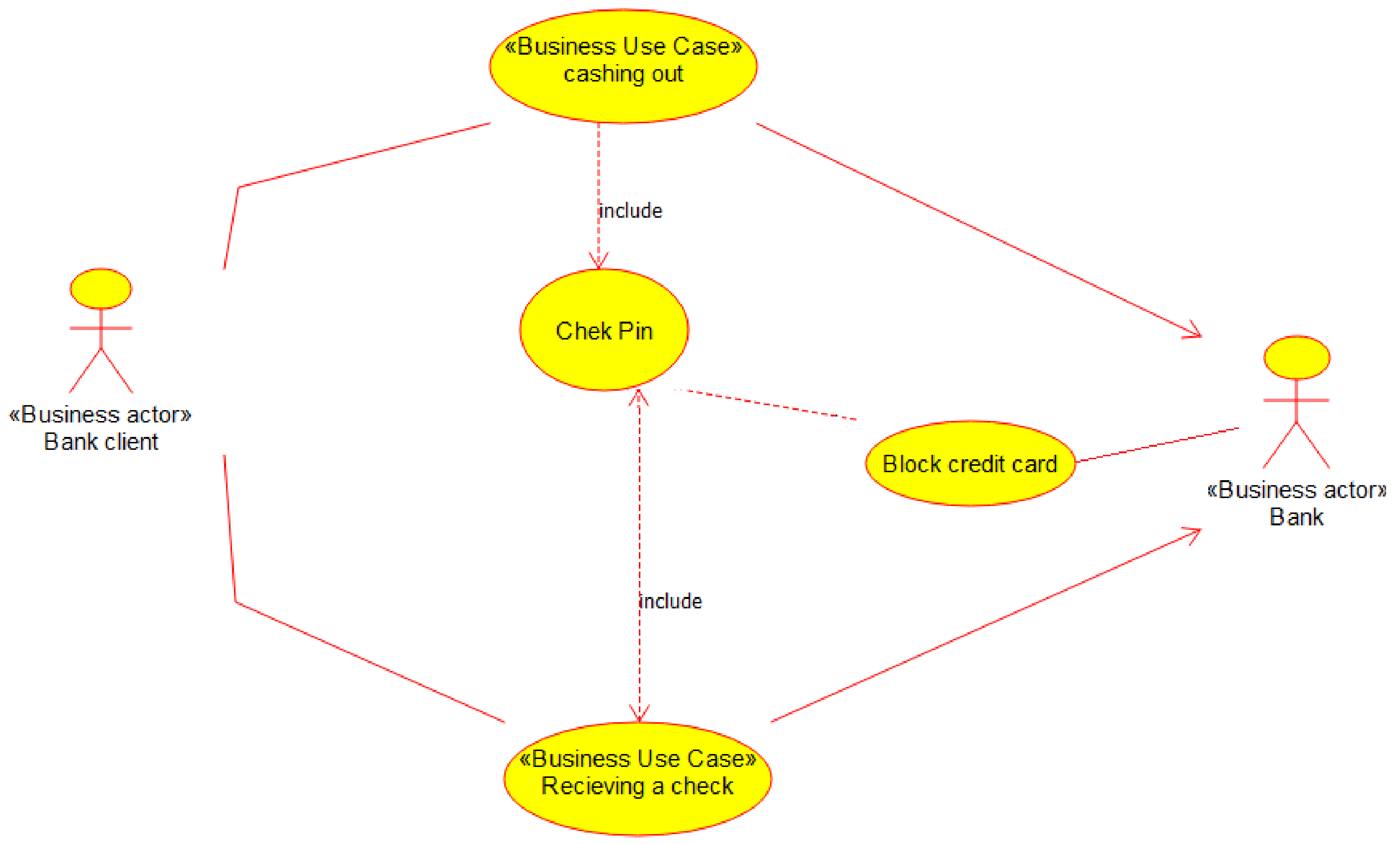
«Business Use Case» cashing out
include
Chek Pin
«Business actor» Bank client
Block credit card
«Business actor» Bank
include
«Business Use Case» Recieving a check

Fig. 2.1 – Chart of use options for the ATM model

At the next stage of development of the use case model for the considered ATM system, this chart should be supplemented with a text script written on the basis of the template proposed earlier (Table 2.1). This scenario will complement the diagram, revealing the content and logical sequence of individual actions that are performed by the system and the actors in the process of withdrawing cash on a credit card. In this case, the script is conveniently presented in the form of three tables, each of which describes a separate section of the template.

The main section of the script (Table 2.2) indicates the name of the use case in question, the names of the actors interconnected with it, the purpose of the option, the conditional type and references to other use cases.

Table 2.2. The main section of the scenario of the use of the "Cash withdrawal on credit card"

Use-case	Cash withdrawal on credit card
Actors	Client, Bank
Purpose	Obtaining the required amount in cash
Short description	The client requests the required amount. The ATM provides access to the customer's account. The ATM gives cash to the customer.
Type	Basic
Links to other uses	include: • Verify credit card PIN • Identify credit card

The next section of the scenario (Table 2.3) describes the sequence of actions leading to the successful implementation of the use case in question. In this case, the initiator of the action should be the actor Client. For the convenience of subsequent links, each action is labeled with a sequence number in the sequence.

Table 2.3. Section Typical course of events in the scenario of the implementation of the "Cash withdrawal on a credit card" use-case

Acting Actors	System Response
1. Customer inserts credit card into ATM reader	2. ATM checks credit card
Exception 1: Credit card is invalid	3. ATM offers to enter PIN
4. Customer enters personal PIN	5. The ATM checks the PIN code
Exception 2: The customer enters an incorrect PIN code	6. ATM displays menu options.
7. The client chooses a cash withdrawal from his account	8. The system makes a request to the Bank and ascertains the current status of the client's account
	9. ATM offers to enter the required amount.
10. Customer enters the required amount	

11. The bank checks the amount entered Exception 3: The required amount exceeds the amount on the client's account	12. The ATM changes the state of the client's account, issues cash and a check
13. The customer receives cash and a check	14 An ATM offers the customer to pick up a credit card
15. Customer receives his credit card.	16. ATM displays a message of readiness for work.

The third section of the script (Table 2.4) describes the sequence of actions performed in the event of exceptional situations or exceptions.

Table 2.4. Section Exclusions of the scenario of the "Cash withdrawal on a credit card" use-case

Exception 1: Credit card is invalid or incorrectly inserted.

Actor Response	System Actions
	3. The ATM displays information about an incorrectly inserted credit card. 14. ATM returns credit card to customer
15. Customer receives his credit card.	
Exception 2. Client enters wrong PIN	
	6. The ATM displays information about the wrong PIN.
4. Customer enters a new PIN	
Exception 3. The required amount exceeds the amount on the client's account	
	12. ATM displays credit overrun information.
10. Customer enters new required amount	

You can supplement this scenario by describing in a similar way not only the use options of “Getting account information certificate” and “Checking a credit card PIN code”, but also considering other exceptions, for example, the client’s refusal to receive cash after checking a PIN code, etc. . At the same time, the completeness of scenarios and models of use options will be determined by the functional requirements that are formulated within the framework of a specific project for the development of the corresponding ATM.

Separate small-sized scenarios can be placed on the diagram in the form of notes.

3 Recommendations for the development of use-case diagrams

As noted earlier, one of the main purposes of the use-case diagram is to formalize the functional requirements of the system. The use case diagram can serve as a basis for agreeing with the customer on the functional requirements for the system at an early design stage. Any of the basic use cases may subsequently be decomposed into private use cases. It is recommended that the total number of actors in the model does not exceed 20, and the use cases - 50. Otherwise, the model loses its visibility and, possibly, replaces one of some other diagrams.

Some steps are recommended for developing a use case diagram:

Identify major or primary and minor actors.

Determine the goals of the main actors in relation to the system

Formulate basic use cases that specify system functional requirements.

Sort the use cases by decreasing risk of their implementation

Consider all basic use cases in descending order of risk.

Select participants, interests, preconditions and postconditions of the selected use case

Write a successful implementation scenario for the selected use case.

Determine exceptions or failures in the use case scenario

Write scripts for all exceptions

Highlight common uses and portray their relationship with the base with the stereotype << include >>

Select the use cases for exceptions and depict their relationships with the base ones with the << extend >> stereotype

Check the diagram for duplication of use cases and actors

The semantics of constructing a chart of use cases should be determined by the following features of the model elements discussed above. A separate instance of the use case is in its content the execution of a sequence of actions that is initialized by means of an instance of a message from an actor instance. As a response or response to the message of the actor, the sequence of actions established for this use case is performed. In this case, the actors can generate new messages to initiate use cases.

This interaction will continue until the required sequence of actions is completed by the use case instance, and the actor instance specified in the model receives the required service instance. The end of the interaction means the lack of initialization of messages from the actors for basic use cases.

Use cases can be additionally specified with text notes, which can later become prototypes of operations and methods along with attributes. Further development of models is associated with the implementation of use cases in the form of an activity graph, by means of a finite automaton or any other mechanism of logical representation of behavior, including preconditions and postconditions. The interaction between use cases and actors can be refined in a collaboration diagram when the relationships between the system containing these use cases and the environment or external environment of the system are described.

In the case when subsystems are used to represent the hierarchical structure of the designed system, the system can be defined in the form of use cases at all levels. Individual subsystems or classes can implement their own use cases. In this case, the most common or abstract use case can be further clarified by a number of particular use cases, each of which will determine the service of the model element contained in the service of the source system. In this context, a generic use case can be considered as a super service for specifying use cases, which, in turn, can be considered as sub-services of the initial use case.

Separate variants of using the lower level can participate in several cooperations, i.e. play a certain role in the performance of services of several variants of the upper level. For individual such cooperatives, the respective roles of actors interacting with specific options for using the lower level can be defined. These roles will be played by actors from the lower level of the system model. Some of these actors may be top-level actors, but this does not contradict the semantic rules for constructing use-case diagrams adopted in UML.

However, it should be noted that the structure of a container element cannot be represented by use cases, since they can only represent the functionality of individual model elements. Subordinate use cases can only cooperate to jointly execute the super service of the top-level use case. This cooperation is also represented on the cooperation diagram in the form of joint actions of individual elements of the model.

The low-level use case environment is a separate element of the model, which, in turn, may contain other model elements defined for these use cases. Thus, from the point of view of a general representation of the conceptual model, the interaction between the lower-level use cases determines the result of performing the service of the top-level option.

The implementation of the use case depends on the type of the model element in which it is defined. For example, options for using a simulated software system can be implemented through model class operations. For business systems, use cases can be implemented by employees of this system. In all cases, the elements of the system must interact with each other to jointly ensure the required behavior and the use of the model.

If the subsystem of the lower level acts as a modeled entity, then individual users of the use cases of this subsystem are modeled by actors. Such actors, in turn, may be internal employees in relation to the modeled systems of the upper levels, which are often not explicitly indicated on the diagrams. These model elements may be contained in other packages or subsystems. In the latter case, the roles of the actors are defined in the package to which the corresponding subsystem belongs.

If the use cases are used for the specification of a part of the system, then they will be equivalent to the corresponding use cases in the subsystem model of the corresponding package. It is important to understand that all the functional requirements for the system must be explicitly specified in the use case diagram, and no other services that are missing in the given diagram can be designed by the system by definition. Moreover, if several models are used to simulate the behavior of a system at once, then the set of options for using all packages of the system should be equivalent to the set of options for using the model as a whole.

Контрольные вопросы:

1. Предназначение диаграммы вариантов использования?

2. Какие основные сущности Используются при построении ДВИ?

3. Какими отношениями могут быть связаны сущности диаграммы….7

Список литературы

1. Леоненков А.В. Самоучитель UML. 2-е издание - СПб.: "БХВ-Петербург", 2004. - 432 с.

2/ Буч Г., Джекобсон А., Рамбо Дж. Язык UML. Руководство пользователя - М.: ДМК, 2000. - 432 с.
3/ Буч Г., Рамбо Дж., Якобсон А. UML: специальный справочник - СПб: "Питер", 2001. - 656 с.

1/ Леффингуэлл Д., Уидриг Д. Принципы работы с требованиями к программному обеспечению. Унифицированный подход - М.: "Вильямс", 2002. - 448 с.

----

* Starr L. Executable UML. How to build class models - Prentice Hall PTR, 2002. - 418 p.
* Wampler B.E. The Essence Object-Oriented Programming with Java and UML - Addison-Wesley, 2002, - 290 p.