Towards Use-Cases Benchmark

Bartosz Alchimowicz, Jakub Jurkiewicz, Jerzy Nawrocki, Miroslaw Ochodek

Towards Use-Cases Benchmark*

Bartosz Alchimowicz, Jakub Jurkiewicz, Jerzy Nawrocki, Mirosław Ochodek

Poznan University of Technology, Institute of Computing Science,
ul. Piotrowo 3A, 60-965 Poznań, Poland
{Bartosz.Alchimowicz, Jakub.Jurkiewicz, Jerzy.Nawrocki,
Miroslaw.Ochodek}@cs.put.poznan.pl

Abstract. In the paper an approach to developing a use-cases benchmark is presented. The benchmark itself is a referential use-case-based requirements specification, which has a typical profile observed in real projects. To obtain this profile an extensive analysis of 432 use cases coming from 11 projects was performed. Because the developed specification represents those found in real projects, it might be used in order to present, test, and verify methods and tools for use-case analysis. This is especially important because industrial specifications are in most cases confident, and they might not be used by researchers who would like to replicate studies performed by their colleagues.

Keywords: Use cases, Requirements engineering, Metrics, Benchmark

1 Introduction

Functional requirements are very important in software development. They impact not only the product 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. [2,3,8,9,20]). Ideas presented by researches must be empirically verified (in best case, using industrial data) and it should be possible to replicate a given experiment by any other researcher [19]. 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 has 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 will be 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 the use-cases-based requirements specification, which has a

* This research has been financially supported by the Polish Ministry of Science and Higher Education grant N516 001 31/0269.

typical profile observed in requirements coming from the real projects. To derive such a profile an extensive analysis of 432 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 Section 3, which presents an analysis of the use-cases, coming from eleven projects. Based on the analysis a profile of the typical use-case-based specification is derived, which is used to create a benchmark specification in Section 4. Finally, a case study is described in Section 5, which presents a typical usage of the benchmark specification in order to compare tools for 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. [16] 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 [13] many approaches for their development were proposed. Russell R. Hurlbut [11], 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 a level of formalism and 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 (see figure 1). It incorporates most of the best-practices presented in [1,5].

It still allows to create specification that contains only scenario (or non-structured story) and actors, which is enough to compose small but valuable use-case, as well as extend it with other elements, for example extensions (using steps or stories), pre- and post-conditions, notes or triggers.

Unfortunately, understanding the structure of specification is still not enough in order 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?".

3 Analysis of use-cases-based specifications structure

In order to discover the profile of typical-specification, data from various projects was collected and analysed. As a result a database called UCDB (*Use Cases Database*) [6] has been created. At this stage it stores data from 11 projects with the total number of 432 use cases (basic characteristics of requirements specifications as well as projects descriptions are grouped in table 1).

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

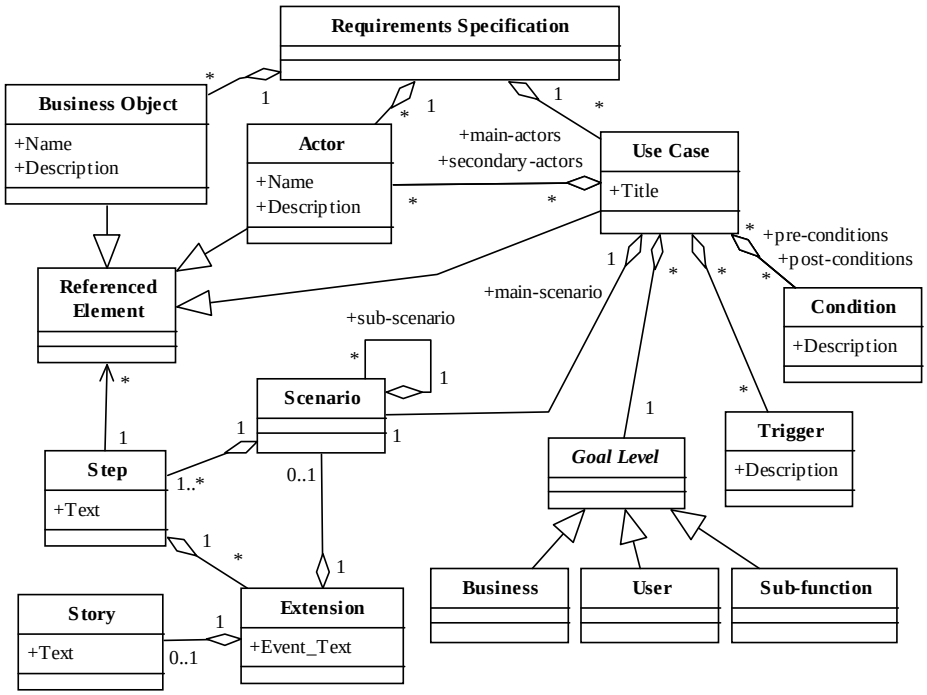

Fig. 1. Use-Cases-based functional requirements specification model

All of the specifications have been analysed in order to populate the model with the information concerning average number of its elements occurrence and some additional quantifiable data.

One of the interesting findings is that 79.9% of the main-scenarios in the use cases consist of 3-9 steps, which means that they fulfil the guidelines proposed by Cockburn [5]. What is more 72.9% of the analysed use cases are augmented with the alternative scenarios (extensions). There are projects which contains extensions for all steps in main scenario, however on the average use case contains 1.5 extension. Detailed information regarding number and distributions of steps in both - main scenario and extensions, is presented in figure 2.

Another interesting observation, concerning the structure of use cases, is that the sequences of steps performed by the same actor, frequently contains more than one step. What is even more interesting this tendency is more visible in case of main actor's steps (37.3% of main actor's steps sequences are longer than one step, and only 19.4% in case of secondary actor - in most cases system being built). This is probably because actions performed by main actor are more important, from the business point of view. This is contradict to the concept of transactions in use cases presented by Ivar Jacobson [12]. Jacobson enumerated four types of actions which may form together use-case transaction. Only one

Table 1. Analysed projects requirements-specifications (*origin: industry - project developed by software development company, s2b - project developed by students for external organisation, external - specification obtained from the external source which is freely accessible through the Internet - this refers to two specifications: UKCDR [21], PIMS [18]; projects D and K come from the same organisation)*

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)

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 look that those actions should frequently form together longer sequences. However, 80.6% of steps sequences performed by system consisted of single step only. The distributions and number of steps in main actor's sequences are presented in figure 3.

If we look deeper into the textual representation of the use cases, some interesting observation concerning their semantic might be made. One of them regards the way use-cases authors describe validations actions. Two different approaches are observed. The most common is to use extensions to represent alternative system behaviour in case of verification-process failure (46.6% of extensions have this kind of nature). 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 3.0% of steps. There is also yet another interesting semantic structure

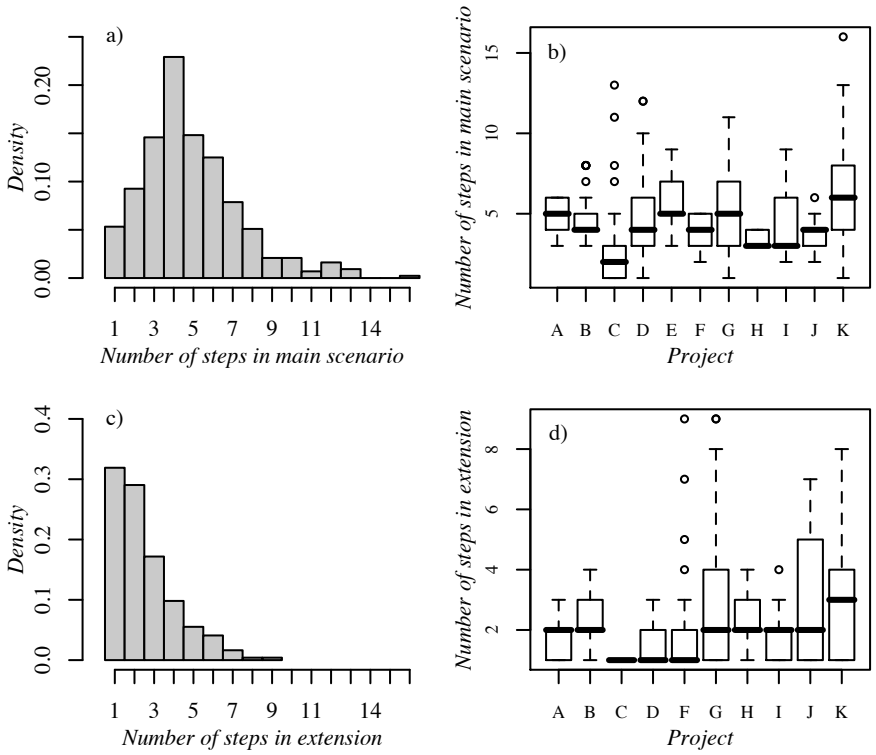

Fig. 2. Scenarios lengths in analysed use cases *a) histogram presents the number of steps in main scenario (data aggregated from all of the projects), b) box plot presents the number of steps in main scenario (in each project), c) histogram presents the number of steps in extension (data aggregated from all of the 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)*

used to represent alternative execution paths - conditional clauses (which in fact, are rather deprecated). Fortunately this kind of statements are observed only in 3.2% of steps (and they occur intensively only in 2 projects).

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 kind of properties are 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 wich are either characteristic only for a certain set of use cases or their occurrence in different specifications are in between of two extremes - full presence or marginal/none. Such proper-

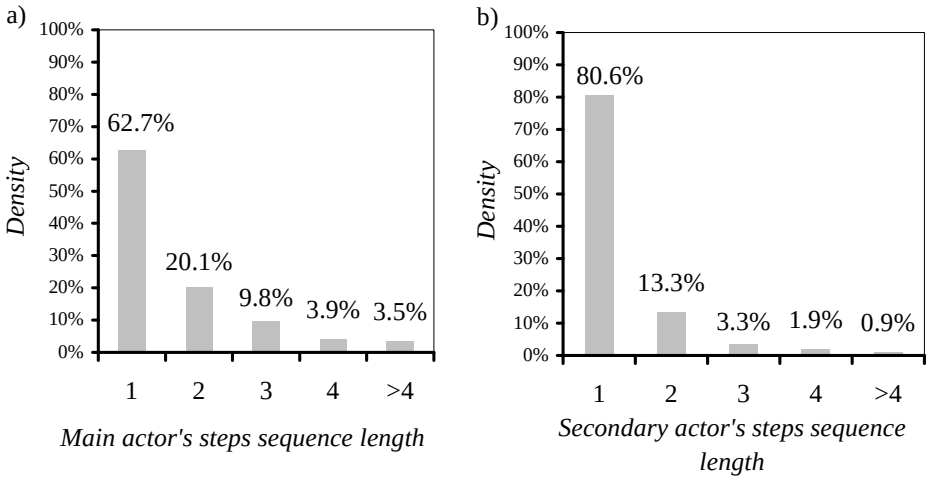

Fig. 3. Distributions of actors steps-sequences lengths in **analysed use cases**, *a) histogram presents the length of the steps sequences performed by main actor, b) histogram presents length of the steps sequences performed by secondary actor - e.g. System*

ties are project-dependent. More detailed description of analysed requirements specifications is presented in table 2.

4 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 approach to develop an instance of the referential specification for the benchmarking purpose. The document is available at the web site [6] and might be freely used for further research.

Table 2. Use-Cases Database analysis according to the presented requirements specification model (see section 2)

	Property		Overall	Project										
				A	B	C	D	E	F	G	H	I	J	K
*Requirements specification **independent***	Number of use cases		432	17	37	39	77	41	10	90	16	21	9	75
	Number of steps in main scenario	Mean	4.87	4.76	4.92	2.95	4.34	5.78	3.90	5.10	3.38	4.33	3.67	6.36
		SD	2.48	0.97	1.46	2.69	2.25	1.26	1.20	2.41	0.50	2.01	1.32	3.25
	Use cases with extensions		72.9%	94.1%	51.4%	41.0%	55.8%	92.7%	100%	68.9%	81.3%	90.5%	55.6%	98.7%
	Number of extensions in use case	Mean	1.50	1.29	0.57	0.95	0.92	1.63	1.00	1.28	2.94	2.33	1.00	2.69
		SD	1.84	0.77	0.60	1.72	1.12	0.62	0.00	1.30	2.98	1.58	1.58	2.91
	Number of steps in extension	Mean	2.51	1.80	2.52	1.00	1.45	N/A	1.10	2.77	2.30	1.64	1.44	3.09
		SD	1.62	0.84	0.87	0.00	0.59	N/A	0.32	1.87	0.65	0.67	0.53	1.80
	Steps with validation actions		3.0%	3.7%	5.5%	11.3%	1.8%	0.0%	0.0%	0.0%	27.8%	17.6%	3.0%	0.0%
	Extensions which are validations		46.6%	9.1%	76.2%	64.9%	63.4%	40.3%	100%	50.4%	78.7%	89.8%	100%	15.3%
	Main actor's steps sequence length in main scenario	1	62.7%	42.3%	93.1%	88.6%	37.0%	61.9%	87.5%	91.7%	47.4%	52.4%	50.0%	52.6%
		2	20.1%	23.1%	3.5%	6.8%	34.8%	36.1%	12.5%	0.0%	0.0%	19.1%	42.9%	16.4%
		3	9.8%	26.9%	3.5%	4.6%	16.3%	0.0%	0.0%	8.3%	31.6%	23.8%	7.1%	12.9%
		4	3.9%	7.7%	0.0%	0.0%	6.5%	0.0%	0.0%	0.0%	21.1%	0.0%	0.0%	8.6%
		>4	3.5%	0.0%	0.0%	0.0%	5.4%	2.1%	0.0%	0.0%	0.0%	4.8%	0.0%	9.5%
	Secondary actor's steps sequence length in main scenario	1	80.6%	82.6%	96.3%	72.2%	67.9%	100%	100%	90.0%	50.0%	12.5%	62.5%	72.1%
		2	13.3%	17.4%	3.7%	22.2%	29.6%	0.0%	0.0%	8.0%	16.7%	43.8%	37.5%	14.8%
		3	3.3%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	2.0%	33.3%	25.0%	0.0%	7.4%
		4	1.9%	0.0%	0.0%	5.6%	2.5%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	4.1%
		>4	0.9%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	18.8%	0.0%	1.6%
*Requirements specification **dependent***	Use cases with additional description		38.0%	0.0%	0.0%	100%	0.0%	0.0%	0.0%	100%	56.3%	100%	0.0%	6.7%
	Number of use cases with sub-scenario		13.7%	0.0%	0.0%	0.0%	32.5%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	44.2%
	Number of steps in sub-scenario	Mean	1.98	0.00	0.00	0.00	3.20	0.00	0.00	0.00	0.00	0.00	0.00	1.09
		SD	1.69	N/A	N/A	N/A	2.02	N/A	N/A	N/A	N/A	N/A	N/A	0.29
	Use cases with pre-conditions		48.5%	82.4%	100%	0.0%	0.0%	97.6%	0.0%	0.0%	0.0%	23.8%	77.8%	0.0%
	Use cases with post-conditions		10.6%	0.0%	0.0%	97.4%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
	Use cases with triggers		36.3%	100%	100%	100%	0.0%	100%	0.0%	0.0%	68.8%	57.1%	0.0%	0.0%
	Steps with conditional clauses		3.2%	1.2%	1.1%	17.4%	0.3%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	9.0%
	Number of steps with reference to use cases		6.3%	0.0%	0.5%	0.0%	0.0%	0.0%	25.6%	14.2%	0.0%	0.0%	6.1%	33.3%
	Number of extensions with scenario		71.9%	27.3%	100%	8.1%	87.3%	0.0%	100%	100%	100%	100%	100%	100%
	Number of extensions with stories		28.1%	72.7%	0.0%	91.9%	12.7%	100%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%

4.1 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 universities is a **students admission process**. Although developed specification will describe hypothetical process and tools, it should be easy to understand for anyone who have 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 arbitral, because number of use cases, actors and business objects may depend on the size of the system and 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 a number of use cases and actors in the constructed specification. The final specification consist 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**.

4.2 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 version of the specification has been constructed. Its profile is presented in table 3, in comparison to the corresponding average values from the UCDB projects analysis (main actor's step sequences lengths for referential specification, are additionally presented in figure 4). 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 suppose 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 tunning process.

5 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 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

Table 3. Referential specification profile in comparison to the average profile derived from the UCDB use-cases analysis

Property		**Referential Specification** *Admission System*	**Observed in Use-Cases Database**
Number of steps in main scenario	Mean	4.89	4.87
	SD	1.47	2.48
Use Cases with extensions		70.3%	72.92%
Number of extensions in use case	Mean	1.50	1.50
	SD	0.69	1.84
Number of steps in extension	Mean	2.49	2.51
	SD	2.12	1.62
Steps of validation nature		2.2%	3.0%
Extensions of validation nature		46,2%	46.6%
Steps with conditional clauses		0.6%	3.2%
Main actor's steps sequence length in main scenario	1	62.3%	62.7%
	2	21.3%	20.1%
	3	9.8%	9.8%
	4	3.3%	3.9%
	>4	3.3%	3.5%
Secondary actor's steps sequence length in main scenario	1	80.7%	80.6%
	2	12.9%	13.3%
	3	4.8%	3.3%
	4	1.6%	1.9%
	>4	0.00%	0.9%

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 asses how the given specification is similar to the typical one
- to asses the time required for a tool to analyse requirement specification
- to asses 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. As a tool for managing requirements in form of use cases we chose the UC Workbench tool [15], which has been developed at Poznan University of Technology since the year 2005. Additionally we used the set

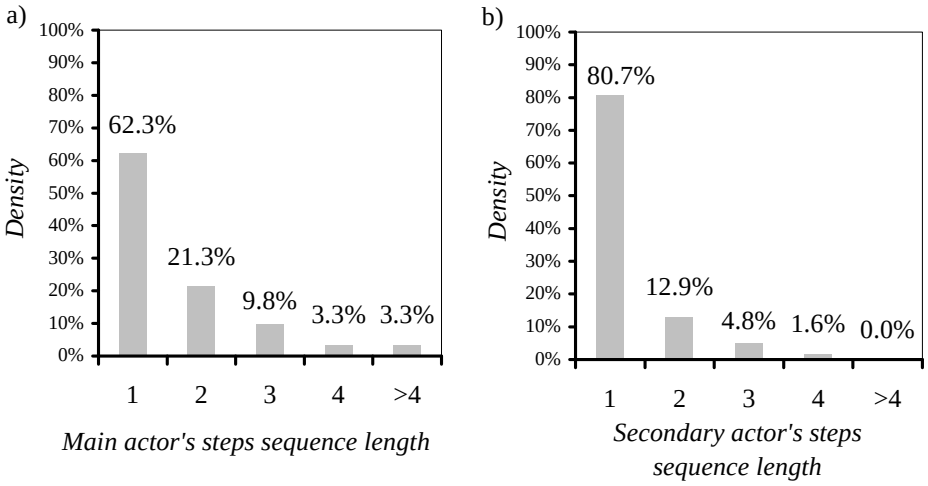

Fig. 4. Distributions of actors steps-sequences lengths in **referential specification**, *a) histogram presents the length of the steps sequences performed by main actor, b) histogram presents length of the steps sequences performed by secondary actor - e.g. System*

of tools for defects detection [4] 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 [7] and/or OpenNLP [17] tools to perform the NLP analysis of the requirements.

5.1 Quality analysis

In order to evaluate the quality of the used tools, a confusion matrix will be used [10] (see table 4).

Table 4. Confusion matrix

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

The symbols used in table 4 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)* - system positive answer (defect occurs)
- *N (negative)* - system negative answer (defect does not occur)

On the basis of the confusion matrix, following metrics [14] can be calculated:

- *Accuracy (AC)* - proportion of the total number of predictions that were correct. It is determined using the 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 the equation 2.

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

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

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

- *Precision (PR)* - proportion of the predicted positive cases that were correct, as calculated using the 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 [4]. Aggregated values of the above accuracy-metrics are as follows:

- AC = 0.99
- TP rate = 0.96
- TN rate = 0.99
- PR = 0.82

This shows that the developed tools for defects detection are rather good, however, the precision could be better and 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 main difficulties the tool can encounter during working with the industrial specifications.

5.2 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 for 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 influence the efficiency of the tool. The overall time required to analyse the typical specification[2] for the tool using Stanford parser was 54.37 seconds and for the one with OpenNLP it was 31.21 seconds. Although the first time value seems to be large, the examined tool needed on average 290 ms to analyse a single use-case step, so it should not be a problem to use it in the industrial environment. Of course, when comparing efficiency, one have 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 5. 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.

6 Conclusions

In the paper an approach to create a referential use-cases-based requirements specification was presented.

In order to derive such a specification 432 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.

[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 Duo 2.0GHz processor and 2GB RAM.

Table 5. Case-study results (*tools are written in Java, so memory was automatically managed – garbage collection*)

	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	54.37	0.29	212	37.44	0.8	27	547
OpenNLP	31.21	0.17	339	14.58	15.6	226	3925

	Quality			
	AC	TP rate	TN rate	PR
Stanford	0.99	0.96	0.99	0.82
OpenNLP	0.99	0.84	0.99	0.79

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 the future it would be beneficial to extend the use-case database with more use-cases coming from commercial projects. This would require an iterative approach to updating the typical-profile as well as the referential specification.

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

This research has been financially supported by the Polish Ministry of Science and Higher Education under grant N516 001 31/0269.

References

1. Steve Adolph, Paul Bramble, Alistair Cockburn, and Andy Pols. *Patterns for Effective Use Cases*. Addison-Wesley, 2002.
2. B. Anda and D.I.K. Sjøberg. Towards an inspection technique for use case models. *Proceedings of the 14th international conference on Software engineering and knowledge engineering*, pages 127–134, 2002.
3. B. Bernardez, A. Duran, and M. Genero. Empirical Evaluation and Review of a Metrics-Based Approach for Use Case Verification. *Journal of Research and Practice in Information Technology*, 36(4):247–258, 2004.
4. Alicja Ciemniewska, Jakub Jurkiewicz, Jerzy Nawrocki, and Łukasz Olek. Supporting use-case reviews. In *10th International Conference on Business Information Systems*, LNCS. Springer Verlag, April 2007.
5. A. Cockburn. *Writing effective use cases*. Addison-Wesley Boston, 2001.
6. Use Case Database. http://www.ucdb.cs.put.poznan.pl.

7. Marie-Catherine de Marneffe, B. MacCartney, and Ch. D. Manning. Generating typed dependency parses from phrase structure parses. In *Proceedings of the EACL workshop on Linguistically Interpreted Corpora (LINC)*, 2006.
8. C. Denger, B. Paech, and B. Freimut. Achieving high quality of use-case-based requirements. *Informatik-Forschung und Entwicklung*, 20(1):11–23, 2005.
9. S. Diev. Use cases modeling and software estimation: applying use case points. *ACM SIGSOFT Software Engineering Notes*, 31(6):1–4, 2006.
10. T. Fawcett. Roc graphs: Notes and practical considerations for data mining researchers. 2003.
11. R. Hurlbut. A Survey of Approaches for Describing and Formalizing Use Cases. *Expertech, Ltd*, 1997.
12. I. Jacobson. Object-oriented development in an industrial environment. *ACM SIGPLAN Notices*, 22(12):183–191, 1987.
13. Ivar Jacobson, Magnus Christerson, Patrick Jonsson, and Gunnar Overgaard. Object-oriented software engineering: A use case driven approach, 1992.
14. K. Krawiec and J. Stefanowski. *Uczenie maszynowe i sieci neuronowe*. Wydawnictwo Politechniki Poznańskiej, 2004.
15. Jerzy Nawrocki and Łukasz Olek. Uc workbench - a tool for writing use cases. In *6th International Conference on Extreme Programming and Agile Processes*, volume 3556 of *LNCS*, pages 230–234. Springer Verlag, Jun 2005.
16. CJ Neill and PA Laplante. Requirements Engineering: The State of the Practice. *Software, IEEE*, 20(6):40–45, 2003.
17. OpenNLP. http://opennlp.sourceforge.net.
18. PIMS. http://www.mole.ac.uk/lims/project/srs.html.
19. Forrest J. Shull, Jeffrey C. Carver, Siram Vegas, and Natalia Juristo. The role of replications in empirical software engineering. *Empirical Software Engineering*, 13(2):211–218, April 2008.
20. S.S. Somé. Supporting use case based requirements engineering. *Information and Software Technology*, 48(1):43–58, 2006.
21. UKCDR. http://www.ukoln.ac.uk/repositories/digirep/index /all_the_scenarios_and_use_cases_submitted#ukcdr.