Tomasz Woch, Eng.
The School of Banking and Management in Krakow
tomasz.woch@gmail.com
Małgorzata Żabińska, Ph.D., Eng.
The School of Banking and Management in Krakow
gosia.zabinska@gmail.com
Janusz Majewski, Ph.D., Eng.
The School of Banking and Management in Krakow
kr951fc@gmail.com

REQUIREMENT DESCRIPTION IN DEDICATED IT SYSTEM DEVELOPMENT

Introduction

A goal without a plan is only a wish – as it was said by Antoine de Saint-Exupéry, a French writer, poet and pilot. Every project, even when insignificant in size, requires a development of an adequate plan. In the development of an IT system that faces real world issues we immediately realize the complexity of the task. Problems and contradictions multiply with each step and the level of detail forces us to verify the assumptions. It quickly becomes clear that a plan, a kind of a roadmap is necessary to accompany us at every decision made and helps not to go off the course. In case of IT systems, the roadmap is constituted by the requirement description.

A proper requirement description is the key to success in developing dedicated IT systems and it should be related to each step of the project development. It should not be a static body but it should grow and change in different phases of the system development process. User environment and needs tend to change in time. Therefore, their description should also be iteratively verified and detailed in the course of system development. Such an approach to design and development of dedicated IT systems results in the decrease in cost and reduction in working time. It makes agile reaction possible and it also enables adjusting to unexpected constraints.

There are numerous methods of preparing requirement descriptions and each one has its advantages and disadvantages. Some descriptions present user needs in a better way, other show information flow processes more precisely while there are some that make it possible to understand at a glance the architecture of the whole project. From the range of the available solutions, we should choose the ones that suit best particular stages of work and the class of the system under development.

1. User Stories

The first thing that comes to mind is to describe the requirements in a natural language, i.e. the one we use on a daily basis. The use of an unsystematized and ambiguous language results in errors in software development that may prove costly to remove and may cause a delay. In order to maintain the flexibility and simplicity of a natural language and to eliminate some errors, a certain convention of requirement notation can be assumed.

User Story is a way of formulating requirements that comes from agile methodologies[1]. User Stories are usually written in a simple language that is comprehensible to every individual involved in the project. They usually follow a simple scheme:

As a <role>**, I want** <goal/desire> **so that** <benefit>[2].

User Stories can also be extended by acceptance criteria and test scenarios. Acceptance criteria refer to a list of requirements that must be met by a project to consider a given user story complete. Test Scenarios are the starting point for a repetitive and measurable system testing and they constitute the basis for acceptance, single, integrating and system tests. An example of a user story is given in Table 1 below.

The engineering thesis[3] concerns analysis of requirements, development of a project and implementation of a system prototype to support the management of a vehicle fleet. Table 1 presents an example of an initial requirement description from the point of view of both a dispatcher and a driver. The presented User Story concerns the need to have adequate vehicle records for both types of users.

User stories are a perfect way for initial grasping of the needs of users but they do not inform in detail how they interact with the system and there is a large interpretation space for implementation.

[1] S. Allen, *Data Modeling for Everyone*, Curlingstone, Birmingham, 2002.
[2] A. Cockburn, *Writing Effective Use Cases*, Addison-Wesley Professional, Boston 2000.
[3] T. Woch, *System to support vehicle fleet management,* Engineering thesis, WSZiB Kraków, 2020.

Table 1. Exemplary User Story

Specification	User story	Acceptance criteria		Test scenarios
		Functional	Nonfunctional	
Vehicle records	**As** a user {driver, dispatcher} **I want** to {display a list of my vehicles, move the data of the selected vehicle to the file (register)} **so that** I can {choose the vehicle, update the data}	• Does each vehicle in the table have its identification number, the driver's name and the status? • Can the driver see in the table only the vehicles that are assigned to him? • Can the dispatcher see all the vehicle in the table?	• Is the list of vehicles displayed in less than 3 seconds?	**Scenario 1:** Vehicle records for the dispatcher **If** the dispatcher is logged in, **when** moving to the display of vehicle records, **then** the list will include all the vehicles. **Scenario 2:** Vehicle records for the driver **If** the driver is logged in, **when** moving to the display of vehicle records, **then** the list will include only the vehicles that are assigned to this driver.

Source: T. Woch, *System to support vehicle fleet management,* Engineering thesis, WSZiB Kraków, 2020.

In order to avoid that trap and the ambiguity, tools can be applied that are provided by the Unified Modeling Language (UML). Modeling languages consist of notations that define system elements and semantics that describes the relationships between them. One of the methods to display the requirements with the use of UML are Use Case Diagrams. They are supported by Use Cases and only a set consisting of Use Case Diagrams and scenarios describing Use Cases from the diagram give a complete picture of requirements.

2. Use cases

Use case is a description of interaction scenarios between the system under discussion and its external actors, related to the particular goal[4]. Scenario descriptions may have various forms but their common feature is the fact that they show how the system reacts to the external world and what tasks it should perform, without getting into the activities of individual parts.

Use cases do not only define how the system should behave but they also help find requirement loops at early stages of a project[5]. When creating use cases, we define at the same time their mutual relations, which gives a more complete picture of the whole system. Due to

[4] A. Cockburn, *Writing Effective Use Cases*, Addison-Wesley Professional, Boston 2000, p. 15.
[5] R. Miles, K. Hamilton, *Learning UML 2.0,* O'Reilly Media, Inc., Cambridge 2007, p. 29.

the formalization of the process it is possible to answer several basic questions: What has caused the particular behavior? What is the proper functioning of the system in a particular situation? What alternative and exceptional situations must be predicted? What should the system state be like after the presented event? The exemplary notation in Table 2 may be used to present the answers to the above questions.

Table 2. Exemplary use case scenario

EP/01	Vehicle records management
Priority:	1
Actors:	Dispatcher
Initial conditions:	Dispatcher is logged in.
Trigger:	Click on navigation tab.
Main scenario:	1. The system displays a table with a list of vehicles sorted by default by the date of the latest modification and limited to 10 lines. The table includes such fields as: • side (identification) number, • registration number, • status (active, serviced, etc.), • driver's first and second name, • the latest modification date. 2. The user may move to the next page through pagination. Active page is in bold.
Exceptional situations:	The table is empty when no vehicle is available.
Final conditions:	The system displays information from vehicle register. Possible operations: addition: (EP/04), modification (EP/05) and deletion (EP/06).
Relationships	EP/04, EP/05, EP/06

Source: Based on T. Woch, *System to support vehicle fleet management,* Engineering thesis, WSZiB Kraków, 2020.

There are numerous methods of the use case presentation. Here, a simplified form was used which includes use cases features that are most frequently applied. Each use case should have a unique identifier and name. When defining a use case, a priority can be assigned to it that informs the system developers which functionalities are critical to the end user. *Actors* are external entities that interact with the system. Most frequently, they are the users but they may also represent other systems or organizational units. *Initial conditions (Preconditions)* refer to the state of the system before interactions have started, while *trigger* is an action or event that initiates a use case. *Use case* in the system is described here in the form of a scenario, i.e. a sequence of steps that lead to the completion of a use case. The state of the application after completing the scenario by the system is referred to as *final conditions* and an alternative course

of action is referred to as *exceptional situations*. Finally, mutual relationships between use cases should be indicated – here unique use case identifiers are used.

3. Graphical modeling of requirements - Use Case Diagrams

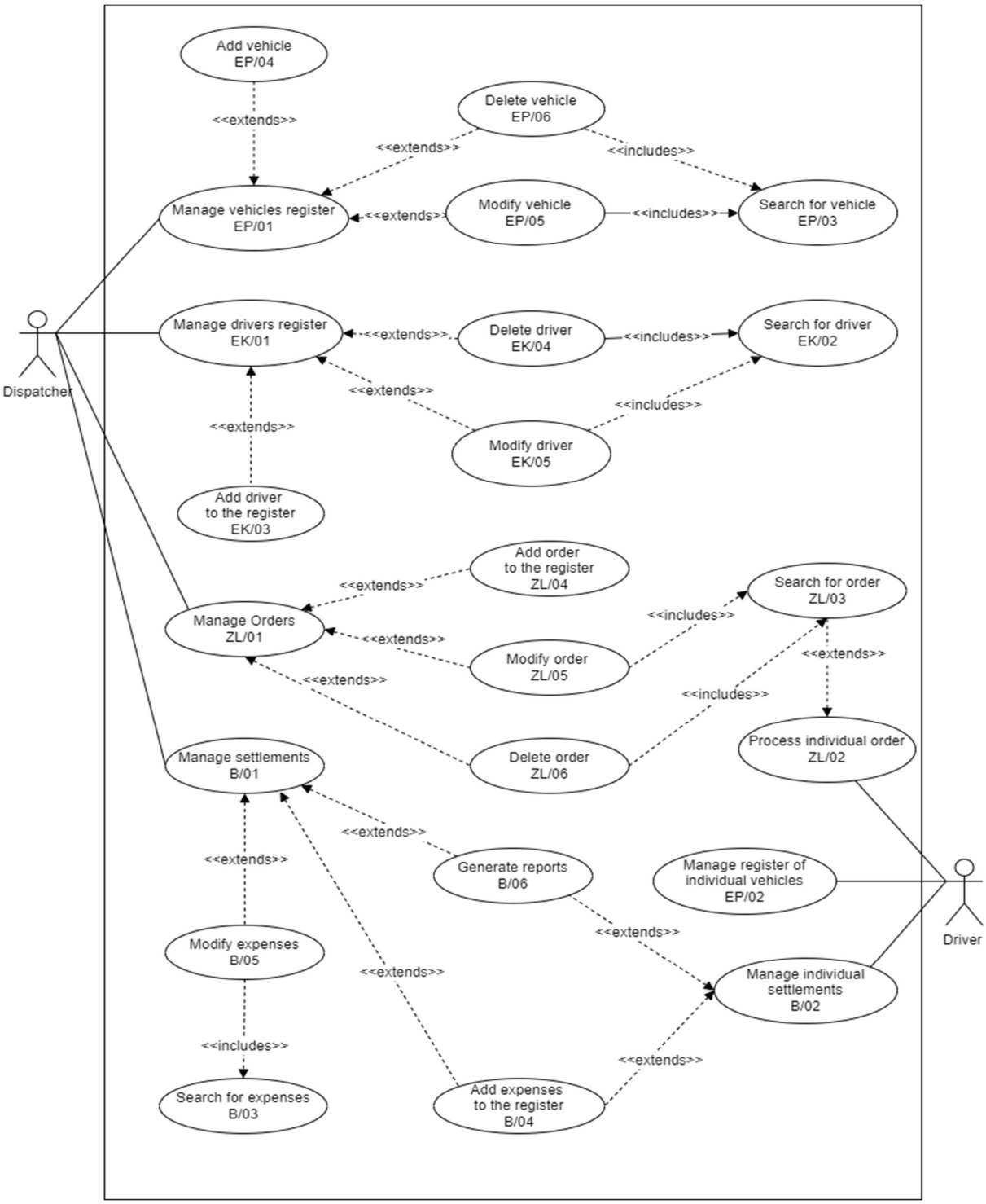
Presenting lists of identifiers of related use cases in a table (cf. Table 2) is not a clear way to present their relationships; to illustrate the relationships a graphical form can be used with the application of a Unified Modeling Language (UML) notation. This form of presenting use cases and their relationships is referred to as the Use Case Diagram [6,7].

Use case diagrams present graphical relationships between system actors and the scenarios used by them. Moreover they show visual relationships between scenarios themselves, e.g. <<include>>, << extend>>.

An example of such a use case diagram for a vehicle fleet management system[8] is given in Figure 1. The diagram shows that use cases to support the work of dispatchers and drivers are grouped in three domain areas: management of registers (both of vehicles and drivers), orders management and settlements management. Such a natural way of separating the parts that are dedicated to the actors (who represent future users of the system) provides the opportunity to organize the needs and ensures a proper approach to the further steps of the system development, i.e. designing (including the design of the graphical interface), implementing and testing (acceptance tests).

[6] M. Fowler, *UML distilled: A brief guide to the standard object modeling language*, Pearson Education Inc., Boston 2004.
[7] C. Larman, *UML i wzorce projektowe*, Helion, Gliwice, 2011.
[8] T. Woch, *System to support vehicle fleet management,* Engineering thesis, WSZiB Kraków, 2020.

Figure 1. Use case diagram for a vehicle fleet management support system



Source: Based on T. Woch, *System to support vehicle fleet management,* Engineering thesis, WSZiB Kraków, 2020.

4. Actions modeling – Activity Diagrams

While use cases answer the question *what* the system should do, activity diagrams [9] give the answer to the question *how* it should be done. Such models, which create workflows in responsibility areas, make it possible to illustrate the behavior of the key system elements and show the control and object flow in the system. They are useful particularly in business process modeling. Such processes are sequences of tasks that result in the completion of a business objective, e.g. of an order or assignment.

Activity Diagrams correspond to the main domain areas of the system and illustrate basic processes that occur within them. Modeling experts consider them to be some of the best diagrams in UML[10] as they have substituted the older data flow modeling methods such as Data Flow Diagrams (DFD) mentioned below in Chapter 5.

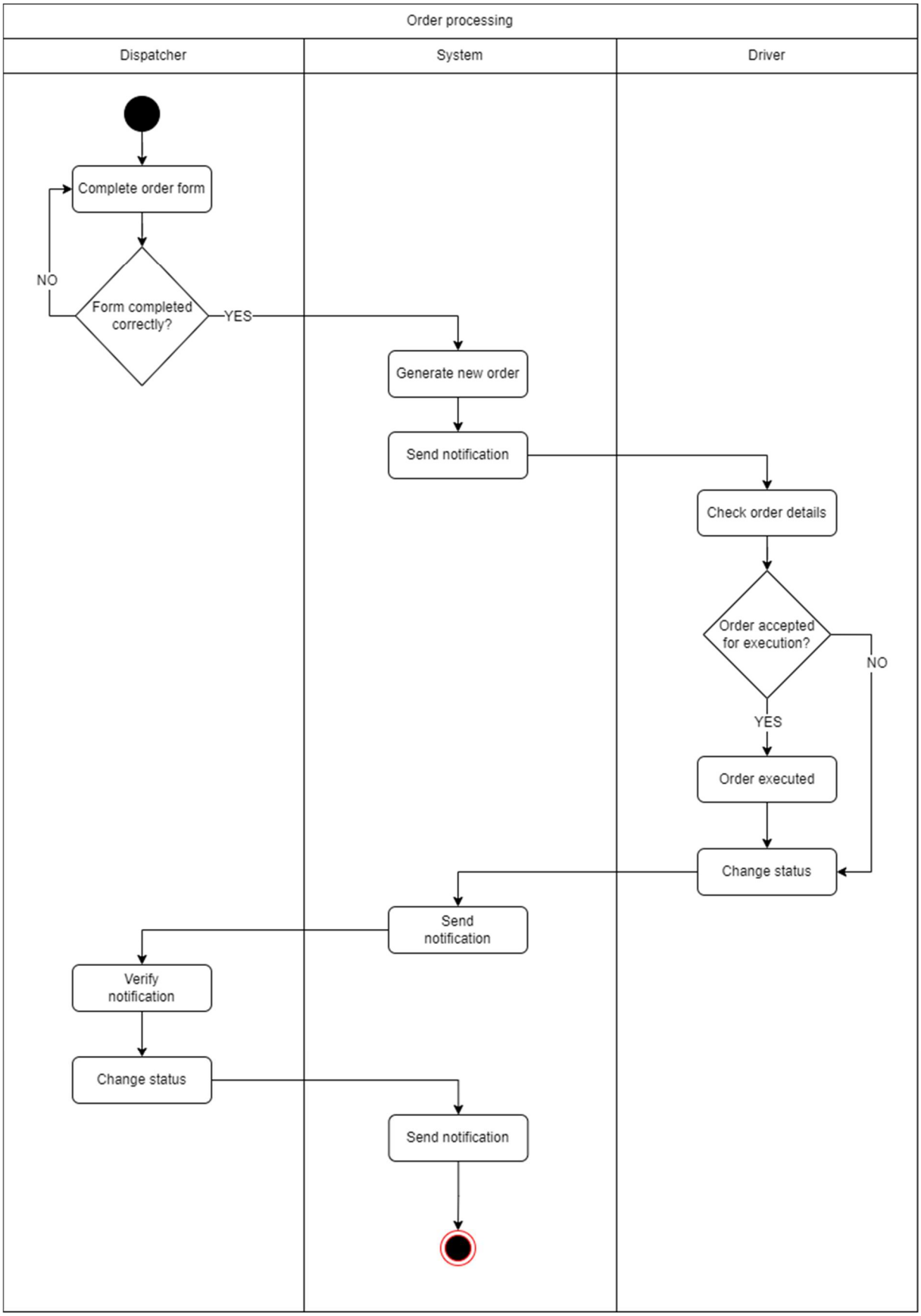
Activity diagrams make it possible to create specific requirement models that concern the behavior of the objects of interest. They present the sequence of actions as well as alternative ones and also parallel flows. The actions are carried out in line with the proposed use cases and with the vision of interactions between system elements and the actors present in models with use case diagrams. The diagrams are drawn with the use of partitions or swim-lanes and they may present behaviors within responsibility areas of particular system's fragments or actors.

An exemplary activity diagram taken from the engineering thesis[11] and given in Figure 2 shows how an order from Dispatcher is sent to Driver. It presents various interactions with the system and points where the system performs particular tasks (order creation and modification, change of the expense status).

The diagram also presents the scope of responsibilities of actors who participate in the interactions with the system by dividing the diagram into three partitions. Each of them includes a sequence of actions related to the modeled behavior of one of the three actors: Dispatcher, Driver and the System.

[9] R. Miles, K. Hamilton, *Learning UML 2.0,* O'Reilly Media, Inc., Cambridge 2007, p. 51.
[10] C. Larman, *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development*, Pearson Education Inc., Boston 2005.
[11] T. Woch, *System to support vehicle fleet management,* Engineering thesis, WSZiB Kraków, 2020.

Figure 2. Activity diagram of transport order processing



Source: Based on T. Woch, *System to support vehicle fleet management,* Engineering thesis, WSZiB Kraków, 2020.

5. Other methods of requirement presentation

There are other older methods of description and presentation of functional requirements that are based on structural approach and were commonly used in the 1970s and the 1980s[12]. They are based on Data Flow Diagrams (DFD) and functional domain decomposition in line with Dijkstra's concept of abstract level modeling. However, this way of requirement presentation was troublesome particularly for more complex requirements and multi-domain scopes related to several groups of users. Multi-level complex models were created that were difficult to interpret, especially after the introduction of several modifications on various levels, and completely unclear for future users; as a result their real and useful verification was impossible or at least very difficult to conduct.

They were accompanied by a data requirement description in the form of conceptual data model using Entity Relationship Diagrams (ERD). The latter ones have stood the test of time and are still applied in the development of relational databases; however object models, especially class and object diagrams that are present in UML, evolved out of them as well as the entire object approach to system analysis and development.

A very interesting method of requirement modeling was proposed by the authors of an article on Model-Driven Development[13]. This is an hybrid approach that shares the elements of structural and object modeling with a general requirement description by Use Case Diagrams that is independent of the methodology and uses universal UML elements. It would be interesting to assess the effectiveness of this approach in requirement modeling that uses data flow diagrams and neglects some object mechanisms.

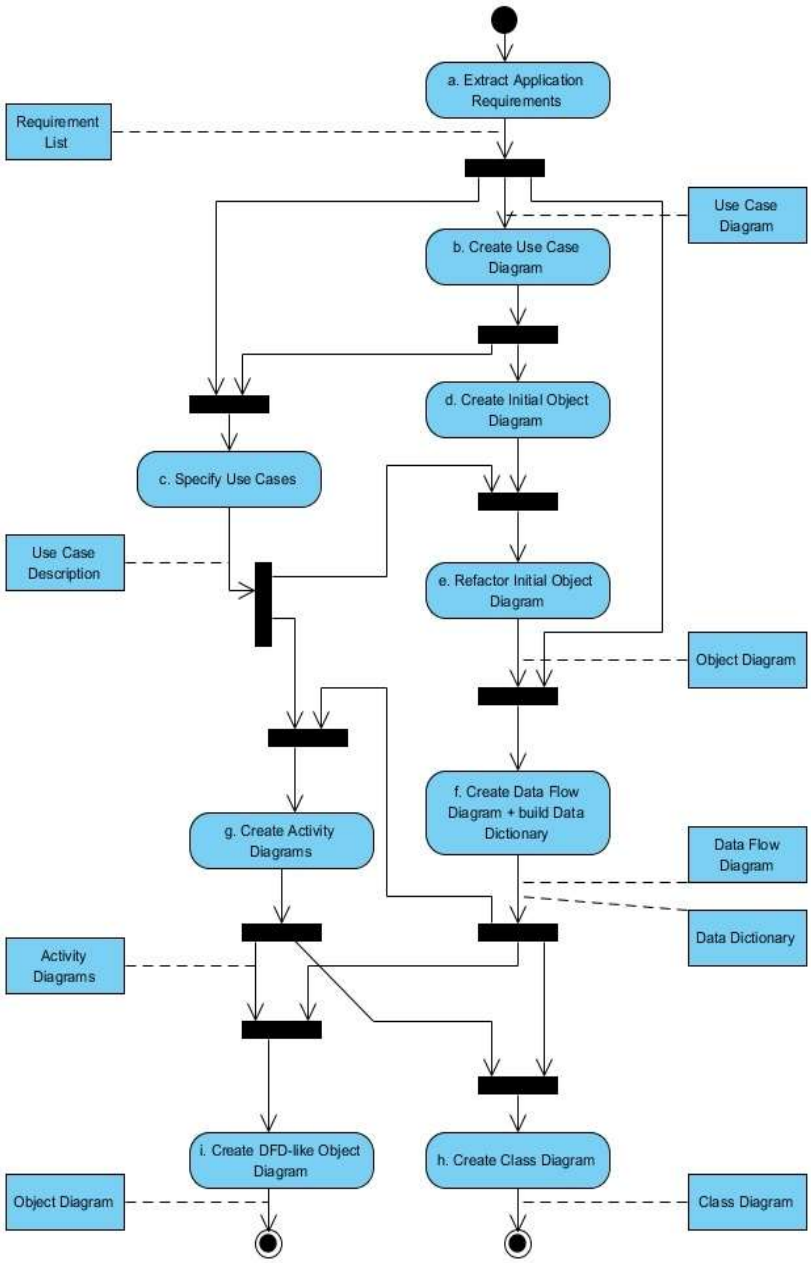
A combination of structural and object approach with the consideration of a universal requirement description with the use of Use Cases (scenarios) and Use Case Diagrams modeling is given in Figure 3 (following the authors of the mentioned above article). The combination takes into account the following sequence of actions:

- Requirements, business analysis
- Functions → *Use Cases*: diagrams + scenario description
- *Initial Object Diagram* (IOD) based on *Use Case Diagram*
- Refactoring IOD based on *Use Cases* (scenarios)

[12] E. Yourdon, *Modern Structured Analysis*, Prentice Hall, Inc., Englewood Cliffs NJ 1988.
[13] M. Alanen, J. Lilius, I. Porres, D. Truscan, *On Modeling Techniques for Supporting Model-Driven Development of Protocol Processing Applications*, [in:] S. Beydeda, M. Book, V. Gruhn (eds.): *Model-Driven Software Development,* Springer-Verlag, Berlin, Heidelberg, 2005.

- Transforming IOD → DFD and building *Data Dictiona*ry (DD)
- Specifying the internal behavior of the system in the form of *Activity Diagrams*
- Transforming DFD into *Class Diagram*
- Transforming DFD into *DFD-like Object Diagram*

Figure 3. *Activity Diagram* illustrating a sequence of actions in a combined approach to requirement modeling in IT system development



Source: M. Alanen, J. Lilius, I. Porres, D. Truscan, *On Modeling Techniques for Supporting Model-Driven Development of Protocol Processing Applications*, [in:] S. Beydeda, M. Book, V. Gruhn (eds.): *Model-Driven Software Development,* Springer-Verlag, Berlin, Heidelberg, 2005, p. 319.

Conclusions

The article presented common methods of requirement description that are applied in IT systems development, particularly the ones that are dedicated to specific uses.

It mentioned the ways of requirement description that are used in the initial steps of system analysis (e.g. descriptions in natural languages as the results of business analysis). It presented the method of requirement description (User Stories) which is currently applied in agile methodology, is perfect for creating descriptions at early development stages and makes it possible to confront the created vision and description following the presented template with the users' needs. According to Alistair Cockburn[14], the initial requirement vision in the form of *User Stories*, when extended and supplemented, is close to scenarios that are used to formulate precisely the needs and it embodies the "concept of convergence" in the area of requirement engineering. Scenarios descriptions are used either in visual modeling with UML as Use Case Diagrams or they supplement the models by defining the details of use cases that appear in diagrams (Use Case specification).

According to A. Cockburn's opinion presented above, *User Stories* should be used to describe initial visions of business requirements at early stages of analysis. The extended version of *User Stories* transformed into proper scenarios should be applied at a later stage when a specific concept of system requirements is developed. This makes the basis for development of the requirement graphical model employing UML (*Use Case Diagrams*).

The next step of analysis and development from the functional point of view is to build a realization model with the use of *UML Activity Diagrams* that present the course of procedures (as sequences of actions) for the domain areas on the basis of the proposed *Use Cases* and *Use Case Diagrams*.

All the methods of requirement description presented above are extremely useful to analysts and IT developers. Each one can be used at the appropriate stage of requirement analysis and development. It should not be forgotten that the better understanding of the requirements of the future system users, the better quality of the product delivered to the customer (i.e. the ultimate end-user) will be achieved. A sound knowledge of various methods of requirement description and the ability to apply them selectively considering complementarity especially in the case of systems dedicated to specific applications is the key to mastering the requirements by analysts and developers.

[14] A. Cockburn, *Writing Effective Use Cases*, Addison-Wesley Professional, Boston 2000.

Bibliography

[1] Alanen M., Lilius J., Porres I., Truscan D., *On Modeling Techniques for Supporting Model-Driven Development of Protocol Processing Applications*, [in:] Beydeda S., Book M., Gruhn V. (eds.): *Model-Driven Software Development,* Springer-Verlag, Berlin, Heidelberg, 2005.
[2] Allen S., *Data Modeling for Everyone*, Curlingstone, Birmingham, 2002.
[3] Cockburn A., *Writing Effective Use Cases*, Addison-Wesley Professional, Boston 2000.
[4] Fowler M., *UML distilled: A brief guide to the standard object modeling language*, Pearson Education Inc., Boston 2004.
[5] Larman C., *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development*, Pearson Education Inc., Boston 2005.
[6] Miles R., Hamilton K., *Learning UML 2.0, O'Reilly Media, Inc.,* Cambridge 2007.
[7] Woch T., *System to support vehicle fleet management,* Engineering thesis, WSZiB Kraków, 2020.
[8] Yourdon E., *Modern Structured Analysis*, Prentice Hall, Inc., Englewood Cliffs NJ 1988.
[9] Żabińska M., *Information systems*, Lectures given in summer semester 2018/2019 in WSZiB Kraków.

Abstract

The article deals with the main issue of requirement engineering, i.e. the method of presenting functional requirements in the development of IT systems, particularly the ones that are dedicated to a specific application. On the basis of the examples from the engineering thesis[15] that describe development process of an IT system supporting the vehicle fleet management, the authors presented various current methods of requirement description: user stories, simple scenarios, scenarios extended to use cases and UML-based methods. Graphical methods: use case diagrams, activity diagrams are also illustrated by the examples from the engineering thesis. Moreover, the authors mentioned some older methods such as data flows and a relatively new hybrid method based on them. In conclusion, the authors indicated the need to adjust the method or the combination of methods of requirement description to a particular stage of the IT system development process.

Key words

Requirement modeling, UML, use case, use case diagram, activity diagram, hybrid approach.

[15] T. Woch, *System to support vehicle fleet management,* Engineering thesis, WSZiB Kraków, 2020.