Computing and Informatics, Vol. 29, 2010, 27–44

BUILDING BENCHMARKS FOR USE CASES

Bartosz Alchimowicz, Jakub Jurkiewicz
Mirosław Ochodek, Jerzy Nawrocki

Institute of Computing Science
Poznan University of Technology
ul. Piotrowo 2
60-965 Poznan, Poland
e-mail: {balchimowicz, jjurkiewicz, mochodek, jnawrocki}@cs.put.poznan.pl

Revised manuscript received 16 October 2009

Abstract. This paper presents how the use-cases benchmark has been built and how it can be applied by researchers. Set of 16 industrial projects (with 524 use cases in total) has been analysed in order to obtain quantitative and qualitative profile of a typical use-case-based requirements specification. Based on the analysis, two types of referential use-case-based requirements specifications have been created, one taking only quantitative data into account and second considering qualitative data. Researchers who analyse use cases can utilise these specifications in order to validate their methods and tools before applying them to real projects. Moreover, such referential specification can be used as a benchmark and allows comparing accuracy and efficiency of tools for use-case analysis.

Keywords: Use cases, requirements, metrics, benchmark, quality

Mathematics Subject Classification 2000: 68N01, 68N30

1 INTRODUCTION

Functional requirements are very important in software development. They impact not only the product itself but also test cases, cost estimates, delivery date, and user manual. One of the forms of functional requirements are use cases introduced by Ivar Jacobson about 20 years ago. They are getting more and more popular in software

industry and they are also a subject of intensive research (see e.g. [4, 5, 10, 11, 22]). Ideas presented by researches must be empirically verified (in the best case, using industrial data) and it should be possible to replicate a given experiment by any other researcher [21]. In case of use cases research it means that one should be able to publish not only his/her results but also the functional requirements (use cases) that have been used during the experiment. Unfortunately, that is very seldom done because it is very difficult. If a given requirements specification has a real commercial value, it is hard to convince its owner (company) to publish it. Thus, to make experiments concerning use cases replicable, one needs a benchmark that would represent use cases used in real software projects.

In the paper an approach to construct use-cases benchmark is presented. The benchmark itself is a set of use-cases-based requirements specifications, which have a typical profile observed in requirements coming from real projects. It includes both quantitative and qualitative aspects of use cases. To derive such a profile an extensive analysis of 524 use cases was performed.

The paper is organised as follows. In Section 2 a model of use-cases-based requirements specification is presented. This model is further used in Sections 3 and 4, which present the analysis of the use cases, coming from 16 projects. Based on the analysis quantitative and qualitative profile of the typical use-case-based specification is derived, which is used to create a benchmark specification in Section 5. Finally, a case study is presented in Section 6, which demonstrates a typical usage of the benchmark specification to compare tools for the use-case analysis.

2 BENCHMARK-ORIENTED MODEL OF USE-CASE-BASED SPECIFICATION

Although use cases have been successfully used in many industrial projects (Neil et al. [18] reported that 50 % of projects have their functional requirements presented in that form), they have never been a subject of any recognisable standardisation. Moreover, since their introduction by Ivar Jacobson [15] many approaches for their development were proposed. For example, Russell R. Hurlbut [13] gathered fifty seven contributions concerning use cases modeling. Although, all approaches share the same idea of presenting actor's interaction with the system in order to obtain his goal, they vary in level of formalism and in presentation form. Thus, it is very important to state what is understood by the term use-cases-based requirements specification.

To mitigate this problem a semi-formal model of use-cases-based requirements specification has been proposed and presented in Figure 1. It incorporates most of the best-practices presented by Adolph et al. [2] and Cockburn [7].

However, it still allows to create specification that contains only scenario (or non-structured story) and actors, which is enough to compose a small but valuable use-case. This specification can be further extended with other elements like extensions (using steps or stories), pre- and post-conditions, notes or triggers.

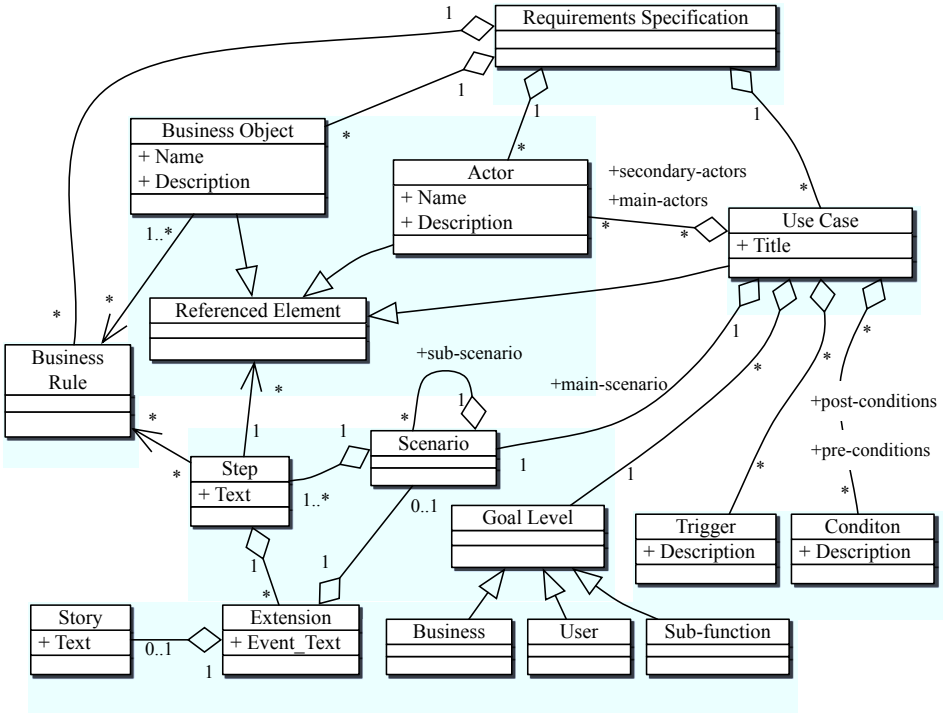

Fig. 1. Use-cases-based functional requirements specification model

Unfortunately, understanding the structure of specification is still not enough to construct a typical use-cases-based specification. What is missing is the quantifiable data concerning the number of model-elements and proportions between them[1]. In other words, one meta-question has to be asked for each element of the model – "how many?"

Another aspect is the content of use cases. In this case what is required is knowledge about the language structures that people use to author use cases. This includes also a list of typically made mistakes.

3 ANALYSIS OF UC-BASED SPECIFICATIONS STRUCTURE

In order to discover the profile of a typical specification, data from various projects was collected and analysed. As a result a database called UCDB (Use Cases Database) [8] has been created. At this stage it stores data from 16 projects with the

[1] A number/proportion of any element of the model will be further addressed as a *property* of the specification

total number of 524 use cases (basic characteristics of requirements specifications as well as short description of projects are presented in Table 1).

ID	Specification language	Origin	Number of use cases				Description
			All	Business	User	Sub-function	
Project A	English	S2B	17	0%	76%	24%	Web & standalone application for managing members of organization
Project B	English	S2B	37	19%	46%	35%	Web-based Customer Relationship Management (CRM) system
Project C	English	External	39	18%	44%	33%	UK Collaboration for a Digital Repository (UKCDR)
Project D	Polish	Industry	77	0%	96%	4%	Web-based e-government Content Management System (CMS)
Project E	Polish	S2B	41	0%	100%	0%	Web-based Document Management System (DMS)
Project F	Polish	Industry	10	0%	100%	0%	Web-based invoices repository for remote accounting
Project G	English	External	90	0%	81%	19%	Protein Information Management System (PIMS)
Project H	Polish	Industry	16	19%	56%	25%	Integration of two sub-system s in ERP scale system
Project I	Polish	Industry	21	38%	57%	5%	Banking system
Project J	Polish	Industry	9	0%	67%	33%	Single functional module for the web-based e-commerce solution
Project K	Polish	Industry	75	0%	97%	3%	Web-based workflow system with Content Management System (CMS)
Project L	English	External	16	0%	31%	69%	Polaris - Mission Data System (MDS) process demonstration
Project M	English	External	26	0%	23%	77%	Vesmark Smartware™ - Financial decission system
Project M	English	External	18	0%	0%	100%	Photo Mofo - Digital images management
Project O	English	External	16	0%	31%	69%	iConf - Java based conference application
Project P	English	External	16	0%	25%	75%	One Laptop Per Child - Web-based Content Management

Table 1. Overview of the projects and their requirements specifications (origin: industry – project developed by software development company; s2b – project developed by students for external organisation; external – specification obtained from external source which is freely accessible through the Internet; projects D and K come from the same organisation)

Specifications have been analysed in order to populate the model with the information concerning average number of its elements and additional qualitative aspects of use cases (see Section 4).

One of the most interesting findings is that 65.8 % of the main scenarios in use cases consist of 3–9 steps, which means that they fulfil the guidelines proposed by Cockburn [7]. Moreover, 72.1 % of the analysed use cases have alternative scenarios (extensions). There are projects in which use cases have extensions attached to all steps in main scenarios; however, on the average a use case contains 1 to 2 extensions (mean 1.57). Detailed information regarding the number and distributions of steps in main scenarios and extensions, is presented in Figure 2.

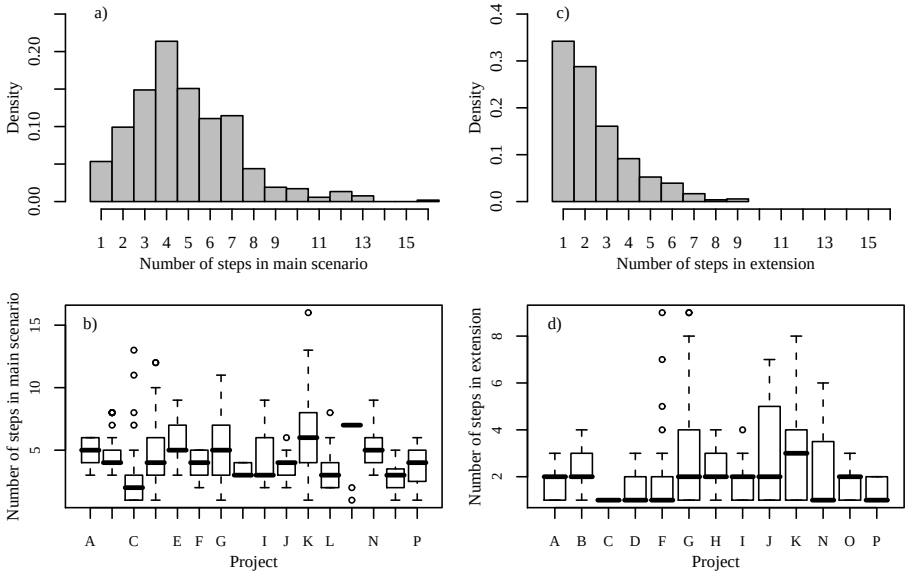

Fig. 2. Scenarios lengths in analysed use cases: a) histogram presents the number of steps in main scenario (data aggregated from all projects), b) box plot presents the number of steps in main scenario (in each project), c) histogram presents the number of steps in extension (data aggregated from all projects), d) box plot presents the number of steps in extension (in each project, note that project E was excluded because it has all alternative scenarios written as stories)

Another interesting observation, concerning the structure of use cases is that the sequences of steps performed by the same actor are frequently longer than one. What is more interesting this tendency is more visible in case of main actor steps (38.4 % of main actor steps sequences are longer than one step, and only 25.5 % in case of secondary actor – in most cases system being developed). One of potential reasons is that the actions performed by main actor are more important, from the business point of view. This contradicts the concept of transaction presented by Ivar Jacobson [14]. Jacobson enumerated four types of actions which together form the use-case transaction. Only one of them belongs to a main actor (user request action). The rest of them are system actions: validation, internal state change, and response. It might seem that those actions should frequently form longer sequences. However, 74.6 % of steps sequences performed by system consisted of a single step. The distributions and number of steps in main actor sequences are presented in Figure 3.

If we look deeper into the textual representation of the use cases, some interesting observation concerning their structure might be made. One of them regards the way how authors of use cases describe validation actions. Two different approaches are

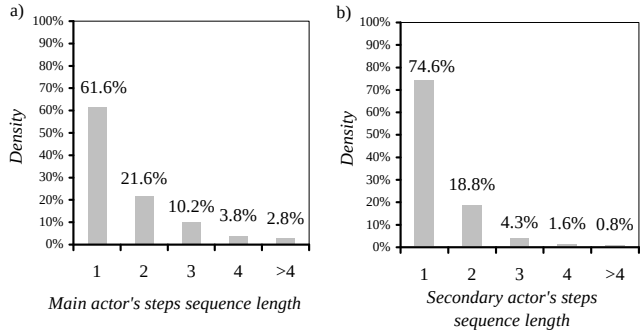

Fig. 3. Distributions of actors steps-sequences lengths in analysed use cases, a) histogram presents the length of the steps sequences performed by main actor, b) histogram presents length of the steps sequences performed by secondary actor – e.g. System

observed. The most common one is to use extensions to represent alternative system behaviour in case of failure of the verification process (41.3 % of extensions have this kind of nature). The second one, and less frequent, is to incorporate validation actions into steps (e.g. *System verifies data*). This kind of actions are observed only in 3.4 % of steps.

Analysed properties can be classified into two separate classes. The first one contains properties which are observed in nearly all of the projects, with comparable intensity. Those properties seem to be independent from the specification they belong to (from its author's writing style). The second class is an opposite one, and includes properties which are either characteristic for a certain set of use cases only, or their occurrence in different specifications is between two extremes – full presence or marginal/none. Such properties are project-dependent. More detailed description of analysed requirements specifications is presented in Table 2.

4 ANALYSIS OF USE-CASES-BASED SPECIFICATIONS DEFECTS

Use cases seem simple in their structure, and easy to understand; however, it is easy to introduce defects, which might influence the communiation between analyst, customer and project stakeholders. Different research [1, 19] indicated that vague, inconsistent or incomplete requirements are the main reasons for project failures. Defects make specification cumbersome, tedious and hard to read, analyse and change, hence they can lead to misunderstandings and finally to higher costs of software projects. Based on patterns and guidelines presented by Adolph et al. and Cockburn [2, 7] a set of defect types has been distinguished. Projects stored in UCDB have been analysed in order to collect statistical data about defects found in the requirements specification. Table 3 presents the results of this analysis.

Table 2. Results of the structure analysis of the use cases stored in UCDB

	Property		Overall	Project															
				A	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**I**	**J**	**K**	**L**	**M**	**N**	**O**	**P**
Requirements specification ***independent***	Number of use cases		524	17	37	39	77	41	10	90	16	21	9	75	16	26	18	16	16
	Number of steps in main scenario	Mean	4.82	4.76	4.92	2.95	4.34	5.78	3.90	5.10	3.38	4.33	3.67	6.36	3.44	6.58	5.22	2.88	3.63
		SD	2.41	0.97	1.46	2.69	2.25	1.26	1.20	2.41	0.50	2.01	1.32	3.25	1.67	1.50	1.40	1.20	1.50
	Use cases with extensions		72.1%	94.1%	51.4%	41.0%	55.8%	92.7%	100%	68.9%	81.3%	90.5%	55.6%	98.7%	75.0%	100%	38.9%	50.0%	62.5%
	Number of extensions in use case	Mean	1.57	1.29	0.57	0.95	0.92	1.63	1.00	1.28	2.94	2.33	1.00	2.69	1.75	4.46	0.39	0.63	0.88
		SD	1.88	0.77	0.60	1.72	1.12	0.62	0.00	1.30	2.98	1.58	1.58	2.91	1.57	1.61	0.50	0.81	0.81
	Number of steps in extension	Mean	2.46	1.80	2.52	1.00	1.45	N/A	1.10	2.77	2.30	1.64	1.44	3.09	N/A	N/A	2.67	1.30	1.36
		SD	1.61	0.84	0.87	0.00	0.59	N/A	0.32	1.87	0.65	0.67	0.53	1.80	N/A	N/A	2.89	0.48	0.50
	Steps with validation actions		3.4%	3.7%	5.5%	11.3%	1.8%	0.0%	0.0%	0.0%	27.8%	17.6%	3.0%	0.0%	0.0%	15.2%	0.0%	2.2%	3.4%
	Extensions which are validations		41.3%	9.1%	76.2%	64.9%	63.4%	40.3%	100%	50.4%	78.7%	89.8%	100%	15.3%	60.7%	21.6%	0.0%	33%	50.0%
	Main actor's steps sequence length in main scenario	1	61.6%	42.3%	93.1%	88.6%	37.0%	61.9%	87.5%	91.7%	47.4%	52.4%	50.0%	52.6%	45.5%	67.4%	52.8%	71.4%	45.0%
		2	21.6%	23.1%	3.5%	6.8%	34.8%	36.1%	12.5%	0.0%	0.0%	19.1%	42.9%	16.4%	27.3%	32.7%	33.3%	14.3%	10.0%
		3	10.2%	26.9%	3.5%	4.6%	16.3%	0.0%	0.0%	8.3%	31.6%	23.8%	7.1%	12.9%	22.7%	0.0%	11.1%	14.3%	30.0%
		4	3.8%	7.7%	0.0%	0.0%	6.5%	0.0%	0.0%	0.0%	21.1%	0.0%	0.0%	8.6%	4.6%	0.0%	2.8%	0.0%	15.0%
		>4	2.8%	0.0%	0.0%	0.0%	5.4%	2.1%	0.0%	0.0%	0.0%	4.8%	0.0%	9.5%	0.0%	0.0%	0.0%	0.0%	0.0%
	Secondary actor's steps sequence length in main scenario	1	74.6%	82.6%	96.3%	72.2%	67.9%	100%	100%	90.0%	50.0%	12.5%	62.5%	72.1%	100.0%	0.0%	0.0%	60.0%	10.0%
		2	18.8%	17.4%	3.7%	22.2%	29.6%	0.0%	0.0%	8.0%	16.7%	43.8%	37.5%	14.8%	0.0%	83.3%	68.4%	40.0%	0.0%
		3	4.3%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	2.0%	33.3%	25.0%	0.0%	7.4%	0.0%	16.7%	26.3%	0.0%	0.0%
		4	1.6%	0.0%	0.0%	5.6%	2.5%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	4.1%	0.0%	0.0%	5.3%	0.0%	0.0%
		>4	0.8%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	18.8%	0.0%	1.6%	0.0%	0.0%	0.0%	0.0%	0.0%
Requirements specification ***dependent***	Use cases with additional description		38.4%	0.0%	0.0%	100%	0.0%	0.0%	0.0%	100%	56.3%	100%	0.0%	6.7%	100%	96.3%	0.0%	0.0%	6.3%
	Number of use cases with sub-scenario		12.4%	0.0%	0.0%	0.0%	32.5%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	44.2%	0.0%	0.0%	33.3%	0.0%	0.0%
	Number of steps in sub-scenario	Mean	1.92	0.00	0.00	0.00	3.20	0.00	0.00	0.00	0.00	0.00	0.00	1.09	0.00	0.00	1.33	0.00	0.00
		SD	1.62	N/A	N/A	N/A	2.02	N/A	N/A	N/A	N/A	N/A	N/A	0.29	N/A	N/A	0.52	N/A	N/A
	Use cases with pre-conditions		37.4%	82.4%	100%	0.0%	0.0%	97.6%	0.0%	0.0%	0.0%	23.8%	77.8%	0.0%	81.3%	38.5%	0.0%	0.0%	56.3%
	Use cases with post-conditions		14.3%	0.0%	0.0%	97.4%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	100%	100%	0.0%	0.0%	12.5%
	Use cases with triggers		33.0%	100%	100%	100%	0.0%	100%	0.0%	0.0%	68.8%	57.1%	0.0%	0.0%	0.0%	0.0%	0.0%	100%	0.0%
	Number of steps with reference to use cases		6.4%	0.0%	0.5%	0.0%	0.0%	0.0%	25.6%	14.2%	0.0%	0.0%	6.1%	33.3%	0.0%	0.6%	0.0%	0.0%	0.0%
	Number of extensions with scenario		66.8%	27.3%	100%	8.1%	87.3%	0.0%	100%	100%	100%	100%	100%	100%	0.0%	0.0%	42.9%	100.0%	100%
	Number of extensions with stories		33.2%	72.7%	0.0%	91.9%	12.7%	100%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	100%	100%	57.1%	0.0%	0.0%
	Explicitly defined business rules		N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A

Table 3. Results of the quality analysis of the use cases stored in UCDB

	Property	Overall	Project															
			A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
Use-case set level	Use case duplicates (the same actions operating on different business objects)	2.55%	0.00%	0.00%	0.00%	0.00%	26.83%	0.00%	2.22%	0.00%	0.00%	0.00%	0.00%	6.25%	0.00%	0.00%	0.00%	0.00%
	Lack of hierarchical structure (Y/N)	56.25%	Y	Y	Y	N	Y	N	N	Y	Y	N	N	N	N	Y	Y	Y
Use-case level	Number of use cases, in which name doesn't describe the goal	4.76%	11.76%	0.00%	51.28%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	31.25%	6.25%
	Number of non-detectable condition in extensions	4.25%	4.55%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	8.51%	0.00%	0.00%	14.85%	0.00%	0.00%	0.00%	0.00%	0.00%
	Number of extensions without scenarios	35.80%	77.27%	0.00%	86.49%	6.49%	100%	0.00%	5.22%	14.89%	8.16%	0.00%	0.00%	100%	100%	0.00%	100%	0.00%
	Number of extensons with nested scenarios	4.13%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	8.51%	8.16%	0.00%	12.87%	0.00%	0.00%	0.00%	0.00%	0.00%
Scenario level	Number of use cases with less than 3 steps in main scenario	11.39%	0.00%	0.00%	51.28%	11.69%	0.00%	20.00%	15.56%	0.00%	9.52%	22.22%	0.00%	37.50%	7.69%	0.00%	37.50%	25.00%
	Number of use cases with more than 9 steps in main scenario	6.80%	0.00%	0.00%	5.13%	7.79%	0.00%	0.00%	5.56%	0.00%	0.00%	0.00%	36.00%	0.00%	0.00%	0.00%	0.00%	0.00%
	Lack of interactions between actors	22.11%	11.76%	10.81%	56.41%	48.05%	9.76%	0.00%	0.00%	50.00%	28.57%	0.00%	12.00%	37.50%	3.85%	44.44%	87.50%	56.25%
	Number of use cases with at least double nested sub-scenarios	1.53%	0.00%	0.00%	0.00%	10.39%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	1.33%	0.00%	19.23%	0.00%	0.00%	0.00%
Step level (steps from main scenario and from extensions)	Number of steps with different tenses used	0.80%	1.11%	0.85%	5.08%	0.00%	0.00%	0.00%	0.00%	0.00%	5.95%	0.00%	0.18%	0.00%	0.00%	6.86%	0.00%	5.17%
	Number of steps in which user interface terms are used	4.56%	0.00%	0.42%	0.00%	0.00%	0.00%	0.00%	12.87%	0.00%	0.00%	2.17%	0.36%	1.82%	0.00%	47.06%	16.67%	1.30%
	Number of steps in which technical terms are used	0.29%	0.00%	0.00%	0.85%	0.24%	0.00%	0.00%	0.00%	0.00%	1.19%	0.00%	0.09%	0.00%	0.00%	5.88%	0.00%	0.00%
	Number of steps in which no actor is specified	19.37%	0.00%	1.69%	38.14%	0.94%	0.84%	0.00%	45.05%	0.68%	0.60%	0.00%	0.54%	0.00%	0.58%	13.73%	51.67%	20.78%
	Number of steps in which passive voice for the action is used	3.63%	1.11%	0.00%	55.93%	0.24%	0.84%	0.00%	2.70%	0.00%	0.00%	0.00%	0.27%	0.00%	0.00%	13.73%	31.67%	16.88%
	Number of steps with complex sentence structure or with more than one sentence	2.51%	0.00%	5.08%	27.12%	0.71%	0.00%	0.00%	1.16%	0.68%	2.38%	2.17%	1.45%	1.82%	2.92%	9.80%	1.67%	2.60%
	Number of steps with possibility of different interpretations	3.79%	0.00%	16.95%	11.02%	0.00%	0.00%	4.00%	0.77%	0.00%	0.00%	4.35%	4.09%	1.82%	12.28%	6.86%	11.67%	2.60%
	Number of steps with details of elements of business objects	3.71%	4.44%	0.85%	3.39%	3.30%	0.00%	0.00%	11.45%	0.00%	0.00%	8.70%	0.91%	0.00%	7.60%	0.98%	3.33%	0.00%
	Number of steps with conditional clauses	1.22%	0.00%	0.00%	4.24%	0.71%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	3.00%	0.00%	0.58%	3.92%	1.67%	0.00%
	Number of steps with language mistakes	3.16%	0.00%	47.03%	0.00%	0.00%	0.00%	0.00%	0.00%	2.05%	0.60%	0.00%	0.00%	0.00%	0.00%	2.94%	0.00%	5.19%

One of the main observations is that almost 86 % of the analysed use cases contain at least one defect, which proves that it is not easy to provide requirements specification of good quality. For researchers who try to analyse use case quality the knowledge of average defects density can be used to tune their methods and tools in order to make them more effective and efficient.

Large number of different defects can be found in most of the projects. For example steps without explicitly specified actor can be found in 13 projects, 11 projects have some extensions missing alternative scenarios, the same number of projects have use cases with less than 3 steps. Some of the defects are unique only to some specifications. For instance langauge mistakes were found mainly in project B (over 47 % of steps have language mistakes), while most of the projects did not have such problems. Scenarios longer than 9 steps is another example of a project-specific defect (it mainly relates to Project K, which has over 30 % of use cases with this defect). The simple explanation for this is that different authors have different writing styles and probably they are not aware of some problems and defects.

Another interesting finding is that one defect can lead to another defect. The analysis shows that steps without actor tend to use passive voice (projects C, N, O, P). A similar situation can be observed in case of language mistakes and complex-steps structure which often lead to different possible interpretations (projects B, C and N).

5 BUILDING REFERENTIAL SPECIFICATION

Combining use-cases model and average values coming from the analysis, a profile of the typical use-cases-based requirements specification can be derived. In such specification all properties would appear with the typical intensity. In other words, analysis performed on such document could be perceived as an every-day task for use-cases analysis tools.

There might be at least two approaches to acquire instance of such specification. The first one would be to search for the industrial set of use cases, which would fulfil given criteria. Unfortunately, most of industrial documents are confidential, therefore they could not be used at large scale. The second, obvious approach would be to develop such specification from scratch. If it is built according to the obtained typical specification profile, it might be used instead of industrial specification. Since it would describe some abstract system it can be freely distributed and used by anyone.

Therefore we would like to propose the approach to develop an instance of the referential specification for the benchmarking purpose. The document is available at the web site [8] and might be freely used for further research.

5.1 Specification Domain and Structure

It seems that large number of tools for the use-cases analysis have their roots in the research community. Therefore the domain of the developed specification should be

easy to understand, especially for the academia staff. One of the well-recognised processes at university is the *students admission process*. Although the developed specification will describe hypothetical process and tools, it should be easy to understand for anyone who has ever participated in a similar process.

Another important decision concerns the structure of the specification (number of actors, use cases and business objects). Unfortunately the decision is rather arbitrary, because the number of use cases, actors and business objects may depend on the size of the system and on the level of details incorporated into its description. In this case a median number of use cases and actors from the UCDB set has been used as the number of use cases and actors in the constructed specification. The first version of specification consisted of 7 actors (Administrator, Candidate, Bank, Selection Committee, Students Management System, System, User), 37 use cases (3 business, 33 user, and 1 sub-function level), and 10 business objects. After extending UCDB with data from additional 5 projects the median number of use cases decreased to 20. Therefore we decided to prepare two versions of the specification. The larger one has 37 use cases and is labeled as FULL, while in case of smaller specification (SMALL) the number of use cases was reduced to 20.

5.2 Use Cases Structure

A breadth-first rule has been followed in order to construct referential specification. Firstly, all use cases were titled and augmented with descriptions. Secondly, all of them were filled with main scenarios and corresponding extensions. During that process all changes made in specification were recorded in order to check conformance with the typical profile. This iterative approach was used until the final versions of both SMALL and FULL *quantitative specifications* were constructed. Their profiles are presented in Table 4, in comparison to the corresponding average values from the UCDB projects analysis The most important properties are those which are observed in all of the specifications (specification independent). In case of those metrics most values in the referential document are very close to those derived from the analysis. This situation differs in case of properties which were characteristic only for certain projects (or variability between projects were very high). If the constructed specification is supposed to be a typical one, it should not be biased by features observed only in some of the industrial specifications. Therefore, dependent properties were also incorporated into the referential use cases; however, they were not a subject of the tuning process.

In addition, based on each of the quantitative specifications, a corresponding *qualitative specification* was proposed. In this case we focused on introducing changes which led to obtaining a qualitative profile close to the typical profile.

6 BENCHMARKING USE CASES – CASE STUDY

Having the example of the typical requirements specification, one can wonder how it could be used. Firstly, researchers who construct methods and tools in order

Property		**Quantitative referential specification *admission system 2.0 SMALL***	**Quantitative referential specification *admission system 2.0 FULL***	**Observed in use-cases database**	**Qualitative referential specification *admission system 2.0 SMALL***	**Qualitative referential specification *admission system 2.0 FULL***
Number of steps in main scenario	Mean	4.80	4.81	4.8206	-	-
	SD	1.21	1.50	2.4114	-	-
Use cases with extensions		75.0%	70.3%	72.1%	-	-
Number of extensions in use case	Mean	1.60	1.50	1.5744	-	-
	SD	0.71	0.69	1.8798	-	-
Number of steps in extension	Mean	2.29	2.49	2.4584	-	-
	SD	1.81	2.12	1.6138	-	-
Steps of validation nature		4.2%	2.3%	3.4%	-	-
Extensions of validation nature		62.5%	46.2%	41.3%	-	-
Main actor's steps sequence length in main scenario	1	64.5%	61.7%	61.6%	-	-
	2	19.4%	21.7%	21.6%	-	-
	3	12.9%	10.0%	10.2%	-	-
	4	3.2%	3.3%	3.8%	-	-
	>4	0.0%	3.3%	2.8%	-	-
Secondary actor's steps sequence length in main scenario	1	73.5%	79.7%	74.6%	-	-
	2	23.5%	15.3%	18.8%	-	-
	3	2.9%	3.4%	4.3%	-	-
	4	0.0%	1.7%	1.6%	-	-
	>4	0.0%	0.0%	0.8%	-	-
Use case duplicates (the same actions operating on different business objects)		-	-	2.55%	5.00%	2.70%
Number of use cases, in which name does not describe the goal		-	-	4.76%	5.00%	5.41%
Number of non-detectable condition in extensions		-	-	4.25%	4.17%	5.13%
Number of extensions without scenarios		-	-	35.80%	33.33%	33.33%
Number of extensons with nested scenarios		-	-	4.13%	4,17%%	5.13%
Number of use cases with less than 3 steps in main scenario		-	-	11.39%	10.00%	10.81%
Number of use cases with more than 9 steps in main scenario		-	-	6.80%	5.00%	5.41%
Lack of interactions between actors		-	-	22.11%	25.00%	24.32%
Number of use cases with at least double nested sub-scenarios		-	-	1.53%	0.00%	2.70%
Number of steps with different tenses used		-	-	0.80%	0.78%	0.80%
Number of steps in which user interface terms are used		-	-	4.56%	4.59%	4.78%
Number of steps in which technical terms are used		-	-	0.29%	0.00%	0.40%
Number of steps in which no actor is specified		-	-	19.37%	19.53%	19.12%
Number of steps in which passive voice for the action is used		-	-	3.63%	3.91%	3.59%
Number of steps with complex sentence structure or with more then one sentence		-	-	2.51%	2.34%	2.39%
Number of steps with possibility of different interpretations		-	-	3.79%	3.91%	3.98%
Number of steps with details of elements of business objects		-	-	3.71%	3.91%	3.59%
Number of steps with conditional clauses		-	-	1.22%	1.56%	1.59%
Number of steps with language mistakes				3.16%	3.13%	3.19%

Table 4. Referential specifications profiles in comparison to the average profile derived from the analysis of the use cases stored in UCDB

to analyse use cases often face the problem of evaluating their ideas. Using some specification coming from the industrial project is a typical approach; however, this cannot lead to the conclusion that the given solution would give the same results for other specifications. The situation looks different when the typical specification is considered; then researchers can assume that their tool would work for most of the industrial specifications. Secondly, analysts who want to use some tools to analyse their requirements can have a problem to choose the best tool that would meet their needs. With the typical specification in mind it can be easier to evaluate the available solutions and choose the most suitable one. To conclude, the example of the typical requirements specification can be used:

- to compare two specifications and assess how the given specification is similar to the typical one;
- to assess the time required for a tool to analyse requirement specification;
- to assess the quality of a tool/method;
- to compare tools and choose the best one.

In order to demonstrate the usage of the constructed specification we have conducted a case study using the qualitative reference specification *Admission system 2.0 FULL*. As a tool for managing requirements in a form of use cases we chose the UC Workbench tool [17], which has been developed at Poznań University of Technology since the year 2005. In addition, we used the set of tools for defects detection [6] to analyse the requirements. The aim of the mentioned tools is to find potential defects and present them to analyst during the development of the requirements specification. The tools are based on the Natural Language Processing (NLP) methods and use Standford parser [9] and/or OpenNLP [20] tools to perform the NLP analysis of the requirements.

6.1 Quality Analysis

In order to evaluate the quality of the tools, a confusion matrix will be used [12] (see Table 5).

		Expert answer	
		Yes	No
System answer	Yes	**TP**	**FP**
	No	**FN**	**TN**

Table 5. Confusion matrix

The symbols used in Table 5 are described below:

- *T (true)* – tool answer that is consistent with the expert decision;
- *F (false)* – tool answer that is inconsistent with the expert decision;
- *P (positive)* – positive system answer (defect occurs);
- *N (negative)* – negative system answer (defect does not occur).

On the basis of the confusion matrix, the following metrics [16] can be calculated:

- *Accuracy (AC)* – proportion of the total number of predictions that were correct. It is determined using Equation (1).

$$AC = \frac{TP + TN}{TP + FN + FP + TN} \tag{1}$$

- *True positive rate (TP rate)* – proportion of positive cases that were correctly identified, as calculated using Equation (2).

$$TP\ rate = \frac{TP}{TP + FN} \tag{2}$$

- *True negative rate (TN rate)* is defined as the proportion of negative cases that were classified correctly, as calculated using Equation (3).

$$TN\ rate = \frac{TN}{TN + FP} \tag{3}$$

- *Precision (PR)* – proportion of the predicted positive cases that were correct, as calculated using Equation (4).

$$PR = \frac{TP}{TP + FP} \tag{4}$$

The referential use-case specification was analysed by the mentioned tools (to perform the NLP processing Stanford library was used) in order to find 10 types of defects described in [6]. Aggregated values of the above accuracy metrics are as follows:

- $AC = 0.99$
- $TP\ rate = 0.97$
- $TN\ rate = 0.99$
- $PR = 0.80$.

This shows that the developed tools for defects detection are rather good, however the precision should be better, so the researchers could conclude that more investigation is needed in this area.

If the researchers worked on their tool using only defect-prone or defect-free specifications the results could be distorted and could lead to some misleading conclusions (e.g. that the tool is significantly better or that it gives very poor outcome). Having access to the typical specification allows the researchers to explore the main difficulties the tool can encounter during working with the industrial specifications.

6.2 Time Analysis

One of the main characteristics of a tool being developed is its efficiency. Not only developers are interested in this metric, but also the users – e.g. when analysts have to choose between two tools which give the same results, they would choose the one which is more efficient. However, it can be hard to asses the efficiency just by running the tool with any data, as this may result with distorted outcome. Use-cases benchmark gives the opportunity to measure the time required by a tool to complete its computations in an objective way. It can be also valuable for researchers constructing tools to consider using different third-party components in their applications.

For instance, in case of the mentioned defect-detection tool there is a possibility of using one of the available English grammar parsers. In this case study two of them will be considered – Standford parser and OpenNLP. If one exchanges those components the tool will still be able to detect defects, but its efficiency and accuracy may change. Therefore, the time analysis was performed to asses how using those libraries influences the efficiency of the tool. The overall time required to analyse the typical specification[2] for the tool using Stanford parser was 53.11 seconds and for the one with OpenNLP it was 29.46 seconds. Although the first time value seems to be large, the examined tool needed 280 ms on the average to analyse a single use-case step, so it should not be a problem to use it in industrial environment. Of course, when comparing efficiency, one has to remember that other factors (like the quality of the results, memory requirements or the time needed for the initialization of the tools) should be taken into account, as they can also have significant impact on the tool itself. Therefore, time, memory usage and quality analysis is presented in Table 6. This summarised statistics allow researchers to choose the best approach for their work.

Although this specification describes some abstract system, it shows the typical phenomenon of the industrial requirements specifications. The conducted case study shows that the developed referential use-case specification can be used in different ways by both researchers and analysts.

[2] Tests were performed for the tools in two versions: with Stanford and OpenNLP parsers. Each version of the tools analysed the referential specification five times, on the computer with Pentium Core 2 Duo 2.16 GHz processor and 2 GB RAM.

	Summarised for all components			*English grammar parser only*			
English grammar parser used	Overall processing time [s]	Mean time needed to analyse one step [s]	Maximal memory utilization [MB]	Overall processing time [s]	Startup time [s]	Initial memory usage [MB]	Memory usage while processing single element [KB]
Stanford	53.11	0.28	212	37.01	0.8	27	539
OpenNLP	29.46	0.16	338	13.58	15.4	225	3977
	Quality						
	AC	TP rate	TN rate	PR			
Stanford	0.99	0.97	0.99	0.80			
OpenNLP	0.99	0.83	0.99	0.79			

Table 6. Case-study results (tools are written in Java, so memory was automatically managed – garbage collection)

7 CONCLUSIONS

In the paper an approach to create a set of reference use-cases-based requirements specifications was presented.

In order to derive a profile of a typical use-cases-based requirements specification 524 use cases were analysed. It has proved that some of the use-case properties are project-independent (are observed in most of the projects). They have been used for creating the referential specification. On the other hand, there is also a number of properties which depend very much on the author.

In order to present potential usage of the benchmark specification, a case study was conducted. Two sets of tools for defect detection in requirements were compared from the point of view of efficiency and accuracy.

In this paper a second release of UCDB is analysed, which was recently extended from 432 to 524 use cases. A positive observation is that despite the number of use cases increased by 21 % the typical profile did not change a lot. Therefore it seems that the typical profile derived based on the use cases stored in the database is stable and should not differ much in the future releases.

Acknowledgments

We would like to thank Piotr Godek, Kamil Kwarciak and Maciej Mazur for supporting us with industrial use cases.

The research presented at the CEE-SET '08 [3] and being part of this paper has been financially supported by the Polish Ministry of Science and Higher Education under grant N516 001 31/0269.

Additional case studies and analysis have been financially supported by Foundation for Polish Science Ventures Programme co-financed by the EU European Regional Development Fund.

REFERENCES

[1] Chaos: A Recipe for Success. 2004, Technical report, Standish Group, 2004.

[2] Adolph, S.—Bramble, P.—Cockburn, A.—Pols, A.: Patterns for Effective Use Cases. Addison-Wesley, 2002.

[3] Alchimowicz, B.—Jurkiewicz, J.—Nawrocki, J.—Ochodek, M.: Towards Use-Cases Benchmark. In: 3rd IFIP Central and East European Conference on Software Engineering Techniques CEE-SET 2008, 2008.

[4] Anda, B.—Sjøberg, D. I. K.: Towards an Inspection Technique for Use Case Models. In: Proceedings of the 14th international conference on software engineering and knowledge engineering, 2002, pp. 127–134.

[5] Bernardez, B.—Duran, A.—Genero, M.: Empirical Evaluation and Review of a Metrics-Based Approach for Use Case Verification. Journal of Research and Practice in Information Technology, Vol. 36, 2004, No. 4, pp. 247–258.

[6] Ciemniewska, A.—Jurkiewicz, J.—Nawrocki, J.—Olek, Ł.: Supporting Use-Case Reviews. In: 10th International Conference on Business Information Systems, Lecture Notes in Computer Science, Vol. 4439, April 2007, pp. 424–437.

[7] Cockburn, A.: Writing Effective Use Cases. Addison-Wesley Boston, 2001.

[8] Use Case Database. Availaible on: `http://www.ucdb.cs.put.poznan.pl`.

[9] de Marneffe, M.-C.—MacCartney, B.—Manning, Ch. D.: Generating Typed Dependency Parses from Phrase Structure Parses. In: Proceedings of the EACL workshop on Linguistically Interpreted Corpora (LINC), 2006.

[10] Denger, C.—Paech, B.—Freimut, B.: Achieving High Quality of Use-Case-Based Requirements. Informatik-Forschung und Entwicklung, Vol. 20, 2005, No. 1, pp. 11–23.

[11] Diev, S.: Use Cases Modeling and Software Estimation: Applying Use Case Points. ACM SIGSOFT Software Engineering Notes, Vol. 31, 2006, No. 6, pp. 1–4.

[12] Fawcett, T.: Roc Graphs: Notes and Practical Considerations for Data Mining Researchers. HP Labs Technical Report HPL-2003-4, 2003.

[13] Hurlbut, R.: A Survey of Approaches for Describing and Formalizing Use Cases. Expertech, Ltd., 1997.

[14] Jacobson, I.: Object-Oriented Development in an Industrial Environment. ACM SIGPLAN Notices, Vol. 22, 1987, No. 12, pp. 183–191.

[15] Jacobson, I.—Christerson, M.—Jonsson, P.—Overgaard, G.: Object-Oriented Software Engineering: A Use Case Driven Approach. 1992.

[16] Krawiec, K.—Stefanowski, J.: Uczenie Maszynowe i Sieci Neuronowe. Wydawnictwo Politechniki Poznańskiej, 2004.

[17] Nawrocki, J.—Olek, Ł.: UC Workbench – A Tool for Writing Use Cases. In: 6th International Conference on Extreme Programming and Agile Processes, Lecture Notes in Computer Science, Vol. 3556, Jun 2005, pp. 230–234.

[18] Neill, C. J.—Laplante, P. A.: Requirements Engineering: The State of the Practice. Software, IEEE, Vol. 20, 2003, No. 6, pp. 40–45.

[19] Niazi, M.—Shastry, S.: Role of Requirements Engineering in Software Developement Process: An Empirical Study. In: Multi Topic Conference, 2003, INMIC, 7th International, 2003.

[20] OpenNLP. Availaible on: `http://opennlp.sourceforge.net`.

[21] Shull, F. J.—Carver, J. C.—Vegas, S.—Juristo, N.: The Role of Replications in Empirical Software Engineering. Empirical Software Engineering. Vol. 13, April 2008, No. 2, pp. 211–218.

[22] Somé, S. S.: Supporting Use Case Based Requirements Engineering. Information and Software Technology, Vol. 48, 2006, No. 1, pp. 43–58.

Bartosz Alchimowicz is a Ph. D. student working at the Faculty of Computing and Management, Poznan University of Technology in Poland. His main fields of interest are requirements engineering, software configuration management and quality of documentation.

Jakub Jurkiewicz is a Ph. D. student at Poznań University of Technology, Poland. He received his M. Sc. degree in computer science in the Faculty of Computer Science and Management of Poznan Univerity of Technology in 2007. His research interests are requirements engineering, quality requirements, software measurement and natural language processing.

Mirosław Ochodek is a Ph. D. student working at the Institute of Computing Science at the Poznan University of Technology. He is mainly working in the domain of requirements engineering, software metrics, functional size measurement, and software effort estimation.

Jerzy Nawrocki received his M. Sc. degree (1980), Ph. D. degree (1984), and Dr. hab. degree (1994) – all in informatics and all from the Poznan University of Technology (PUT), Poznan, Poland. Currently he is the Dean of the Faculty of Computing and Management at PUT, and a Secretary of IFIP Technical Committee 2: Software Theory and Practice.