Clustered Requirements in System Engineering Project Estimation

Radek Silhavy, Petr Silhavy, and Zdenka Prokopova

***Abstract*—** The requirements engineering is mandatory phase which all development process start with. Mistakes in requirements elicitation therefore take very important role in a project success. In these article requirements elicitation methods are described in context of the system development and finally the clustered requirements are used for project estimation and the generic requirements engineering process is described.

***Keywords*—**System engineering, system modeling, requirements, estimation, use case points.

I. Introduction

The system engineering is important discipline, which takes key role in the scientific and engineering area. Because of the system are more and more complicated.

The set of the system requirements describes the functionality of a system, its goals and constrains, which are applicable to the future system. The requirements engineering process takes very important role in the system engineering. Many systems are become software centric and in this background appropriate design of the functionality are more than important. The system engineering as a discipline covers a broad number of subject at present time. It can focused on military systems [11], robotics [12,13] or management systems [14].

The aim of this contribution is to introduce and discuses benefits of employing the requirements engineering techniques in the system engineering.

According to [2, 3, and 4] the system design projects have very often important issues. The factors, which are the most problematic, are cost of the project, delays in terms and technical issues. The requirements engineering is the phase, which takes probably the most responsibility of named issues in the projects.

Manuscript received June 20, 2011. This work was supported by the Ministry of Education, Youth and Sports of the Czech Republic under the Research Plan No. MSM 7088352102.

R. Silhavy, Tomas Bata University in Zlin, nám. T. G. Masaryka 5555, 760 01 Zlín, Czech Republic (phone: +420 57 603 5015; e-mail: rsilhavy@fai.utb.cz).

P. Silhavy, Tomas Bata University in Zlin, nám. T. G. Masaryka 5555, 760 01 Zlín, Czech Republic (phone: +420 57 603 5015. e-mail: psilhavy@fai.utb.cz).

Z. Prokopová, Tomas Bata University in Zlin, nám. T. G. Masaryka 5555, 760 01 Zlín, Czech Republic (phone: +420 57 603 5011. e-mail: prokopova@fai.utb.cz).

The organization of this contribution is as follows. Chapter 2 describes the requirements classification. Chapter 3 discusses the methods of requirements documentation. Chapter 4 describes the gathering of requirements. In the chapter 5 is the discussion of the generic requirements engineering process. Finally chapter 6 is a conclusion.

II. Requirements definition

The requirement [1], [15] is a description of the functionality or condition which stakeholders define for the system. The requirements should be classified under the several criteria. Firstly is talked about the raw requirements are list of functionality or condition for the proposed system, which is unanalyzed yet. The most important in this phase is to establish the project goals, which should be achieved.

The next group of the requirements is non-functional requirements. The purpose of these requirements is familiarized system designers with problem domain and conditions in the domain. Well-known examples of these are reliability, performance, safety or security. These non-functional requirements are critical in the system evaluation very often.

The next group is so-called system characteristics. The system characteristics are commonly prepared in negative way. It means, that system design specifies what irrelevant system behavior is.

Constraint requirements are used for set limits upon the design alternatives the systems. No matter how the problem is solved the constraint requirements must be adhered to.

The appropriate requirement is unitary and atomic; it means describes strictly individual part. Secondly the requirement has to be complete and consistent. This is very important in context of requirements contradiction.

The requirements have to be verifiable. The implementation of the requirement can be determined through inspection or in form of some tests cases.

III. Requirements documentation

The requirements should be documented in form of free text, in form of structured text and in graphical notation.

The form of free text is common, but brings a many of misinterpretation issues [5]. In the free text description the technical jargon or acronyms have to be omitted.

The structured text and graphical form are commonly used

together.

For user goals or for overall overview of the user needs, technique which is adopted from the agile methodologies is used. In the extreme programming, which belongs to the agile methodology group, user stories are used. In the extreme programming user takes important role in the software development.

User stories are forms or cards which contains goal of selected task. The tasks are described in free language without any internal structure. When user stories are written, vague subjects, adjectives, prepositions, verbs and subjective phrases are avoided.

A. Graphical Method - SysML

A requirement as graphical entity [6] is based on class model node, which its own stereotype, called requirement. This can be seen on the figure No. 1.

The requirements modeling constructs [6] (all examples and definition are adopted from this standard - [6]) are intended to provide a bridge between traditional requirements management tools and the other SysML models.

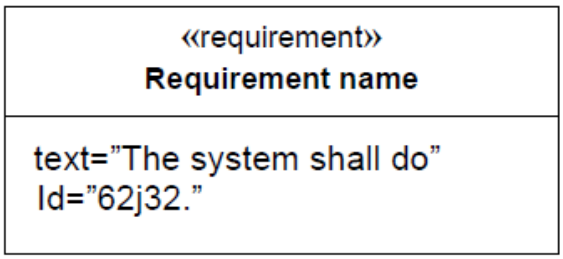

Fig 1: Individual Requirement Entity

A requirement [6] is defined as a stereotype of UML Class subject to a set of constraints. A standard requirement des properties to specify its unique identifier and text requirement. Additional properties such as verification status can be specified by the user.

Several requirements relationships are specified. These include relationships [6] for defining a requirements hierarchy, deriving requirements, satisfying requirements, verifying requirements, and refining requirements.

A composite requirement can contain sub requirements in terms of a requirements hierarchy. This relationship enables a complex requirement to be decomposed into its containing child requirements. A composite requirement may state that the system shall do A and B and C, which can be decomposed into the child requirements that the system shall do A, the system shall do B, and the system shall do C.

An entire specification can be decomposed into children requirements, which can be further decomposed into their children to define the requirements hierarchy.

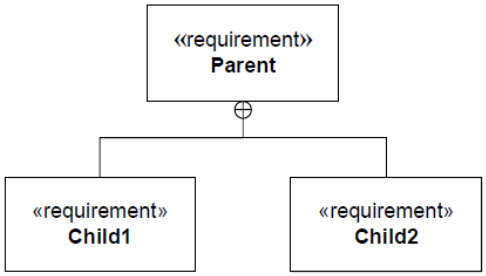

Fig 2: Containment Relationship

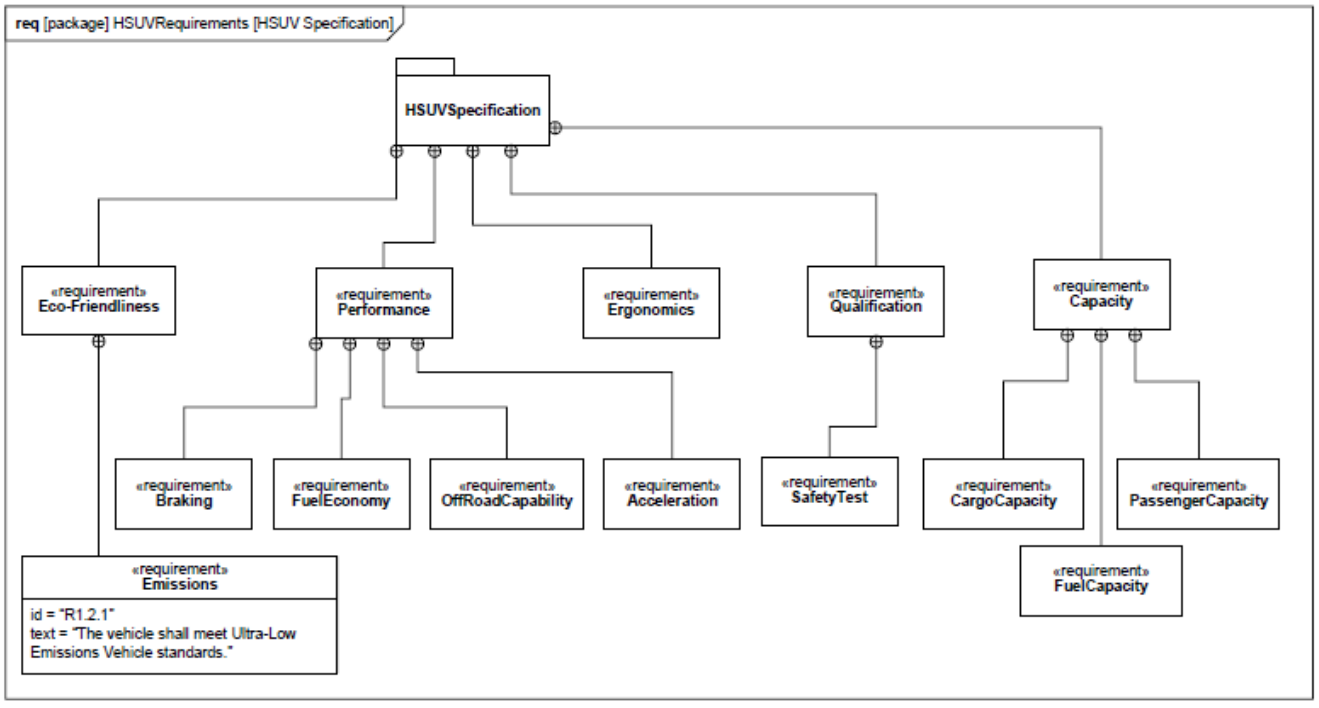

Fig 3: SysML Requirements model

The "derive requirement" relationship relates a derived requirement to its source requirement. This typically involves analysis to determine the multiple derived requirements that support a source requirement. The derived requirements generally correspond to requirements at the next level of the system hierarchy.

The verify relationship defines how a test case or other model element verifies a requirement. In SysML, a test case or other named element can be used as a general mechanism to represent any of the standard verification methods for inspection, analysis, demonstration, or test. There can be tagged values – for example verification status.

IV. Requirements Gathering

The selected methods of the system requirements gathering are described in this chapter. There are many research methods, which were adopted or modified for requirements gathering or elicitation [7].

The Interviews; very common in either in the scope of system and software engineering [1, 7]. The interview is origin for social science research. It is human based activity. The requirements engineer has to be able to discuss with stakeholders or with future system users.

The interview has three basic types; structured, semi-structured and un-structured. The most valuable for the requirements definition is the un-structured interview. Contrastingly, un-structured interview have to be held by some experienced engineer. The structured interview commonly leads to missed some of important requirements. The set of questions is not well prepared and structured.

As well-working compromise; the semi structured interview, where basic set of the question is prepared and used.

The brainstorming is very useful addition to the semi-structured interview [7]. If there are more then on stakeholders, the whole group is questioned in same time. The answers and discussions of the all group are noticed.

The laddering [7] is a technique, which is used as part of the brainstorming. It allows moderating the debate and brings hierarchical structures of the stakeholder (or stakeholders) answers.

The questionnaires; is probably the most important impersonal method. It is very useful in preliminary [7] requirements gathering. The questionnaires lack in discovering new facts or dimensions of the proposed system. Therefore have to be prepared by an experienced requirement engineer with huge knowledge of the problem domain.

The task analysis; an activity which is based on task decomposition [7,8]. The top-level tasks are designed first and then by top-down approach all sub-task are derivate and described. In the task analysis, users' tasks and system's tasks are at same level of importance. These tasks can be documented in form user or system stories, which were already described in the chapter 3. The individual task should be resulted in more than one story, even further in more than one requirement.

The observation; a method, which comes to system engineering from the ethnography and other sociological research. The observation allows observing users in their own environment. This method is valid for human-centric system design. For observation are valuable hidden cameras, with respect to privacy. Users very often change their behavioral, if they are informed about observation.

The prototyping is one of the most valuable solutions for requirements capturing. The prototyping has no value for early phase of the requirements engineering. It allows determining very concrete and detailed requirements, in the time, when introductory requirements are already collected.

V. Requirements Engineering Process

The requirements engineering process represents a formal view of all necessary activities to achieve a

Complete and balanced system requirements.

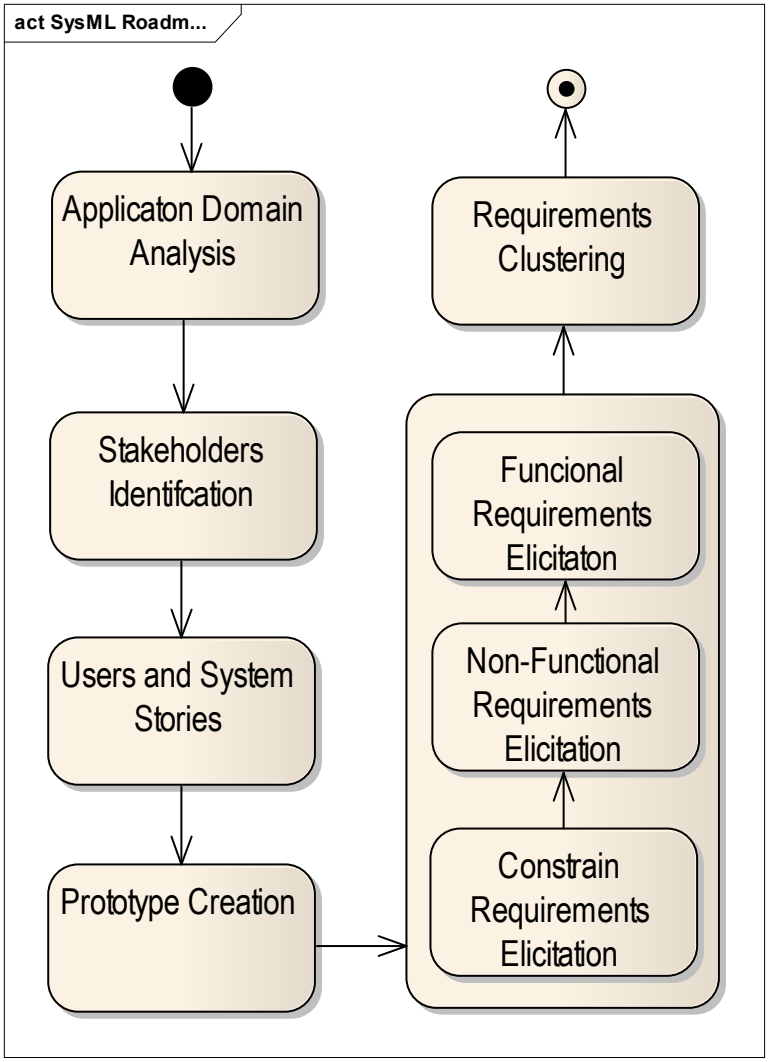

Fig 4: Requirements gathering process

The requirements engineering process contains these steps:

1.Application Domain Analysis
2.Stakeholders Identification
3.Users and System Stories
4.Prototype Creation
5.Functional Requirements Elicitation
6.Non-Functional Requirements Elicitation
7.Constrain Requirements Elicitation
8.Requirements Clustering

The application domain analysis is very important phase. During this phase all of the environmental aspects are determined. Understanding of the domain concepts is very important for next steps.

The stakeholders Identification is next logical step, which is based on the domain analysis. In this step is necessary to

resolve budget holder and all users of the modeled system. The list of stakeholder is created and then used for interview or questionnaires.

The domain analysis and stakeholder's analysis are then used for creation of user and system stories.

These basic stories are then reworked in form of prototype. This prototype is used in our proposed methodology for requirements gathering. We expected that prototype is appropriate for functional, non-functional and for constrain requirements elicitation.

In the phase No. 5, 6, 7 methodology offers interview in the form of brainstorming, which is moderate by laddering.

The final phase of proposed approach is requirements clustering, in which system modules are designed.

VI. Estimation based on Use Cases

In this article we present a system engineering, which is based on SysML langauge and requirements gathering process.

The clustered requirements are mapped to use cases.

Use Case is used for the system function description. This model is composed of the actors, which are external entity and of the use cases. The use cases represent system functions or

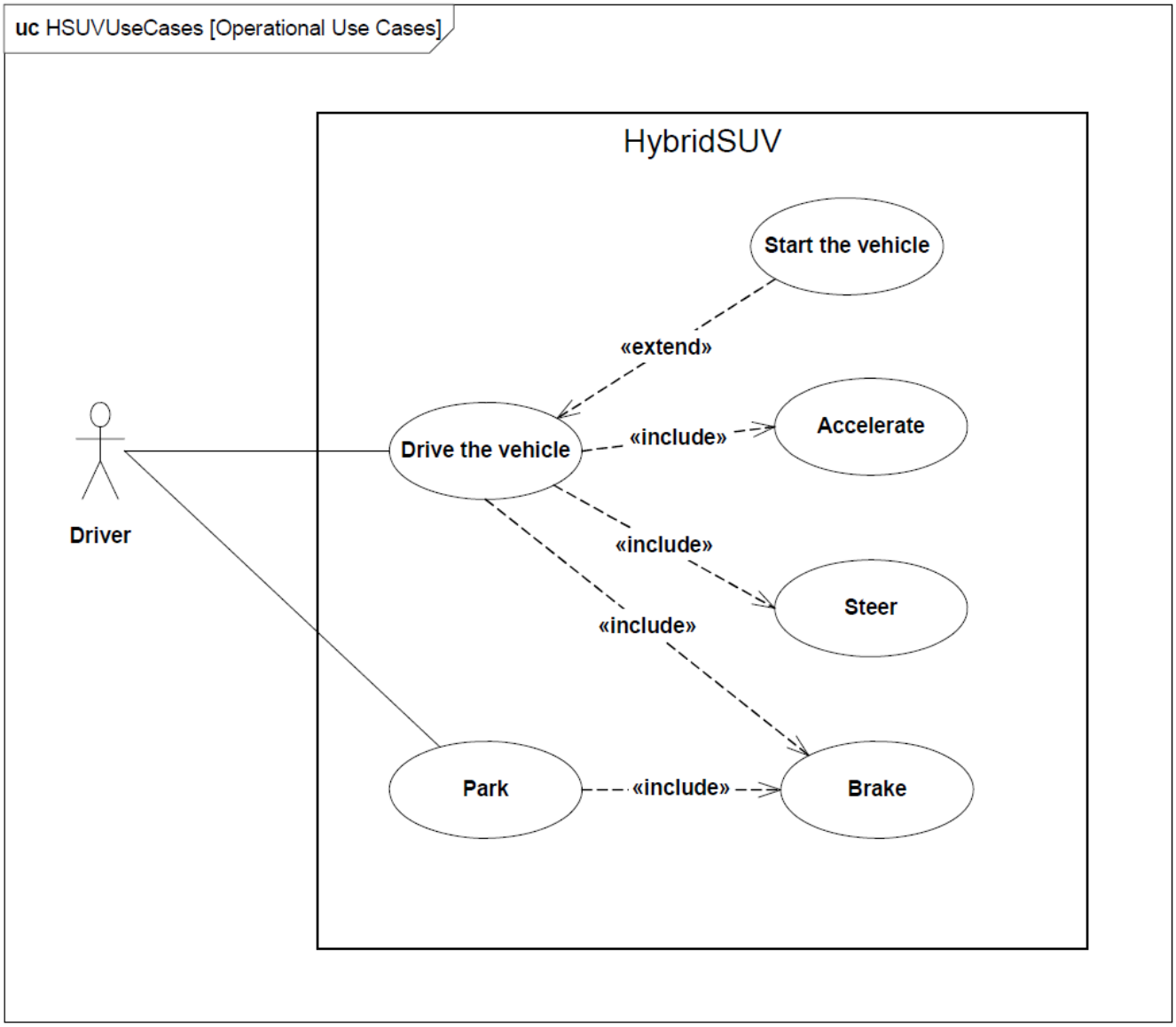

Fig. 5: Use Case Model Sample [6]

algorithms. Each of the use case has to realize one of the requirements as minimum.

The Use Case is description of the activity of the action in the system. The use case model in the system engineering consist of the actors, use case and associations.

An use case is written in form of the scenario. The scenario represents sequence of the steps, which represents using of the system by an actor.

The actors and scenarios are the base factor in the estimation methodology, called use case points.

Scenarios provide a brief description of an activity of an actor in a system. Other common definition describes a scenario as internal part of a business process, which is solved by a system itself.

For purpose of estimation number [9,10] of steps in the scenario is significant. The number of steps represents the scenario or use case complexity, therefore a system complexity is represents by the number of use cases.

For analyzing the system complexity is necessary to analyze the use cases, which were created based on clustered requirements. The quality of use cases is important for the correct estimation without occurring error.

The estimation is based on[9,10]:

1) The number of steps to complete the use case.
2) The number and complexity of the actors.
3) The technical requirements of the use case such as concurrency, security and performance.
4) The development team experience,

The generic use case point methods [9,10] is described as following steps:

1.Technical Complexity Factor.
2.Environment Complexity Factor.
3.Unadjusted Use Case Points.

B. Technical Complexity Factor

Thirteen standard technical factors exist to estimate [9,10] the impact on productivity that various technical issues have on an application. Each factor is weighted according to its relative impact. A weight of 0 indicates the factor is irrelevant and the value 5 means that the factor has the most impact.

Table 1: Technical factors

Technical Factor	Description	Weight
T1	Distributed system	2
T2	Performance	1
T3	End User Efficiency	1
T4	Complex internal Processing	1
T5	Reusability	1
T6	Easy to install	0,5
T7	Easy to use	0,5
T8	Portable	2
T9	Easy to change	1
T10	Concurrent	1
T11	Special security features	1
T12	Provides direct access for third parties	1
T13	Special user training facilities are required	1
T14	Sociotechnical system	2
T15	Business Critical	2

C. Environmental complexity factor

Environmental Complexity estimates the impact on productivity that various environmental factors have on a system.

Each environmental factor is evaluated and weighted according to its perceived impact and assigned a value between 0 and 5. A value of 0 means the environmental factor is irrelevant for this project; 3 is average; 5 means it has strong influence.

Table 2: Environmental Complexity Factor

Environmental Factor	Description	Weight
E1	Familiarity with System Designing	2
E2	Systém Domain Experience	1
E3	SysML Experience	2
E4	Lead analyst capability	2
E5	Motivation	1
E6	Stable Requirements	2
E7	Sub-Contractors	-2
E8	Difficult Integration	2
E9	Ecological Evaluation	-2
E10	Public Importancy	-2

D. Use case classification

The Use Cases are clustered into three sets. Simple, Average and Complex.

The simple set contains use cases, which have software based interface and main path scenario has 4 steps as a maximum and implementation contains of 2 subsystems.. Calculation value is 5.

The average set contains use cases, which have user interface, data processing and main path scenario has 8 steps as a maximum and implementation is group of 5 subsystems. Calculation value is 10.

The complex set contains use cases, which involves a

technology control, mechanical user interface or very complex data processing. Main path has Over 8 steps; its implementation involves more than 5 subsystems. Calculation value is 15.

E. System Actor Clasification

Actor are clustered in same sets as use cases. Sets are simple, avarage, complex.

Simple. The Actor represents another system with a defined standardized interconnection. Calculation value is 1.

Average. The Actor represents another system interacting through a protocol, like TCP/IP. Calculation value is 5.

Complex. The Actor is a person interacting via an software or mechanical interface. Calculation value is 10.

F. Example of calculation

The recapitulation of the example in the figure 5, can be found in the table 3.

Table 3: Actor and Use Cases Characteristic

Actors:	Complexity	Weight	Number
Driver	Complex	10	1
Use cases:			
Drive the vehicle	Complex	15	1
Park	Avarage	10	1
Start the vechicle	Simple	5	1
Accelarete	Avarage	10	1
Steer	Avarage	10	1
Brake	Avarage	10	1

Unadjusted Use Case is 60, based on sum of weight and number.

Unadjusted Actors is 10, based on sum of weight and number.

Significance of the technical and environmental factors should set as 0 * 10, where 0 has to effect and 10 represents the most significant factor.

Summary of the technical and environmental factor can be found in the tables 4 and 5.

Table 4: Technical factors for the example

Technical Factor	Weight	Effect
T1	2	5
T2	1	0
T3	1	5
T4	1	10
T5	1	0
T6	0,5	0
T7	0,5	0
T8	2	10
T9	1	0
T10	1	5
T11	1	5
T12	1	5
T13	1	5
T14	2	5
T15	2	5

Technical Factors are calculated as TTF: 0.6+(0.01*GlobalTechnicalFactor. The global technical factor is calculated as sum of Weight*Effect for each T.
For sample project TTF is 1.45.

Table 5: Environmental factors for the example

Environmental Factor	Weight	Effect
E1	2	5
E2	1	10
E3	2	0
E4	2	10
E5	1	0
E6	2	0
E7	-2	5
E8	2	0
E9	-2	5
E10	-2	5

Environmental Factors are calculated as ETF: 1.4+(-0.03*GlobalEnviromentalFactor. The global environmental factor is calculated as sum of Weight*Effect for each E.
For sample project ETF is 1.1.

Final result of calculation, the number of use case points is calculated as follows:
(Unadjusted Use Case + Unadjusted Actors) * TTF * ETF. In our example, total number of use case points is 111.65.

Final estimation is based on number of working our per 1 use case point. It is usually set approx. 30 hours per 1 point.

VII. Conclusion

The idea of the contribution was to introduce System Modeling Language for modeling system behavior. The system engineering were described and connection between system modeling language diagrams and the system engineering phases were illustrated.

The modeling project support analysis, specification, design, verification and validation of systems. SysML therefore support all phases of the system engineering lifecycle.

The SysML uses number of diagram, which allow to a system designer model the proposed system in many views.

The system behavioral modeling deals with requirements engineering, use cases elicitation and state modeling of the system.

The proposed requirements gathering model leads to clustered requirements, which are used for mapping use cases. The scenarios in use cases are used for system estimation.

Further research in system modeling is focused on the improvement of the appropriate calculation values, which probably the most important in the estimation factors.

Further research will be focused in improving accuracy of technical and environmental factors.

Acknowledgment

This work was supported by the Ministry of Education, Youth and Sports of the Czech Republic under the Research Plan No. MSM 7088352102.

References

[1]SOMMERVILLE, Ian. Software Engineering. Eighth Edition. Harlow: Pearson Education Limited, 2007. 824 s. ISBN 978-0-321-31379-9.

[2]The Standish Group, "CHAOS Chronicles v3.0." The Standish Group, Tech. Rep., 2003. [Online]. Available: http://standishgroup.com/chaos/toc.php

[3]M. van Genuchten, "Why is Software Late? An Empirical Study of Reasons For Delay in Software Development." IEEE Transactions on Software Engineering, vol. 17, no. 6.1991.

[4]H. F. Hofmann and F. Lehner, "Requirements Engineering as a Success Factor in Software Projects." IEEE Software, vol. 18, no. 4.2001.

[5]E. Kamsties, "Requirements Engineering: Dealing with the Complexity of Sociotechnical Systems Development." in Engineering and Managing Software Requirements, A. Aurum and C. Wohlin, Eds. Springer-Verlag, 2005.

[6]Object Management Group. Systems Engineering Domain Special Interest Group [online]. 2007-2011 [cit. 2011-03-20]. Available on WWW: <http://syseng.omg.org/>.

[7]Engineering and Managing Software Requirements. Berlin: Springer Berlin Heidelberg, 2005. Requirements Elicitation: A Survey of Techniques, Approaches, and Tools, s. 19-49. ISBN 978-3-540-28244-0.

[8]Carlshamre P, Karlsson J.A usability-oriented approach to requirements engineering. In: Proceedings of the 2nd International conference on Requirements Engineer-ing, April 15–18, Colorado Springs, CO. 1996.

[9]SEHLHORST, Scott . Software Cost Estimation With Use Case Points [online]. 2007 [cit. 2011-06-03].

[10]RIBU, Kirsten. Estimating Object-Oriented Software Projects with Use Cases. University of Oslo, Department of Informatics. 2001. Master of Science Thesis

[11] Balla, J. 2011, "Dynamics of mounted automatic cannon on track vehicle", International Journal of Mathematical Models and Methods in Applied Sciences, vol. 5, no. 3, pp. 423-432.

[12]Ouarda, H. 2011, "Cognitive tasks behavior of intelligent autonomous mobile robots", International Journal of Mathematical Models and Methods in Applied Sciences, vol. 5, no. 3, pp. 610-619.

[13]Tokarz, K. & Manger, C. 2011, "Geometric and rough set approach to inverse kinematics for arm manipulator", International Journal of Mathematical Models and Methods in Applied Sciences, vol. 5, no. 1, pp. 1-8.

[14]Nan, M.S., Nicolescu, C., Jula, D., Bolovan, C., Voicu, G.V. & Petre, G. 2011, "Practical aspects regarding spare parts reliability evaluation within an integrated management system", International Journal of Mathematical Models and Methods in Applied Sciences, vol. 5, no. 2, pp. 238-246.

[15]Loginov, D. 2011, "Possibilities of modeling the creative part of engineering design process using the synergetic approach", International Journal of Mathematical Models and Methods in Applied Sciences, vol. 5, no. 1, pp. 95-104.

Radek Silhavy was born in Vsetín in 1980. He received a B.Sc. (2004), M.Sc. (2006), and Ph.D. (2009) in engineering informatics from Faculty of Applied Informatics, Tomas Bata University in Zlin. He is a senior lecturer and researcher at the Computer and Communication Systems Department. His Ph.D. research was on the verification of the distributed schema for the electronic voting system. Major research interests are software engineering, empirical software engineering and system engineering.

Petr Silhavy was born in Vsetín in 1980. He received a B.Sc. (2004), M.Sc. (2006), and Ph.D. (2009) in engineering informatics from Faculty of Applied Informatics, Tomas Bata University in Zlin. He is a senior lecturer and researcher at the Computer and Communication Systems Department. His Ph.D. research was on the electronic communication and services in a medical information systems.

Major research interests are data mining, database systems and web-based services.

Zdenka Prokopova was born in Rimavská Sobota, Slovak Republic in 1965. She graduated from Slovak Technical University in 1988, with a master's degree in automatic control theory. Doctor's degree she has received in technical cybernetics in 1993 from the same university.

She worked as assistant at Slovak Technical University from 1988 to 1993. During years 1993-1995 she worked as programmer of database systems in Datalock business firm. From 1995 to 2000 she worked on position lecturer at Brno University of Technology. Since 2001 she has been at Tomas Bata University in Zlin, Faculty of Applied Informatics. She presently holds the position of associating professor at the Department of Computer and Communication Systems. Her research activities include programming and application of database systems, mathematical modeling, computer simulation and control of technological systems.