THE COMPUTATION INDEPENDENT VIEWPOINT: A FORMAL METHOD OF TOPOLOGICAL FUNCTIONING MODEL CONSTRUCTING

TOPOLOĢISKĀ FUNKCIONĒŠANAS MODEĻA KONSTRUĒŠANAS FORMĀLA METODE NO SKAITĻOŠANAS NEATKARĪGA VIEDOKĻA APRAKSTA IEGŪŠANAI

E. Asnina, J. Osis

CIM, problem domain, topological functioning model

1. Introduction

The first step in system modeling is to analyze the client's current situation as precisely as possible. Only after that, the new product functionality is carried out during the requirement phase. The real objective of this phase is to determine what software the client needs [1].

Today, use-case driven object-oriented techniques are driven primarily by application domain analysis, and a problem domain is regarded almost as a black box describing a number of aspects of the system. However, if we develop the software for some purposes in the real world, we must know how it will affect it [2], [3], [4].

In 2001, the Object Management Group suggested Model Driven Architecture (MDA). The key idea of the MDA is separating the specification of system functioning from the details of a way that the system uses the capabilities of its platform [5]. The MDA distinguishes three viewpoints on a system: a computation independent, a platform independent and a platform specific viewpoint. As is stated in [6], the computation independent viewpoint focuses on the environment of the system, and the requirements for the system; the details of the structure and processing of the system are hidden or as yet undetermined. In the MDA, for each viewpoint a specification is defined. For the computation independent viewpoint, it is a Computation Independent Model (CIM) that does not show details of the structure of systems. It is sometimes called a domain model and a vocabulary [6]. The CIM has to bridging the gap between the domain practitioners and experts of the design and construction of artifacts. Unfortunately, the CIM is only fuzzy defined in the MDA and its using and formal transformation to a Platform Independent Model (PIM) is not specified. However, we suppose that the CIM ought to be mathematically formalized and CIM requirements should be transformable to PIMs. We are suggesting a topological model of system functioning that completely satisfies a formalized CIM definition, which shows system functionality and the structure in a formal and comprehensive way. Formal foundation of this is given in [4].

From this viewpoint we will discuss here the formal method for constructing a topological functioning model, which is an expressive and powerful instrument for clear representation of system functioning and an environment the system works within.

2. Topological Modeling of System Functioning: a General Approach

Topological modeling of system functioning was developed at Riga Technical University [7], [8], and its application in different areas is being developed today as well [2], [3], [4], [9], [10], [11], [12], [13], [14], [15]. The topological functioning modeling is a formal approach, which is based on assumption that a complex system can be described in abstract terms as a topological space ($X<\Theta$). Where X is a finite set of properties or functional features, and Θ is a topology in the form of a directed graph. The additional information is described more in detail in [2], [4], [7], and [8].

In this paper, the formal method for construction of a topological model of functioning of a complex technical or business system from its informal verbal description is suggested. It is as follows:

1. Define a set X of physical or business functional characteristics of the considered system and other systems interacting with the system itself. This process is separated into the following steps:
   - Definition of objects and their properties from a system description
   - External system identification
   - Functional features definition in accordance with the defined actions
2. Introduce in X a topology in the form of a digraph G(X, U) indicating the cause-and-effect relations existing among elements of X (where U is a set of directed lines connecting elements of X).
3. Topological functioning model separation from a constructed topological space.

Necessary condition for the construction of such a topological space G(X, U) is a meaningful and exhaustive verbal, graphical, mathematical description of the system. The adequacy of the model to describe the functioning of a concrete system can be then achieved by analyzing the mathematical properties of such an abstract object [4], [8].

Subsequent sections describe this formal approach in detail.

2.1. Definition of Physical or Business Functional Characteristics

The process of physical or business characteristic definition from an informal description of a problem domain consists of the following steps:

- Definition of objects and their properties from a system description. In an informal system description, every noun must be underlined together with a direct object that is expressed as a numeral or a pronoun. These nouns are real world objects and their attributes (see Sect. 3)
- External system identification. A defined object list contains inner system objects as well as objects that affect this system from the external environment and that get something from the system. They can be such objects as human roles, services, products, goods etc.
- Functional features definition in accordance with the defined actions. Actions, which help to realize system functionality, can be considered as (specialized) functional features [7]. In contrast of functional features in Feature-Driven Development [16], these functional features represent the current situation in the problem domain, not client's desired requirements for the system. In the same informal system description, verbs and action preconditions or business rules are to be underlined. Each action, precondition or business rule has to be either transformed into a corresponding

functional feature or attached to the defined one. Each functional feature should represent:

- An action of the object
- A result of the action;
- An object that get a result of an action or an object that is used in this action; This could be a role, a time period or moment, catalogues etc.;

Each functional feature is to be expressed in the following form:

<action>-ing the < result> [to, into, in, by, of, from] a(n) <object>

Receiving the book from a reader

Functional features may be joined in a functional feature set, which correlates with a certain business process. A business process is a specific event in a chain of structured business activities. The event typically changes the state of data and/or a product and generates some type of output. Examples of business processes include receiving orders, invoicing, shipping products, updating employee information, or setting a marketing budget. Business processes occur at all levels of an organization’s activities and include events that the customer sees and events that are invisible to the customer. Therefore, a set of specialized functional features can be correlated with the corresponding business process. The functional feature set should be written down in the following form

<action>-ing a(n) <object>

Registering a reader

We should note that a distinction between specialized functional features and functional features is their level of abstraction.

2.2. Topology Introducing

After defining a list of functional features, cause-and-effect relations among them are to be defined or, in other words, a topology Θ must be set. As stated in [7], a topology on X is a family Θ of open subsets A_n of the set X, satisfying the two Kolmogorov’s axioms.

The cause-and-effect relations may be represented as arcs between vertices of a directed graph, which are oriented from a cause vertex to an effect vertex.

Cause-and-Effect Relations

A cause-and-effect relation between two functional properties of a system exists if the appearance of one property is caused by the appearance of the other one without participation of any intermediate property. Identification of cause-and-effect relations is rather intuitive work.

The connection between a cause and an effect is represented by a certain conditional expression, the causal implication. It is characterized by the nature or business laws or rules not by logic rules. They are such concepts as ontological necessity, probability etc. In causal connections "something is allowed to go wrong", whereas logical statements allow no exceptions. As is stated in [17], cause-and-effect relations have the following properties:

- A cause chronologically precedes an effect.
- Cause-and-effect relation is necessary: each time if there a cause exists, an effect occurs inevitably.
- A cause not only precedes an effect and always is followed by it, it causes (gives rise to, generates) and is condition on an effect. The concept of causing (generating) is necessary to distinguish from the simple consequence that is not causal.

* The causality is universal.

A cause-and-effect relation structure can form a causal chain. Causal chains begin with the first cause and follow with a series of intermediate actions or events to the final effect. “A causal chain is like a row of toppling dominoes – one event causing another, repeated until a final effect is reached” writes the author in [17]. Though one link may not be as important or as strong as the other ones, they are all necessary to the chain. If just one of there intermediate causes were absent, the final effect would not be reached.

Even if you change something, you cannot remove the effect without removing the cause. However, most cause-and-effect relationships involve multiple factors. The factors can be in series or in parallel. A complete cause in any conditions gives rise to its effect. However, a partial cause makes for its effect happening, and this effect is realized only in the case if this partial cause is joined with others conditions.

Some advice helps to identify cause-and-effect relations in a verbal text [18]:

* A cause makes something happen;
* An effect is what happens as a result of that cause;
* Some words and phrases can signal cause-and-effect relationships, e.g. accordingly, because, effect, in order that, since, cause, for, therefore, as a result, if…then, why, consequently, due to etc;
* Certain verbs are causative verbs, i.e. verbs that express cause-and-effect relationships (describes an effect one thing has on another);
* Some suffixes can indicate changes, causes and effects: “-ate” can mean to become, to cause (e.g. to activate), “-ation” – the result of _ing (e.g. summation), “-ize” – to make, a cause to be (e.g. terrorize) etc.

2.3. Topological Functioning Model Separation from a Constructed Topological Space

In order to define the topological model (X, Θ), we must first take into consideration a system represented by the expression (1), where N is a set of properties or functional features of the system and M is a set of properties or functional features of other systems, constituting the environment interacting with the system itself.

$$Z = N \cup M \tag{1}$$

The set M can be divided into two subsets: an input and an output subset. The input subset forms those functional features of external systems, which are causes for system functional features or other functional features of external systems. The output subset constitutes those ones that are effects of system functional features. Finally, Definition 1 can be given.

Definition 1: The finite closed set of system functional features is

$$X = [N] = \bigcup_{\eta=1}^{n} X_\eta \tag{2}$$

Where [N] is closing of the set N; X_η is an adherence point of the set N, i.e. the point, each *neighborhood* of which contains at least one point of N. Capacity of X is the number n of adherence points of N. The *neighborhood* of a topological digraph vertex *a* is a subset of the vertex *a* and all vertices, which incoming arcs outgo from the vertex *a*.

By studying a complex system, it is necessary to separate it into a series of subsystems: these amounts taking into consideration the closures of subsets of N.

A topological functioning model has topological (connectedness, closure, neighborhood, continuous mapping) and functional (cause-effect relations, cyclic structure, inputs and outputs) properties [4]. The following statements formalize these properties:

* Topological functioning model separation from a topological space
  - Statement 1: A topological space, which represents the functioning of a business or of a technical system, must be connected.
  - Corollary 1-1: The topological digraph G(X, U) of the functioning of a system cannot include any isolated vertices.
  - Corollary 1-2: Every one of business and technical systems is a subsystem of the environment.
* Cyclic structure. In every topological models of system functioning there must be at least one directed closed loop (i.e. a directed closed path).
* Continuous mapping. Some functional property of a system can be considered as a more detailed subset of specialized properties.
  - Statement 2: If some more detailed functioning system is formed by substitution of a subset of specialized properties for some functional property, then continuous mapping exists between more detailed and simple parent topological models of the same system.
  - Corollary 2-1: In the topological digraph G* (X*, U*), the direction of arcs, which join the specialized point subset nodes with other nodes, is determined by the direction of the arcs, which join the replaced point with the according nodes of the digraph G (X, U).
  - Corollary 2-2: A data lack, which sometimes arises at the composing of a topological model of functioning, can be filled up by data that are found, when models of the same type systems are being continuously mapped into the model of the system under study.

The additional information is described in [2], [3], [4], and [7].

3. A Case Study

In this section, we will consider a case study of the application of the abovementioned general approach. Let consider a small part of library work.

Let assume that the following informal, verbal description of the problem domain exists:

"A *reader* `comes` to the *library*. The *registering clerk* `receives` his *reader card*. `If` the *registration* `exists, then` the *registering clerk* `checks` a *period of validity* of the *registration*. `If` the *period of validity* `is gone, then` the *registering clerk* `offers` the *reader* `to pay for` *prolongation* of the *period of validity* of the *registration*. `If` the *reader* `has paid for` *prolongation*, `then` the *registering clerk* `writes out` a *receipt*. In the *receipt* he/she `indicates` a *date, a name and a rate of the payment*. `If` the *payment* `is done, then` a *registration period* `will be prolonged` for one year from the *payment date*. `If` the *registration* `does not exist` or it `is` a new *man* `came, then` the *registering clerk* `receives` *a man pass*. `The` *registering clerk* `may service` *a man* `only if he has` the *pass*. He `fills down` a *registering form* with a *number of the pass* and *the man's name, the man's surname, the birth date* and *the address*. Then *the registering clerk* `fills down` a new *reader card* with the *man's name, the man's surname,* *the reader's number* and *the period of registration validity*. The *registering clerk* `gives` this *reader card* to the new *reader*.

`If` the *registration* `is valid`, `then` the *reader* `looks for` a necessary *book* in the library *catalogue*. *The reader* `can search` *a book* by the *book author, by the book theme or by the book title*. *The reader* `fills down` *a request* for *the book* with *his reader's number* and *the title, the code, the publisher and the publishing date of the book*. When *a librarian* `receives` the filled *request,* `first` he/she `checks` reader's unpaid *fines*. `If` such *fines* `exist`, `then` the *reader* must `pay` the *fines* `first`. `If` *the reader* `has paid` them or he/she `has not any fine`, the *librarian* `checks` a free *copy of the book*. `If` such a *copy* `exists`, the *librarian* `takes` it from the library *book fund*. `Otherwise`, the *librarian* `fails` *the request*. The *librarian* `fills down` *a loan form* with *the book code*, *the reader's number*, *the loan date* and the *return date*. Then the *librarian* `gives` the *book copy* to the *reader*.

The *reader* `returns` a *loaned book* to the *librarian*. The *librarian* `checks` a *return date*. `If` *the return date* `is gone`, the *librarian* `calculates` and `imposes` *a fine* to the *reader*. `Then` the *librarian* `checks` *book condition*. `If` the *book condition* `is normal, then` the *librarian* `returns` the *book* in the library *book fund.* `If` the *book* `is damaged`, `then` the *librarian* `sends` it to *restoration* and `imposes` *a fine* to the *reader*.

`If` *the reader* `wants to pay for` *a fine*, `then` the *librarian* `searches` a *reader's fine*. `After` *payment* `making`, the *librarian* `writes out` a *receipt*. In the *receipt* he/she `indicates` *the date, the name and the rate of the payment*. `Then` he/she `gives` the *receipt* to the *reader*."

3.1. Definition of Physical or Business Functional Characteristics

Definition of Objects and Their Properties from a System Description

Having this informal description of the library functioning, we can identify objects and their attributes. Nouns representing objects and their attributes are emphasized in the informal description with the italic style. The identified objects and their attributes in parentheses are as follows: a man (name, surname, birth date, address), a reader's card, a registering clerk, registration (period of validity), a reader (number), prolongation, a receipt, a payment (date, name, rate), a pass (number), a registering form, a book (author, theme, title, code, publisher, publishing date, condition, copy), a catalogue, a request, a librarian, a book fund, a loan form, loan (date, return date), restoration. We do not define operations for objects, because it is not necessary in this stage. Operations for objects are to be defined further during the analysis stage.

External System Identification

In our case, the external objects are a man, a pass and a reader.

Functional Features Definition in Accordance with the Defined Actions

After object defining, the list of functional features of the system must be identified. Appropriate verbs and conditions are emphasized with the Courier new font. They define the appropriate actions.

Let take for an example the first expression "A reader comes to the library". The action is "come". The appropriate functional feature "Coming of a reader" is defined (Table 1, the 1st row), written according with the form given in Section 2.1. The “The registering clerk receives his reader card” is transformed into “Receiving the reader card from a reader” (Table 1, the 2nd row).

Let us consider the sentence “If the registration exists, then the registering clerk checks a period of validity of the registration”. Here, a cause is “if the registration exists” and an effect is “then the registering clerk…”. Correspondingly, a new functional feature that result is the status of the registration is “Checking the existence of a registration” (Table 1, the 3rd row). The same is in the case of checking the period of validity of the registration (Table 1, the 4th row).

Sentences “If the registration does not exist or it is a new man came, then the registering clerk receives a man pass” and “The registering clerk may service a man only if he has the pass” can be transformed into two functional features “Coming of a man”, “Receiving the pass from a man”, and one precondition “If the registration does not exist” (Table 1, row 9 and 10). The second sentence explains that a registration is possible only if a man has his pass.

Table 1. The functional feature list, where EE means the External Environment

N.	Functional features	Preconditions or business rules
1.	Coming of a reader	
2.	Receiving the reader card from a reader	
3.	Checking the existence of a registration	
4.	Checking the period of validity of a registration	[if the registration exists]
5.	Offering the prolongation of a registration	[if the period of validity is gone]
6.	Writing out the receipt for a payment	[if the reader has paid] [after payment making]
7.	Making the payment by a reader	
8.	Prolonging the period of validity of a registration	
9.	Coming of a man	
10.	Receiving the pass from a man	[if the registration does not exists]
11.	Filling down the registering form for a man	
12.	Filling down the reader card for a man	
13.	Giving the reader card to a man	
14.	Looking for the book in a catalogue	
15.	Filling down the request for a book	
16.	Receiving the request for a book	
17.	Failing the request for a book	
18.	Checking the unpaid fines by a reader	
19.	Checking the book by a request	[if the reader has not any unpaid fine]
20.	Taking the book from a book fund	[if the book exists and is free]
21.	Filling down the loan form by a reader	
22.	Giving the book to the reader	
23.	Returning the book by a reader	
24.	Checking the return date of a book	
25.	Checking the condition of a book	
26.	Calculating the fine to a reader	[if the return date is gone]
27.	Imposing the fine to a reader	[if the book is damaged]
28.	Sending the book to restoration	
29.	Returning the book into a book fund	
30.	Searching the fine by a reader	[if the reader wants to pay for a fine]
31.	Giving the receipt to a reader	
32.	Existing of a man (EE)	
33.	Existing of a pass (EE)	

External system existence must be added to the functional feature list, too (Table 1 row 32 and 33). If during this stage the list of functional features is refined, the system informal description also must be correspondingly extended.

Table 1 shows the final list of functional features, which are defined from the abovementioned informal verbal description of the problem domain.

3.2. Topology Introducing

Setting the topology Θ in the set of functional features from Table 1, we get a digraph that is illustrated by Figure 1. The number of each vertex in Figure 1 is equal to the appropriate row number in Table 1 (e.g. the vertex 1 and the 1st row).

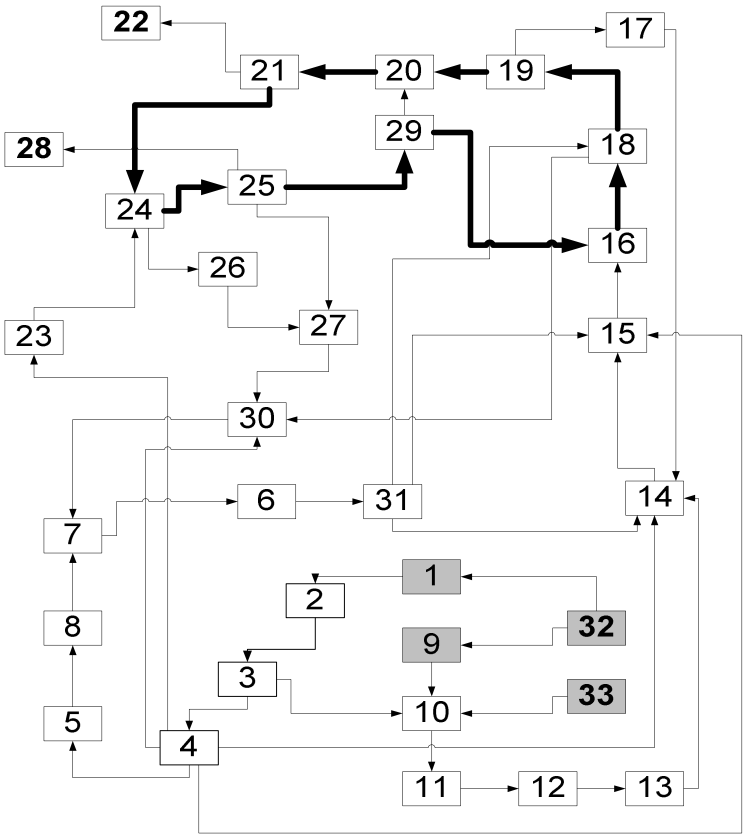

Figure 1. A topological space of the library

Let us describe a corresponding process of cause-and-effect relation identification. First, a chronological sequence of actions must to be defined. Then, causes or effects for each functional feature must be discovered. Let us take for an example the functional feature “Checking the period of validity of a registration” (the 4th row in Table 1 and the vertex “4” in Figure 1). This functional feature is chronologically preceded by the functional feature “Checking the existence of a registration” (the 3rd row in Table 1 and the vertex “3” in Figure 1). The result of this action, i.e. existence of the registration, causes execution of the 4th functional feature. Note that if the registration does not exist, then the 4th functional feature will not be executed.

Additionally, let consider an example of a causal chain. Figure 1 shows the functional feature chain “7-6-31” that describes the payment for something. In accordance with library business rules, the payment is completed only if the receipt is given to a reader. Thus, even if the cause-and-effect relation exactly between features 31 (“Giving the receipt to a reader”) and 14 (“Looking for the book in a catalogue”), which can be the first cause for the book request chain, is not evident at first sight, it becomes clear, when we think about a reader making a new book request after paying.

In the end, let consider the relation between the functional feature "Filling down the loan form by a reader" (the vertex 21) and "Checking the return date of a book" (the vertex 24). The causal relation between the vertex 21 and 24 exists since the functional feature "Filling down the loan form by a reader" is a partial cause for the book return date check. Another part of this complete cause is the functional feature "Returning the book by a reader" (the vertex 23). Therefore, the causal relation between the vertex 21 and 24 must be represented in the model.

As it is stressed in Statement 2 (see Sect. 2.3), a valid topological functioning model allows substitution of a functional feature by a subset of specialized properties and vice versa. According to Corollary 2-1, a simplified model must keep every cause-effect relation correspondingly to them in the detailed model and vice versa. According with the abovementioned and the modified MDA software development lifecycle [4], the feedback from the code deployment is not directed to the analysis phase like in the traditional MDA life cycle, but to the topological functioning model. Therefore, all corrections and modifications of the software product start in a formal way at the very beginning of the development cycle.

In the case of a data lack, models of the same type system may be continuously mapped into a topological functioning model according to Corollary 2-2.

In our case, to facilitate causal chain recognition or to recognize cause-effect relations, specialized functional features might be continuously mapped into existing business processes. The example of such simplification is as follows: 1) the business process "Checking a registration" covers functional features 2, 3 and 4; 2) "Making a payment" – 6, 7, and 31; 3) "Prolonging a registration" – 5 and 8; 4) "Registering a reader" – 10, 11, 12 and 13; 5) "Making a book request" – 14, 15 and 16; 6) "Loaning a book" – 17, 18, 19, 20, and 21; 7) "Returning a book" – 23, 24, 25 and 29.

3.3. Separation of the Topological Functioning Model of the Library

After constructing a topological space of the given problem domain, a topological model of system functioning can be separated from it. Thus, functional features of the system itself, the set N, and functional features of the external systems, the set M must be defined (1). In the case of our example, the set of functional features of the external systems is the set M = {1, 9, 22, 28, 32, 33}. The set of system functional features N = {2, 3, 4, 5, 6, 7, 8, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 23, 24, 25, 26, 27, 29, 30, 31}. Therefore, the topological model of system functioning may be separated from the topological space of the problem domain by applying the closure operation (2) over the set of library functional features as follows:

[2] = {2, 3}
[3] = {3, 4, 10}
[4] = {4, 14, 15, 5, 23, 30}
[5] = {5, 8}
[6] = {6, 31}
[7] = {7, 6}
[8] = {8, 7}
[10] = {10, 11}
[11] = {11, 12}
[12] = {12, 13}
[13] = {13, 14}
[14] = {14, 15}
[15] = {15, 16}
[16] = {16, 18}
[17] = {17, 14}
[18] = {18, 19, 30}
[19] = {19, 20, 17}
[20] = {20, 21}
[21] = {21, 22, 24}
[23] = {23, 14}
[24] = {24, 25, 26}
[25] = {25, 27, 28, 29}
[26] = {26, 27}
[27] = {27, 30}
[29] = {29, 20, 16}
[30] = {30, 7}
[31] = {31, 14, 15, 18}

Hence, the set X = [N] = {2, 3, 4, 5, 6, 7, 8, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31}.

Figure 1 shows input functional features {32, 33} and output functional features {22, 28}; the shadowed vertices show external functional features not included in the set X.

Note the fact that the great attention is put on the description and analyzing of functional cycles in this modeling approach [4]. Let consider the model represented in Figure 1 more in detail. This model has all abovementioned functional properties: it describes cause-effect relations, has the cyclic structure, vertex inputs and outputs. The bold-lined arcs show the main cycle of the system that connects functional features {16, 18, 19, 20, 21, 24, 25, 29, 16} (Table 1). They connect those functional features, which form the main feedback circuit of the business system; in words it can be described as the cycle “book request - book loan - book return – book request”.

Moreover, an advantage of the topological functioning system is that it shows not only the main functioning cycle, but also functioning subcycles. The author of [7] points out that the main difference between the main cycle and a subcycle is the following - “if one of the subcycles is cut up, the system is malfunctioning, but cutting up the main cycle leads to complete functional disability”.

There are sub-cycles in our example. Some of them are as follows:

- “18-30-7-6-31-18” - this subcycle describes a situation that can occur during book request, when a reader has a fine and wants to repay it; after repaying, his request will be executed.
- “20-21-24-25-29-20” – this subcycle represents the path of a book in the system from the book fund to a reader to the book fund, if it is not damaged and is returned on time.
- “15-16-18-19-20-21-24-26-27-30-7-6-31-15” – this subcycle illustrates a situation, when a reader fills down a request, then loans a book, but returns it not on time. Therefore a librarian calculates and imposes him a fine. The reader repays the fine, gets a receipt and may make a new book request.

4. Conclusions

This paper suggests the formal method for a topological functioning model construction as a formalized CIM [4]. It is illustrated by the case study of library functioning.

Having a description of processes happening in the system, the considered formal method can be applied. It consists of the following steps: 1) functional feature definition (definition of inner system objects, external systems, and system functional features); 2) cause-and-effect relation definition; 3) separation of a topological model of system functioning; 4) conceptual class structure derivation and transformation into Platform Independent Model (isn’t shown in this paper). Topological functioning modeling provides formal transformation from a topological functioning model into conceptual class diagram. Then this formalized CIM can be formally transformed into PIM constructs. This process and results are discussed more in detail in [4].

The goal of our future research is to create a tool for this formal method. Despite necessary human participation, we can facilitate model constructing process and formal model verification, and checking as topological as functional properties satisfaction of topological functioning models.

This work has been partially supported by the European Social Fund within the National Programme “Support for the carrying out doctoral study program’s and post-doctoral researches” project “Support for the development of doctoral studies at Riga Technical University”.

References

1. Schach St. R. Classical and Object-Oriented Software Engineering with UML and Java // International edition, 4th edn., WCB/McGraw-Hill, 1999.
2. Osis J. Extension of Software Development Process for Mechatronic and Embedded Systems // In: Proceeding of the 32nd International Conference on Computer and Industrial Engineering, Limerick, Ireland, University of Limerick, August 11-13, 2003, p. 305-310.
3. Osis J. Software Development with Topological Model in the Framework of MDA // In: Proceedings of the 9th CAiSE/IFIP8.1/EUNO International Workshop on Evaluation of Modeling Methods in Systems Analysis and Design (EMMSAD'2004) in connection with the CAiSE'2004, Vol. 1, RTU, Riga, 2004, p. 211 – 220.
4. Osis J. Topological Functioning Model within the MDA Life Cycle // In: the current Proceedings.
5. Architecture Board ORMSC (Miller J., Mukerji J. (eds.).): Model Driven Architecture (MDA) // ormsc/2001-07-01, www.omg.org/mda/, 2001.
6. OMG: Miller J., Mukerji J. (eds.), MDA Guide Version 1.0.1. // www.omg.org/mda/, 2003.
7. Osis J. Topological Model of System Functioning. In: Automatics and Computer Science // J. of Acad. of Sc., Riga, Latvia #6, 1969, p. 44-50 (in Russian).
8. Osis J. Diagnostics of Complex Systems (Dissertation of Dr.Habil.Sc.Eng.) // Latvian Academy of Sciences, Riga, 1972.
9. Osis J. Development of Object-Oriented Methods for Hybrid System Analysis and Design // In: Proc. of the 23rd Conference of the Association of Simula Users (ASU), Stara Lesna, Slovakia, 1997, p. 162 - 170.
10. Osis J., Sukovskis U., Teilans A. Business Process Modeling and Simulation Based on Topological Approach // In: Proceedings of the 9th European Simulation Symposium and Exhibition, Passau, Germany, 1997, p. 496 - 501.
11. Osis J., Beghi L. Topological Modelling of Biological Systems // In: Proc. of the 3rd IFAC Symposium on Modelling and Control in Biomedical Systems (Including Biological Systems), Elsevier Science Publ., Oxford, UK, 1997, p. 337 - 342.
12. Asnina E. Topological Modeling and Arrow Diagram Logic Formalism Application for Software Development // In: Scientific Papers of University of Latvia, Volume 673, Databases and Information Systems, Doctoral Consortium, Sixth International Baltic Conference BalticDB&IS 2004, Riga, Latvia, June 6-9, 2004, p. 220 – 231.
13. Asnina E. Formalisation Aspects of Problem Domain Analysis // In: Proceedings of the 8th Conference "Information System Implementation and Modelling", Hradec nad Moravicí, Czech Republic, April 19-20, 2005, p. 195-202.
14. Markovitch Z., Markovitcha I. Modelling as a Tool for Therapy Selection// In: Proc. Of the 14th European Simulation Multiconference "Simulation and Modelling", Ghent, Belgium, May 23-26, 2000, p. 621-623
15. Grundspenkis J. Causal Model Driven Knowledge Acquisition for Expert Diagnostic System Development// In: Application of AI to Production Engineering. Lecture Notes of the Nordic-Baltic Summer School'97. K. Wang and H.Pranavicius (ed.), Kaunas University Press, 1997, p. 251-268
16. Coad P., Lefebvre E., De Luca J. Java Modeling in Color with UML// Prentice-Hall, 1999, p. 182-203.
17. Farlex Inc.: The Free Dictionary Com // In: http://encyclopedia.thefreedictionary.com/causality.
18. Holt, Rinehart, Winston. Elements of Language: Examining Causes and Effects // 4th Course, p. 92-103, http://go.hrw.com/elotM/0030526671/student/ch03/lg1403092_103.pdf, 2005.

Erika Asnina, Riga Technical University, Faculty of Computer Science and Information Technology, Institute of Applied Computer Systems, Meza iela 1/3, Riga, LV-1048, Latvia, M.cs.ing, erika.asnina@cs.rtu.lv

Janis Osis, Riga Technical University, Faculty of Computer Science and Information Technology, Institute of Applied Computer Systems, Meza iela 1/3, Riga, LV-1048, Latvia, Professor, Dr.habil.sc.ing., phone nr. 371 7089523, osis@egle.cs.rtu.lv

Asņina Ē., Osis J. Topoloģiskā funkcionēšanas modeļa konstruēšanas formāla metode no skaitļošanas neatkarīga viedokļa apraksta iegūšanai
Dotajā rakstā ir aprakstīta topoloģiskā funkcionēšanas modeļa vieta un lietošana Modeļu Vadītas Arhitektūras (MDA) ietvaros. Šajā publikācijā ir piedāvāta topoloģiskā funkcionēšanas modeļa konstruēšanas formāla metode. Topoloģiskais funkcionēšanas modelis parāda sistēmu no skaitļošanas neatkarīga viedokļa kā matemātisko topoloģisko ar cēloņu un seku attiecību saistīto funkcionālo īpašību telpu. Modeļa topoloģiskās un funkcionālās īpašības nodrošina sistēmas modeļa formālo izdalīšanu no vides topoloģiskās telpas, ka arī modeļu precizēšanu un abstrahēšanu. Topoloģiskais funkcionēšanas modelis nodrošina formālo informācijas attēlošanu pašā programmatūras izstrādes sākumā, tātad, tas dod noteiktu iespēju iegūt augstas kvalitātes rezultātu MDA ietvaros. Piedāvātā metodē ļauj formāli iegūt sistēmas modeli no neformāla verbāla sistēmas apraksta. Galvenās metodes fāzes, kuras ir aprakstītas šajā rakstā, ir sekojošas: sistēmas iekšējo objektu identificēšana, ārējo sistēmu identificēšana, sistēmas funkcionālo īpašību definēšana, cēloņu un seku attiecību starp funkcionālām īpašībām identificēšana, un topoloģiskā sistēmas funkcionēšanas modeļa matemātiskā izdalīšana no sistēmas vides topoloģiskās telpas. Aprakstītā metode ir detalizēti izskaidrota un ilustrēta ar bibliotēkas funkcionēšanas piemēru.

Asnina E., Osis J. The Computation Independent Viewpoint: a Formal Method of Topological Functioning Model Constructing
This paper describes the place and usage of a topological model of system functioning in the framework of OMG's Model Driven Architecture. In this paper we suggest the formal method of constructing a topological model of system functioning. The topological functioning model shows a system from the computation independent viewpoint as a mathematical topological space of functional features interconnected by cause-and-effect relations. Topological and functional properties of the model allow formal identification of a system model from the topological space of an environment, model refinement and abstraction. The topological functioning model allows capturing information in a formal way at the very beginning of software development, thus it gives a well-defined opportunity to get a high-quality end result within the framework of MDA. The suggested method supposes formal obtaining of the system model from the informal verbal description of the system. The main stages of the method, which are described in this paper, are as follows: system inner object identification, external system identification, system functional feature definition, cause-and-effect relation identification among the found functional features, and topological model of system functioning mathematical separation from the topological space of system environment. The described method is explained in detail and illustrated by an example of library functioning.

Аснина Э., Осис Я. Формальный метод построения топологической модели функционирования с независимой от вычисления точки зрения
В данной статье описано место и использование топологической модели функционирования в рамках управляемой моделями архитектуры (Model Driven Architecture). В статье предложена формальная методика построения топологической модели функционирования системы. Топологическая модель функционирования отображает систему с вычислительно-независимой точки зрения в виде математического топологического пространства, состоящего из функциональных свойств, связанных между собой причинно-следственными отношениями. Функциональные и топологические свойства топологической модели функционирования позволяют формальным способом выделить модель системы из топологического пространства среды, а также уточнить и абстрагировать модель. Топологическая модель функционирования позволяет формально отобразить информацию в самом начале разработки программного обеспечения. Следовательно, предоставляет определенную возможность получить в рамках управляемой моделями архитектуры высококачественный конечный результат. Предложенный метод предлагает возможность формального получения модели системы из неформального вербального описания системы. Основные этапы метода, описанные в данной статье, следующие: идентификация внутренних объектов системы, идентификация внешних систем, определение функциональных черт системы, идентификация причинно-следственных отношений между найденными функциональными чертами и математическое выделение топологической модели функционирования системы из топологического пространства среды, в которой работает система. Предложенная методика детально описана и проиллюстрирована на примере функционирования библиотеки.