SOFTWARE MODEL COGNITIVE VALUE

Elena Chebanyuk, Krassimir Markov

***Abstract**: An approach for estimation of Software Models (SMs) from Cognitive Science point of view is outlined in the paper. The basic notion of this approach is the new term "Software Model Cognitive Value" (SMCV). Software models are represented as Unified Modeling Language (UML) [UML 2.5, 2012] diagrams that are used in Agile approach [Beck et al, 2001; Allen, 2015].*

In order to define peculiarities of SM human perception, cognitive principles of comprehension are considered in this paper. According to these principles, the peculiarities of SM comprehension in different situations when software is developed following Agile approach, are formulated.

The proposed approach offers an estimation of SM from the points of view both Software Engineering and Cognitive Science. From the Software Engineering view, characteristics of SM designing are considered. The cognitive features of SM such as its comprehension and understanding are taken into account.

The process of applying the proposed approach to choose the best type of SM for requirement analysis for project of designing 3D-graph is also outlined in this paper.

Applications of this approach and advantages of its applying for solving typical Software Engineering tasks are formulated.

***Keywords**: Software Model Cognitive Value; UML Diagram; Agile Development; Model-Driven Development; Software Lifecycle Process; Software Designing, Software Requirement Analysis; 3D-grpah.*

***ACM Classification Keywords:** D.2 Software Engineering; D.2.1 Requirements/Specifications; D.2.9 Management* - Life cycle; Software process models; *I.2.0 General: Cognitive science.*

Introduction

Human cognitive abilities have limits [Green & Blackwell, 1998]. For example, Miller [Miller, 1956] found that a person's short term memory has limited capacity to remember chunks of information [Endres & Rombach, 2003]. Modern psychology has even more sophisticated models of how memory works. Simon [Simon, 1982] argued that "bounded rationality" is an important aspect of human problem solving and design activities, in particular.

Cognitive science is concerned with understanding the brain processes aimed to accomplish complex tasks including: perceiving; learning; remembering; thinking; predicting; Inference; problem solving; decision making; planning; moving around the environment; and etc.

The goal of a **cognitive model** is to scientifically explain one or more of these basic cognitive processes, in particular, to understand how these processes interact [Thagart, 1996].

Cognitive modeling is the process of explaining human intelligence behavior by means of designing models that represent different cognitive processes [J. Olson & G. Olson, 2015].

In this paper, we outline an approach for estimation of SMs from cognitive science point of view. The basic notion of this approach is the new term **"Software Model Cognitive Value"** (SMCV).

Cognitive value is an evaluation of both the convenience of SM comprehension and understanding by humans and its design characteristics effectiveness. Factors that influence resulting SMCV meaning are analyzed in this paper.

When a large scale software project is created, the process of its creation is characterized by great amount of information to be processed. The effectiveness of execution many such operations is defined by entire information representation. When some laws of software artifacts representation are kept, it facilitates their analysis and processing.

Process of software creation has several stages that are parts of **software development lifecycle**. According to standard ISO 12207 (definition 5.1.12) "the life cycle model is comprised of a sequence of stages that may overlap and/or iterate, as appropriate for the project's scope, magnitude, complexity, changing needs and opportunities" [ISO/IEC 12207:2008(E)]. Each stage is described with a statement of purpose and outcomes development artifacts [Lassenius et al, 2015]. According to standard UML 2.5, software model is a UML [UML 2.5, 2012] diagram. One of the peculiarities of Agile approach [Beck et al, 2001; Allen, 2015] is that SMs replace other software artifacts and have both cognitive and communicative functions:

– **Cognitive functions** of SMs: using software models one can acquaint with algorithms, processes or software structure. The aspect of obtaining new knowledge depends upon SM notation and purpose of its usage.
– **Communicative functions** of SMs: one can express his understanding about software functionality, structure or algorithm, to collaborate with other stakeholders. Then these models are used with cognitive purpose. Using strict notation avoids misunderstanding.

The most widespread tools for expressing models are UML [UML 2.5, 2012] and Business Process Modeling Notation (BPMN) [BPMN, 2011].

The reason of UML and BPMN choice is that graphical representations of models (diagrams) make their comprehensions convenient for human perception. For effective comprehension of these diagrams they

are necessary to be designed considering both some comprehension patterns and principles of visualization.

Task and challenges

Task:

To propose an approach for estimation of SMCV, considering convenience of SM comprehension and designing by humans, when software is created according to Agile approach. Doing this, it is necessary to define the most valuable parameters and investigate their influence to resulting SMCV.

Also, it is necessary to propose rules for estimation of common SMCV when all software requirements are covered by means of SMs of specific type.

The proposed approach should consider both the peculiarities of human comprehension and purposes of specific software development process.

Challenges:

An application of the proposed approach will help to:

- design a model of stakeholder for leading an interview by means of comparing SMCV obtained by a candidate with etalon values;
- estimate compatibility of stakeholder with other software team members;
- choose the best SMs from a set of SMs, describing scalable project, to provide effective processing of large amount of information about software;
- ground the choice of the best SM for effective organizing of concrete software development lifecycle process;
- design rules for SM visualization from such formats as eXtensible Markup Language (XML).

Related papers

In general, when software is designed according to Agile approach, the cognitive skills of all stakeholders are important. There are some activities that involve cognitive processes, for example software artifacts comparison.

Comprehension of software engineering diagrams is studied well. For instance, Mangano et al. [Mangano et al, 2015] analyzed the role of sketches when pairs of software designers are working on design problems.

At a cognitive level, processing of UML diagrams consists of constructing (generating, transforming, and evaluating) their representations until they became precise and concrete [Allen, 2015]. This question was explored by Visser [Visser, 2006]. She defines the process of construction of cognitive artifacts that

represent a software product. Effective visualization of complex system is possible when graphical representation of this system allows comprehending a system as a set of components.

Cognitive design principles

Investigation of cognitive design principles is represented in [Tversky et al, 2006]. Authors underlined two cognitive principles, namely principles of ***congruence*** and ***apprehension***. The idea of **congruence principle** is to compare visual patterns which are known for person with new ones. Then one can recognize components of some complex structure using **apprehension principle**. Collaboration of these two principles provides a **common cognitive comprehension** of visual models.

Some examples of different models visual comprehension are also considered in the [Tversky et al, 2006]. One model is routing maps comprehension and processing. Another one is a representation of set of sequences goal-oriented actions. In order to represent goal oriented actions, processes of complex objects assembling are considered. These two examples allow considering that common verbal-oriented approach can facilitate a process of existing visual model modification. Examples of visual models are maps, in the first case, and drawings in the second case. It is difficult to estimate effectiveness of verbal description method reading the paper [Tversky et al, 2006]. Authors do not propose alternative methods of visual models comprehension. Also measurement to estimate cognitive characteristics of verbal description method absents.

Authors of the paper [Gureckis & Love, 2009] define two main principles of comprehension, namely ***direct associations*** and ***internal transformations***.

Using ***direct associations principle*** one can comprehend a sequence of patterns. The content of particular pattern from this sequence can be forgotten partially of fully, for example such situation occurs when memory is over. After comprehension of such a sequence in people's memory just common model is left. Using this model some properties of investigated object can be predicted. Such a principle is used when a sequence of movie or sound frames is comprehended.

Using ***internal transformation*** principle, one can match incoming patterns with ones that are already exist in memory. Existing patterns can be modified by means of adding or removing details. Such a principle is used when structural schemas are refined or new routes are established.

In the next section an application of these principles while UML diagrams are comprehended in different purposes and situations is considered.

Comprehension of UML diagram within cognitive principles

The cognitive mechanism, proposed in the paper [Tversky et al, 2006], is applied when a separate UML diagram is comprehended by stakeholder. According to this mechanism, SM is comprehended following the next principles (Figure 1):

1. When SM structure is recognized, every element in SM notation is matched with a set of templates that compose a notation of specific SM *(congruence principle)*.
2. Every defined template is juxtaposed with specific behavior *(apprehension principle)*.
3. The process of UML diagram comprehension as a whole consists on *uniting functionalities* of all recognized templates. This process is based on structure components processing which is made by human brain.

Figure 1. Comprehension of a separate SM by Tversky Principles

The cognitive mechanism, described by [Gureckis & Love, 2009], is applied when a sequence of UML diagrams is comprehended (Figure 2):

1. The human brain comprehends the sequence of UML diagrams according to *direct association principle.* Such a sequence can be formed from SMs that are designed in different Agile iterations or SMs that describe different software components.
2. Then, considered SM is matched with the closest SM in sequence according to *internal transformation principle*.

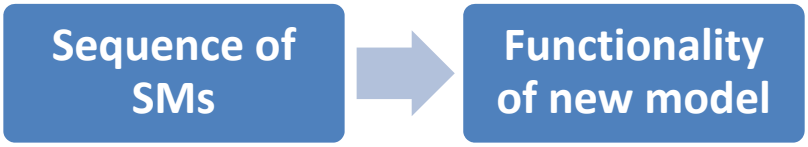

Figure 2. Matching a new SM with sequence of existing ones

Factors that influence on software models cognitive value

Different software lifecycle processes need various SMs to represent considering aspects of software with given level of details. The SMCV is characterized by a set of cognitive parameters and software designing characteristics which are different both for various SMs and process of software development lifecycle. These parameters should be integrated in a common mathematical model and this way the

SMCV may be estimated. The aim of SMCV is to consider whether the usage of this model is advisable for concrete software lifecycle development process.

Main cognitive parameters of software models are outlined below.

Cognitive value of SM depends upon **complexity of its notation** for its representation. Every type of SM has its own notation. The complexity of notation depends upon number of elements and their combination that can represent some software process or structure. The more difficult model requires more efforts to comprehend it. That is why the cognitive value of model decreases when model is expressed by means of complex notation. It requires more time to comprehend all details and more tension to memorize it. Denote the complexity of notation as ***comp***.

The parameter ***prec*** depends on the **level of representation precision** of software process (behavioral SM) or structure (static SM). More precise model contains more information about process details. Stakeholder who acquaints with SMs that allow precise representation of process or software structure can get more concrete knowledge about algorithm or architectural solution. But precise models usually represent small amount of software features.

Also for estimating the SMCV, it is necessary to consider the **time of SM designing.** When this parameter is increased the complexity of model is increased too.

Analyze the influence of every introduced parameter to SMCV. Table 1 contains information about influence of every parameter on common SMCV.

Table 1. Influence of base parameters to common SMCV

Parameter	**Estimation of SMCV when the considered parameter is**	
	increased	**decreased**
Complexity of notation	Reduced	Raised
Precision of process (structure) representation	Raised	Reduced
Scale of software functionality	Raised	Reduced
SM creation time	Reduced	Raised
The resulting SMCV for effective managing of chosen software development process	Raised	Reduced

Proposed approach

Analyzing the Table 1, define the SMCV:

$$CV_{type} = \frac{prec \cdot scale}{comp \cdot time} \quad (1)$$

where: CV_{type} – cognitive value of given type SM (types of SMs: Collaboration, State, Class and others according to UML standard);

- ***prec*** – level of precision for representation of software process or structure. This parameter is measured by means of coefficient. Matching this coefficient for every type of SM is based of subjective decision. This parameter varies from 0.1 to 1;
- ***scale*** – the number of features from software requirement specification that are covered by SM. This parameter also is measured by the following way:

$$scale = \frac{repr}{total} \quad (2)$$

where: ***repr*** – number of software requirements represented in SM;

total – number of all software functional requirements to the project.

Coefficient *scale* is defined for concrete SM, considering its tasks. This parameter also varies from 0.1 to 1;

- ***comp*** – complexity of SM notation. This parameter is defined by number of elementary components in the specific SM notation and quantity of combinations created from them. The range for this parameter is also from 0.1 to 1;
- ***time*** – is a time for one software model creation. This parameter is set for concrete specialist.

Such parameters as *prec* and *comp* are general. Values of the *time* and *scale* parameters are defined for every SM, software development lifecycle process and stakeholder separately.

Denote an amount of SMs that are necessary to cover all functionality of software requirement specification as C_{type} . That is why common cognitive value of software models of specific type is defined as follows:

$$C_{type} = CV_{type} \cdot n \tag{3}$$

where: *n* - is a number of SMs that are necessary to represent all software functionality.

When parameter C_{type} is defined it is very important to prove every SM has unique content that describes software requirements specification. An approach for defining whether the content of specific SM is unique is proposed in the paper [Chebanyuk, 2014].

Rules for estimating SMCV

Expressions (1)-(3) define SMCV from the point of software engineering view. But cognitive aspects of SM effective processing and ***human perceptional abilities*** should be considered [Green & Blackwell, 1998; Miller, 1956; Endres & Rombach, 2003]. In order to precise the proposed approach, rules ***of estimating SMCV*** for specific software development lifecycle process regarding comprehension of obtained SMCV are proposed below:

1. ***The best SM has the highest cognitive value***.
2. ***Every SM from the set of*** C_{type} ***must have unique content, namely non repeatable elements***.
3. ***Number of SM elements must be nearly to number of Miller [Miller, 1956], namely seven.***

Both the rules for estimating SMCV and mathematical apparatus (1)-(3) allows to precise the SMCV by the following:

$$CV_{type} = \frac{prec \cdot repr \cdot miller \cdot unique}{total \cdot comp \cdot time} \tag{4}$$

where:

- ***unique*** – a coefficient, defining correspondence of CVSM to the second rule. Measurements of this coefficient are proposed in the Table 2.
- ***miller*** – the coefficient, considering correspondence of CVSM to the third rule. Measurements of this coefficient are also proposed in the Table 2.

To introduce the recommended values of *miller* and *unique* coefficients (Table 2), an additional parameter *elem* – number of SM elements, is used. Respectively, for comparing two SMs: $elem_1$ - number of elements in the first SM and $elem_2$ in the second one.

Table 2. Recommended values of miller and unique coefficients

Diapason	Considered coefficient
miller	
$elem < 9$	miller = 1.0
$10 < elem < 13$	miller = 0.6
$13 < elem < 15$	miller = 0.2
$elem > 15$	miller = 0.1
unique	
$\mid elem_1 - elem_2 \mid < 3$	unique = 1.0
$3 < \mid elem_1 - elem_2 \mid < 5$	unique = 0.5
$5 < \mid elem_1 - elem_2 \mid < 7$	unique = 0.25
$\mid elem_1 - elem_2 \mid > 7$	unique = 0.15

Defining of cognitive value for different types of software models in requirement analysis process

Requirement analysis process activity is to represent exact software requirements by means of behavioral software models [ISO/IEC 12207:2008(E)]. The aim of these models is to analyze the future software system in general and to see details.

In order to assign coefficients to behavioral SMs [UML 2.5, 2012] that are used for requirements analysis process it is proposed to estimate SM characteristics of a typical software project that contains 10000 lines of code. The purpose of software is to design 3D-graph using Unity 3D and scripting language C#. Example of 3D-graph is represented on the figure 3 [Markov, 2011].

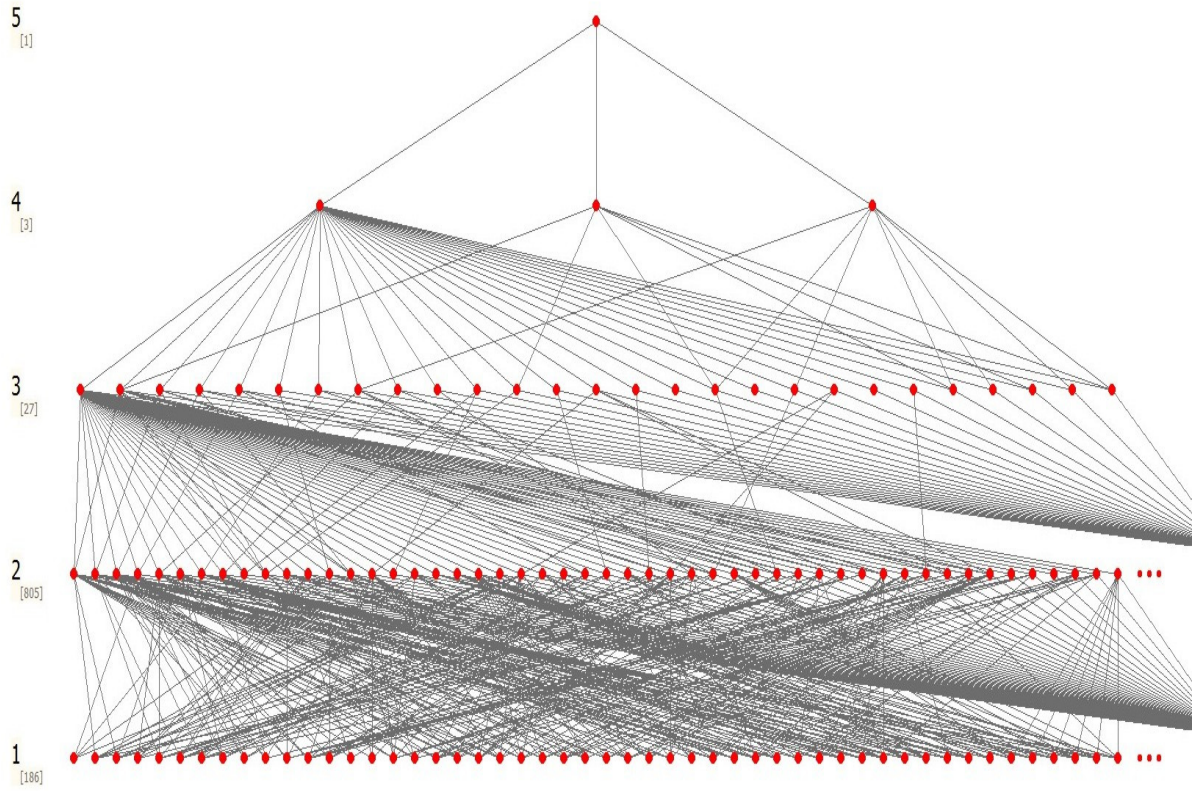

Figure 3. 3D-graph example

Ideas of 3D-graph structuring and finding routes in it are represented in [3D-graph]

Software requirement specification for this project is represented in Table 3.

Table 3. Requirement specification for project 3D-grpah creation

Requirement code	Requirement description
F1	Add different kinds of vertices to 3D-graph
F2	Move both vertices with edges and single vertices
F3	Add and remove edges connecting two vertices
F4	Save and load graph
NF1	Operation system for application working is Android

Table 4 contains information about estimation of general values for factors that influence on SMCV (1)-(4). In Table 4, general values consider the experience of creation different projects of such type.

Table 4. Estimation of parameters that influence the cognitive value of different SMs types in requirements analysis process

	Use Case		Collaboration		Sequence	
	General value	Chosen value for consi-dered project	General value	Chosen value for consi-dered project	General value	Chosen value for consi-dered project
Complexity of notation	0,1		0,5		0,7	
Precision of process representation	0,2-0,3	0,2	0,4-0,5	0,4	0,6-0,7	0,7
Scale of software functionality	0,5-1.0	1.0	0,1-0,8	0,8	0,03-0,1	0,1
Creation time (hours)	0,1-0,2	0,15	0,2-0,8	0,4	0,2-0,9	0,5
Number of software models that are needed for specific requirement analysis process	1-100	1	2-20	2	5-70	5

Represent a requirement analysis for this project. Doing this, design different types of SMs and compare their cognitive value.

According to UML standard 2.5 [UML 2.5, 2012] Use Case diagrams are used to represent general software behavior. Description of the requirement specification (Table 3) by means of Use Case diagram is given on the Figure 4.

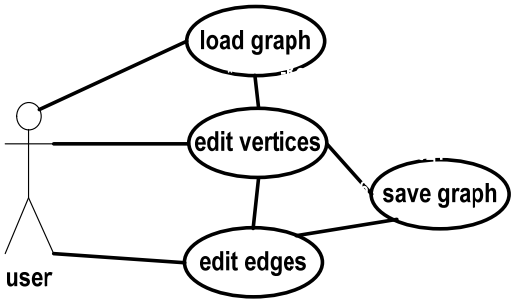

Figure 4. Description of requirement specification by means of Use Case diagram

Use Case diagram in the Figure 4 contains five elements and describes the functionality of whole software requirement specification. This description is not precise but the percent of representation of software requirements is 100%. Time for creation of this diagram is 0.15 hour. Cognitive value of this diagram is the highest because user can get information about whole functionality for project 3D-graph creation.

Consider a Collaboration Diagram for describing software requirements. These diagrams show both objects and data flows between them. Data flows are represented both by messages and conditions [UML 2.5, 2012].

Figure 5 represents the requirement specification (Table 3) by means of Collaboration Diagram notation. According to standard UML 2.5 [UML 2.5, 2012].

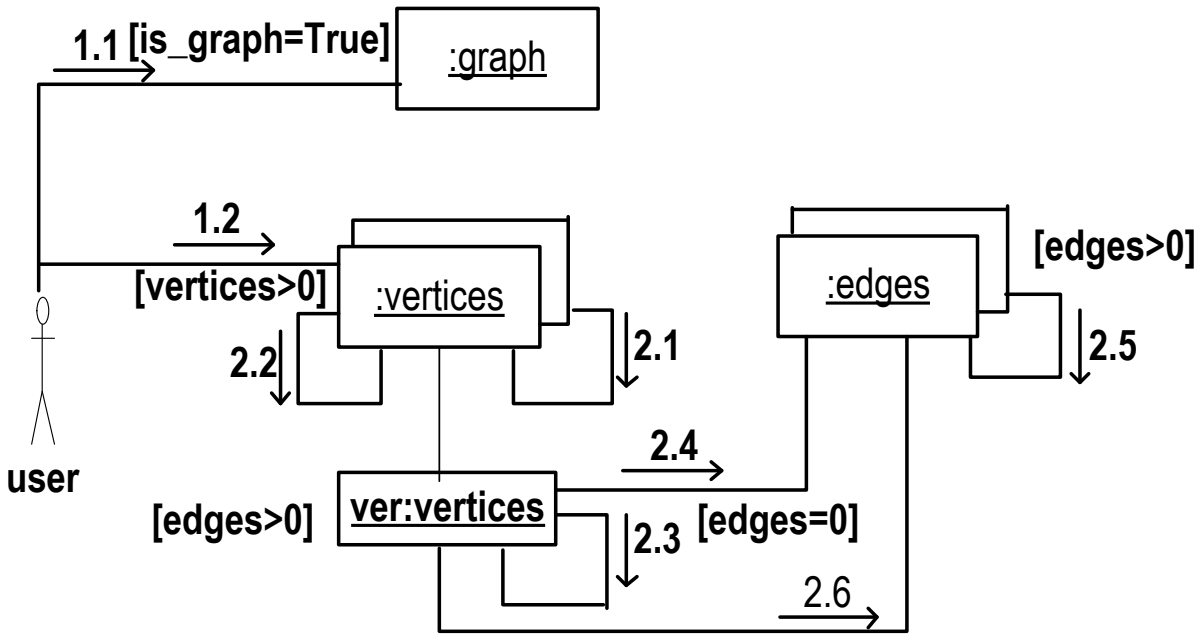

Figure 5. Description of requirement specification by means of Collaboration Case diagram

Messages for Collaboration Diagram (Figure 4) are explained in the Table 5

Table 5. Collaboration Diagram messages

Code of message	Message explanation
1.1	Load a 3D graph from file
1.2	Add vertex to 3D graph
2.1	Move vertex of 3D graph
2.2	Remove vertex of 3D graph
2.3	Remove edge of 3D graph
2.4	Move vertex of 3D graph with nested edges
2.5	Remove edge of 3D graph

Number of elements in Collaboration Diagram corresponds to number of Miller. The operation "Save graph" is not represented in this diagram (Figure 5, Table 5). Consequently, this diagram covers requirement specification on 80%. Time for creation of this diagram is 0,4 hour.

Consider a Sequence Diagram for describing software requirements, which reflects the stages of some algorithms execution in details. Main elements in Sequence Diagram notation are objects and messages between them. Also the Sequence Diagram notation allows representing such operations as conditional statements, loops, parallel execution of processes and others [UML 2.5, 2012]. Due to high level of precision for processes representation, Sequence Diagram in the Figure 6 that satisfies the human perception abilities, covers the only requirement F1 from the Table 3.

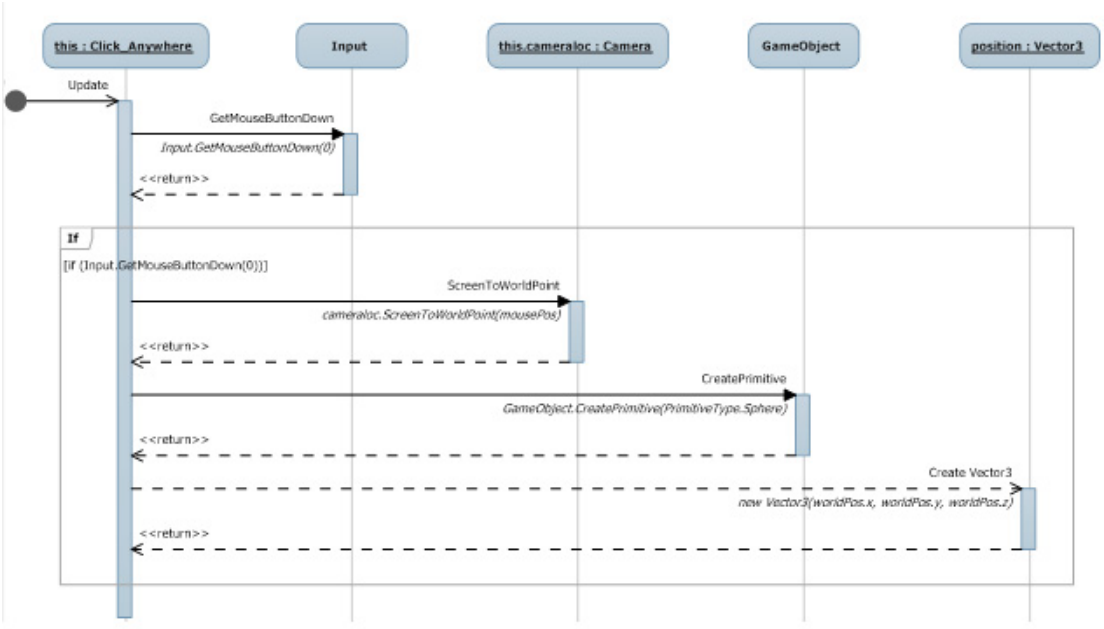

Figure 6. Description of requirement specification by means of Sequence Diagram

This SM covers 25% of software requirements (see Table 3).

Time for creation of this diagram is 0,5 hour.

Estimation of cognitive value for different behavioral SMs according (4) for the project of 3D-graph creation:

- for all SMs coefficient *unique=1* because they represent unique software requirements;
- coefficient *miller=1* (see Table 2);
- parameter *total=4* (see Table 3);
- parameter *repr*=4 for Use Case (see Figure 3), *repr*=3 for Collaboration diagram (see Figure 4), repr=1 for Sequence diagram (see Figure 5), Parameters *time* and *prec* are taken from the Table 2.

$$CV_{use\_case} = \frac{0{,}2 \cdot 4 \cdot 1 \cdot 1}{4 \cdot 0{,}1 \cdot 0{,}15} = 13{,}33 \quad (5)$$

$$CV_{collaboration} = \frac{0{,}4 \cdot 3 \cdot 1 \cdot 1}{4 \cdot 0{,}4 \cdot 0{,}4} = 1{,}87 \quad (6)$$

$$CV_{sequience} = \frac{0{,}5 \cdot 1 \cdot 1 \cdot 1}{4 \cdot 0{,}7 \cdot 0{,}5} = 0{,}35 \quad (7)$$

Analyzing the expressions (5)-(7) according to the first rule of estimation SMs, one can make a conclusion that cognitive value of Use Case diagram is more valuable in requirement analysis process.

Refer to some facts from software engineering.

When requirement analysis is done, it is necessary to manipulate with general representation of software functionality and use simple notation for understanding software tasks in general. This process is characterized by necessity of creation ***large*** amount of SMs. And also the specific of requirement analysis process that it is necessary to make a lot of changes in software models rapidly. Using complex notation and precise representation of software model can slower this process.

Then estimate C_{type} value according to (3). Parameter n is taken from the Table 3.

$$C_{use\_case} = 13{,}33 \cdot 1 = 13{,}33 \tag{8}$$

$$C_{collaboration} = 1{,}87 \cdot 2 = 3{,}74 \tag{9}$$

$$C_{sequence} = 0{,}35 \cdot 5 = 1{,}75 \tag{10}$$

The analysis of the expressions (8)-(10) shows that SMCV has the highest meaning when software functionality is described by means of Use Case diagrams.

Make a note that cognitive value of Use Case diagram is various for different software development lifecycle processes.

Conclusion

The approach for estimation the **Software Model Cognitive Value** (SMCV) is proposed in this paper. Represented model (4) considers both the characteristics of SM designing features (Table 1) and human perception (Table 2).

When SM are designed by different stakeholders, such parameters as *time, scale, repr, miller* and *unique* are changed (4). Parameter *prec* depends on the complicity of concrete SM type notation [UML 2.5, 2012].

Proposed model is extendable and can be modified by adding parameters reflecting:

- process of SM comprehension with different purposes according different cognitive principles (Figure 1 and 2).
- characteristics, specific for different software development lifecycle processes [ISO/IEC 12207:2008(E)].
- rules and recommendations for SM visualization on different screens, including mobile devices,
- operations of SM processing in Model-Driven Architecture (MDA) area [MDA, 2001]
- other software engineering tasks.

Application of this model is used to predict stakeholder's behavior for the next situations:

– to lead an interview by means of comparing SMCV obtained by a candidate with etalon values;
– to estimate compatibility of stakeholder with other software team members;
– to choose the best SMs from a set of SMs, describing scalable project, to provide effective processing of large amount of information about software;
– to ground the choice of the best SM for effective organizing of concrete software development lifecycle process.

Using the proposed approach, the process of defining the best SM (namely Use Case) for requirement analysis was represented in this paper (5)-(10). This choice matches with practical recommendations and experience of stakeholders from different software development companies. For other software development lifecycle processes different level of details for representation of software process and structure are needed. Consequently other SMs will be chosen.

Further research

Using the mathematical apparatus (1)-(4) as a ground, to design an approach for SM visualization tools, considering human cognitive abilities:

– from formats of text SM representation, such as XML;
– for effective SMs processing performing main MDA operations [MDA, 2001], namely model transformations, refactoring, merging and comparison.

Bibliography

[Allen, 2015] Edward B. Allen. Design Artifacts are Central: Foundations for a Theory of Software Engineering. Technical Report MSU-20150420. Mississippi State University, Mississippi State, Mississippi 39762 April 2015. http://web.cse.msstate.edu/~allen/Allen15MSU20150420.pdf (accesed 01.09.2015)

[Beck et al, 2001] K.Beck, M.Beedle, A.van Bennekum, A.Cockburn, W.Cunningham, M.Fowler, J.Grenning, J.Highsmith, A.Hunt, R.Jeffries, J.Kern, B.Marick, R.C.Martin, S.Mellor, K.Schwaber, J.Sutherland, D.Thomas. Agile Manifesto Copyright 2001. Access mode http://www.agilemanifesto.org/ (accessed 02.09.2015)

[BPMN 2.0, 2011] OMG Standard. Business Process Modeling and Notation (BPMN 2.0) access mode http://www.omg.org/spec/BPMN/2.0/PDF/ (accessed 05.09.2015)

[Chebanyuk, 2014] E. Chebanyuk. Framework to Manage Scrum Meeting Artifacts. In: O. Voloshyn, V. Velychko, Kr. Markov (eds.). Proceedings of the XX-th Internationa Conference "Knowledge-Dialogue-Solution" (KDS 2014). ITHEA®, 2014, Kyiv, Ukraine, Sofia, Bulgaria, ISSN 1313-0087 (printed), ISSN 1313-1206 (online). pp. 111-114.

[J. Olson & G. Olson, 2015] J. R. Olson and G. M. Olson. The Growth of Cognitive Modeling in Human Computer Interaction Since GOMS. University of Michigan. Access mode http://www.ics.uci.edu/~kobsa/courses/ICS205/03F/goms.ppt (accesed 01.09.2015)

[Endres & Rombach, 2003] A. Endres and D. Rombach. A Handbook of Software and Systems Engineering: Empirical Observations, Laws and Theories, Pearson - Addison Wesley, Harlow, England, 2003. ISBN 978-0321154200. 327 pp.

[Green & Blackwell, 1998] Thomas Green and Alan Blackwell. Cognitive Dimensions of Information Artefacts: a tutorial. October 1998.

http://www.cl.cam.ac.uk/~afb21/CognitiveDimensions/CDtutorial.pdf (accesed 01.09.2015)

[Gureckis & Love, 2010] Todd M. Gureckis, Bradley C. Love. Direct Associations or Internal Transformations? Exploring the Mechanisms Underlying Sequential Learning Behavior. Cognitive Science 34 pp. 10–50. Cognitive Science Society, Inc. 2010. ISSN: 0364-0213 print / 1551-6709 online.

[ISO/IEC 12207:2008(E)]. ISO/IEC 12207:2008(E) IEEE Std 12207-2008 https://www.iso.org/obp/ui/#iso:std:iso-iec:12207:ed-2:v1:en (accessed 02.09.2015)

[Lassenius et al, 2015] C. Lassenius,T Dingsoyr, M. Paasivaara (ed.) Agile Processes, in Software

Engineering, and Extreme Programming: 16th International Conference, XP 2015, Helsinki, Finland, May 25-29, 2015, Proceedings, (accessed 04.09.2015)

[Markov, 2011] Kr. Markov et al. Intelligent Data Processing in Global Monitoring for Environment and Security. ITHEA, 2011, Kiev, Ukraine - Sofia, Bilgaria. ISBN: 978-954-16-0045-0 (printed), ISBN: 978-954-16-0046-7 (CD/DVD), ISBN: 978-954-16-0047-4 (online). ITHEA® IBS ISC No.: 21. 410 p., 561 bibliographical references, 12 tables, 183 figures http://foibg.com/ibs_isc/ibs-21/ibs-21.htm page 367.

[Mangano et al, 2015] N. Mangano, T. D. LaToza, M. Petre, and A. van der Hoek,"How Software Designers Interact with Sketches at the Whiteboard," IEEE Transactions on Software Engineering, vol. 41, no. 2, Feb. 2015, pp. 135–156.

http://ieeexplore.ieee.org/xpl/articleDetails.jsp?reload=true&arnumber=6922572 (accessed 01.09.2015)

[Miller, 1956] G. A. Miller, "The Magical Number Seven Plus or Minus Two: Some Limits on Our Capacity for Processing Information," Psychological Review, vol. 63, no. 2, Mar. 1956, pp. 81–97. http://www.psych.utoronto.ca/users/peterson/psy430s2001/Miller%20GA%20Magical%20Seven%20Psych%20Review%201955.pdf (accesed 01.09.2015)

[MDA, 2001] Object Management Group Model-Driven Architecture standard, 2001 access mode http://www.omg.org/cgi-bin/doc?ormsc/14-06-01 (accessed 02.09.2015)

[Simon, 1982] H. A. Simon, Models of Bounded Rationality, MIT Press, Cambridge, Massachusetts, 1984, ISBN: 9780262690867. 392 pp.

[Thagard, 1996] P. Thagard Mind: Introduction to cognitive science. – Cambridge, MA : MIT press, 1996. – T. 4.

[Tversky et al, 2006] Barbara Tversky, Maneesh Agrawala, Julie Heiser, Paul Lee, Pat Hanrahan, Doantam Phan, Chris Stolte, Marie-Paule Daniel. Cognitive Design Principles: from Cognitive Models to Computer Models. In: Lorenzo Magnani, editor, Model-Based Reasoning in Science and Engineering, pp. 1–20 . 2006 http://www.purdue.edu/discoverypark/vaccine/assets/pdfs/publications/pdf/Cognitive%20Design%20Principles.pdf (accessed 01.09.2015)

[Visser, 2006] Willemien Visser, The Cognitive Artifacts of Designing, Lawrence Erlbaum Associates, Mahwah, New Jersey, 2006. ISBN 978-0805855111. 280 pp. http://www.amazon.co.uk/Cognitive-Artifacts-Designing-Willemien-Visser/dp/0805855114/ (accessed 01.09.2015)

[UML 2.5, 2012] OMG standard. Unified Modeling Language 2.5, 2012 Access mode http://www.omg.org/spec/UML/2.5/Beta1/ (accessed 02.09.2015)

Authors' Information

Elena Chebanyuk – *Software Engineering Department, National Aviation University, Kyiv, Ukraine,*

Major Fields of Scientific Research: *Model-Driven Architecture, Model-Driven Development, Software architecture, Software development.*

e-mail: chebanyuk.elena@ithea.org

Krassimir Markov – *Information Modeling Department, Institute of Mathematics and Informatics, Bulgarian Academy of Sciences, Sofia, Bulgaria*

Major Fields of Scientific Research: *Software Engineering, Cognitive Science, Information Modeling, Multi-dimensional Graph Data Bases, Business informatics, General Information Theory*

e-mail: markov@ithea.org