System Thinking for Formal Analysis of Domain Functioning in the Computation Independent Model

Erika Asnina[1], Janis Osis[1] and Asnate Jansone[2]
[1]*Department of Applied Computer Science, Riga Technical University, Meza iela 1-k.3, LV-1007, Riga, Latvia*
[2]*Institute of Applied Computer Systems, Riga Technical University, Meza iela 1-k.3, LV-1007, Riga, Latvia*

Keywords: System Thinking, Analytical Models, System Analysis and Design, Topological Functioning Model.

Abstract: A gap between two domains, the system and its supporting software, is a well-known issue in software development. The analysis of the system is often considered as a redundant unwanted activity. However, software development driven by models will not be able to close the gap, if these models focus only on software and ignore the system, since software is a subsystem that helps to conduct some system's activities. Thus, the system must be accurately analyzed before the software. For this purpose, this paper suggests a formal engineering model, Topological Functioning Model, and analysis of system functioning based on the system theory, algebraic topology, and classical logic.

1 INTRODUCTION

Software is a solution that is dedicated to address problems within the system it will operate, and which becomes its subsystem after introduction. Therefore, a problem domain includes both system and software sub-domains. Due to many reasons, the analysis of a system domain is quite superficial in software development. That leads to the well-known issue in software development. It is a gap between the system and its supporting software, i.e., between the problem and the solution (Osis, 2006), (Osis and Asnina, 2011 b).

A power of traditional engineering is a solid theory, mathematics and formal methods. A traditional civil engineer investigates a surrounding environment, analyzes requirements for an object to be built, builds mathematical models of the object, verifies them by a formal means, and only then builds a real object. Conversely, a software developer investigates a surrounding environment a little bit, analyzes thoughts (requirements) about an object to be built, and starts to build this object based on a sketch of the object (in a better case). Caper Jones was right when he said that "The way software is built remains surprisingly primitive" (as cited in Osis and Asnina, 2011 b), because it is requirements-based and chaotic. Requirements-based development differs from requirements-initiated development similarly as software engineering differs from traditional engineering. Software development must be requirements-initiated, this means that analysis and modeling of the system as a whole must precede design of the software. Modeling is a better means to deal with complexity and a large size of a system.

System thinking is based on the system theory. Paul Weiss and Karl Ludwig von Bertalanffy are the first founders of the general system theory (GST) (Drack and Apfalter, 2007), which had have many elaborations in different disciplines such as biology, cybernetics, system, social and management science. Weiss considered "it to be essential to deal with a system as a whole, because system reactions (e.g., concerning functions and development) could only be adequately understood by taking into account the system as a whole" (Drack and Apfalter, 2007). Ludwig von Bertalanffy wrote in 1969 that "A system may be defined as a set of elements standing in interrelation among themselves and with [the] environment" (as cited in Drack and Apfalter, 2007). *System thinking* has been useful in multi-project management for software development, where the proposed management model joins both Weiss' and Bertalanffy's definitions of GST in an enhanced *causal-loop diagram* (Lee and Miller, 2004). Successful applications of the system theory to operational research and management science are overviewed in (Mingers and White, 2010).

Asnina E., Osis J. and Jansone A..
System Thinking for Formal Analysis of Domain Functioning in the Computation Independent Model.
DOI: 10.5220/0004090602320240
In *Proceedings of the 7th International Conference on Evaluation of Novel Approaches to Software Engineering* (MDA&MDSD-2012), pages 232-240
ISBN: 978-989-8565-13-6

We believe that the main principle of OMG Model Driven Architecture (MDA) – architectural separation of concerns in specifications – is a step towards practical application of system thinking in software development. MDA (Miller and Mukerji, 2003) suggests three architectural viewpoints on the system, namely, a computation independent, a platform independent and a platform specific view. By considering both the software and the system, the computation independent viewpoint should be able to provide compliance of the software model with the system model (Osis and Asnina, 2008), (Asnina and Osis, 2010). This puts a strong responsibility on the computation independent model – this must be an engineering model. Characteristics of a good engineering model given in (Lavagno et al., 2004) are the following: abstraction, comprehension, accuracy, possibility to make accurate predictions about interesting properties of the modeled system from the information provided by the model, and significant inexpensiveness. Most modern models provide abstraction, understandability and inexpensiveness. But accuracy and predictability are challenges.

A topological functioning model (TFM) has all the five characteristics (Osiset al., 2007 a), (Osis et al., 2007 b) (Osis et al., 2008 a),. Its properties and application as a computation independent model in the context of MDA are described in the below mentioned sources. Because of its holistic formal nature, the TFM is a means for verification of requirements completeness (Osis et al., 2008 b), determination of shared functionality and derivation of use cases (Osis and Asnina, 2011 a), (Osis and Asnina, 2011 c), integration of system knowledge that usually are expressed as a set of interrelated fragments (Slihte et al., 2011), and derivation of a system's structure (Osis and Donins, 2010), (Donins et al., 2011) and behaviour (Asnina and Osis, 2011).

This paper discusses two of the above mentioned engineering model characteristics, namely, accuracy and predictability. Accuracy and predictability of the software model should be based on a solid theory, which application should be elaborated in formal methods, and results of application of these formal methods should be formally presented. System thinking and a formal model, TFM, can address these challenges. Section 2 presents theoretical foundations of the TFM and analysis of domain functioning. Section 3 illustrates application of the suggested approach on an example. Section 4 discusses related work. Conclusions discuss the obtained results and directions of future research.

2 SYSTEM THINKING, DOMAIN FUNCTIONING AND CAUSAL RELATIONS

The "traditional" way of problem domain analysis is when developers explore the problem domain by small parts, at the beginning trying to understand each fragment of the problem domain and only after that trying to join those fragments together in the joined and more formal representation. The alternative way is *system thinking* that is based on a system theory. It anticipates that interdependency of fragments is understood before analysis of each particular fragment. This interdependency may have dynamic and structural nature. In order to simplify problem analysis "reduction to dynamics" and "reduction to components" are applied (Laszlo and Krippner, 1998). This section discusses the alternative way of simplification; it could be called "reduction to functional dependence" that joins both the mentioned reductions. Section 2.1 introduces the TFM and its implementation of the system theory in brief. Section 2.2 and 2.3 illustrate two main elements of the TFM, a functional feature and a cause-and-effect relation, which are responsible for accuracy and predictability of a system model. Section 2.4 discusses improvement of accuracy and predictability of the model.

2.1 Topological Functioning Model in Brief

The TFM is based on principles of algebraic topology and system theory. Mathematically, the TFM is represented in the form of a topological space (X, Θ), where X is a finite set of functional features (characteristics) of the system under consideration, and Θ is the topology that satisfies axioms of topological structures and is represented in the form of a directed graph (Osis, 1969). Properties of topological spaces are described in detail in (Osis, 2006), (Osis, 2004), (Basener, 2006).

The process of construction of the TFM consists of definition of system's functional features, cause-and-effect relations among them, and separation of the TFM from the topological space of the system. The details are described in (Osis et al., 2008 b), (Osis and Asnina, 2011 c), (Donins et al., 2011). The stage we consider here is related to determination of cause-and-effect relations.

The TFM has topological (come from algebraic topology) and functioning (come from system theory) properties. The topological properties are *connectedness, closure, neighborhood and continuous mapping*. The functional properties are *cause-and-effect relations, cycle structure, inputs and outputs* (Osis and Asnina, 2011 d). This section focuses on TFM functional properties.

2.2 Functional Features

A functional feature is a characteristic of the system that is designed for achieving some system's goal. The functional feature is an activity that helps the system in conducting its functionality. Functional features can be joined in a functional feature set that represents a certain business function (Osis, 1969).

A functional feature is defined as a unique tuple <Name, PrCond, PostCond, Pr, Ex>, where (Asnina and Osis, 2010)*:*

- **Name** that consists of a tuple <A, R, O>, where A is an action linked with an object; R is a result of that action (it is an optional element); and O is an object (objects) that get the result of the action or an object (objects) that is used in this action; it could be a role, a time period or a moment, catalogues etc.;
- **PrCond** is a set PrCond = $\{c_1, \ldots, c_i\}$, where c_i is a precondition or an atomic business rule (it is an optional element) of the action;
- **PostCond** is a set PostCond = $\{c_1, \ldots, c_i\}$, where c_i is a post-condition or an atomic business rule (it is an optional element) of the action;
- **Pr** is a set of responsible entities (systems or subsystems), which provide or suggest the action with the set of certain objects;
- **Ex** is a set of responsible entities (systems or subsystems), which enact the action.

Functional features are connected by causal relations.

2.3 Cause-and-effect Relations

Cause-and-effect relations connect functional features and may form *functioning cycles*. A *cause* functional feature must have at least one effect, as well as an *effect* functional feature must have at least one cause. At the same time, causes and effects are stimulus sent to the system by the external environment *(inputs)* and reactions sent to the external environment by the system *(outputs)*. In other words, a cause-and-effect relation is a control flow from one functional feature to another one.

The formal specification of a cause-and-effect relation is a unique tuple <C, E, N, S>, where:

- **C** (cause) is a functional feature that generates functional feature E, this may not be empty;
- **E** (effect) is a functional feature that is generated by functional feature C, this may not be empty;
- **N** is the necessity of the functional feature C for generating the functional feature E; the values are *true* or *false*;
- **S** is the sufficiency of the functional feature C; the values are *true* or *false*.

Necessity N and *sufficiency S* are concepts of classical logic; they induce substantial and consistent effects on conditional reasoning performance. The *necessity* of the cause is determined when the occurrence of the effect indicates the occurrence of the cause. The *sufficiency* of the cause is determined when the occurrence of the cause indicates the occurrence of the effect. The *necessary and sufficient* cause is when the occurrence of the effect is possible <u>if and only if</u> the cause occurred, and occurrence of the effect indicates the obligatory occurrence of the cause.

Identification of cause-and-effect relations is *intuitive work* based on a modeler's knowledge and understanding of system operation. As stated in (Osis, 1969) "it is assumed in topological functioning modeling that a cause-and-effect relation between two functional features of the system exists if the appearance of one feature is caused by the appearance of the other feature without participation of any third (intermediary) feature." In terms of classical logic, this means that *a cause must be either sufficient, or both necessary and sufficient.* However, cases when a single cause is both necessary and sufficient are not often in complex domains. More often cases are when *a combination of causes* is either sufficient, or both necessary and sufficient, and generates an effect.

2.4 Analysis of Domain Functioning

The main objective of system thinking is to move implicitly or informally expressed knowledge to formal specifications of the system; in our case, in the form of TFM elements – functional features and cause-and-effect relations. As mentioned above, a general case is when a functional feature may have multiple causes and multiple effects, i.e., multiple control flows. The question is how to put the discovering of cause combinations, as well as the branching and joining of logical flows, on the formal base.

Possible formalization is <u>obligate determination and specification of all pre- and post-conditions of</u>

every TFM functional feature. Then it would be possible "to connect" a post-condition of one functional feature with an equal precondition of another functional feature. In such a way, a *sequence* of functional parts would be defined. However, the question about *logical (control) relations* between such sequences within a behavioral scenario and among behavioral scenarios cannot be solved without introducing logical operators into the textual or visual specifications of functionality.

Logical operators Lop are relations from classical logic such as conjunction (AND), disjunction (OR, XOR), and negation (¬). Conjunction indicates synchronous occurrence of referenced causes. Disjunction indicates asynchronous occurrence of referenced causes. Negation indicates that referenced causes did not occur.

Relations between causes and effects are causal implications. This means that a cause may or may not occur. There are four possible classical combinations (Cummins, 1995):

- *Modus Ponens*. IF cause THEN effect. The cause occurs. Thus, the effect follows.
- *Modus Tollens*. IF cause THEN effect. The effect does not occur. Thus, the cause did not precede.
- *Affirmation of the Consequent*. IF cause THEN effect. The effect occurs, thus the cause preceded.
- *Denial of the Antecedent*. IF cause THEN effect. The cause did not occur. Thus, the effect does not follow.

In order to define complete (compound) causes, a modeler needs to analyze all possible combinations of cause occurrences and to elect only those which are both necessary and sufficient.

The possible combinations of necessity and sufficiency may indicate the following outcomes:

- ***One cause:***
  - ***An*** incorrectly defined ***cause*** in a cause-and-effect relation between functional features: when a cause functional feature is not necessary and not sufficient for generation of an effect functional feature, then this cause-and-effect relation between features is defined incorrectly;
  - ***An incomplete cause*** is when it is necessary but not sufficient, or sufficient but not necessary. This may indicate that some needed causes were ignored.
- Existence of logical operators between ***two causes*** (Table 1):
  - An AND operator must be set between two causes if they *all are necessary, but not sufficient*;
  - An OR operator must be set between two causes if they *all are sufficient, but not necessary*.
- If ***each cause in a combination*** is *both necessary and sufficient* (Table 1), then these causes are joined by the logical operator XOR (exclusive OR).
- ***In the general case,*** when a cause is not necessary or sufficient, this indicates an incompleteness of causes. This means that we must first find missing causes (i.e., functionality presented by functional features), and then review all the combinations of occurrences and non-occurrence of causes and *elect those combinations where sufficiency is true, or both necessity and sufficiency are true.* If there are more than one combination, then they are joined by XOR, as mentioned in the

Table 1: Analysis of combinations of two causes (0-a cause does not occur, 1 – a cause occurs).

Cause1 – c_1	Cause2 – c_2	Logical combination	Necessary	Sufficient	Effect
Case "c_1 OR c_2 generates Effect"					
0	1	$\neg c_1$ AND c_2	false	true	1
1	0	c_1 AND $\neg c_2$	false	true	1
1	1	c_1 AND c_2	true	true	1
Case "c_1 AND c_2 generates Effect"					
0	1	$\neg c_1$ AND c_2	true	false	0
1	0	c_1 AND $\neg c_2$	true	false	0
1	1	c_1 AND c_2	true	true	1
Case "c_1 XOR c_2 generates Effect"					
0	1	$\neg c_1$ AND c_2	true	true	1
1	0	c_1 AND $\neg c_2$	true	true	1
1	1	c_1 AND c_2	false	false	0

previous point. We have to note that only one combination is excluded – when all causes do not occur, since it completely satisfies *Denial of the Antecedent*.

The result of these activities should be an accurate model of system's functioning with completely defined inputs, outputs, functioning cycles, and logical relations among control flows within the system.

Thus, in order to handle these "combinations of causes", which actually are "firing conditions" of effects, the tuple of a functional feature must be supplemented with an element that represents them.

3 APPLICATION OF THE ANALYSIS FOR PROBLEM SOLVING

Let us take a small problem domain for illustration of the suggested theory. An informal description of the problem "Management of the research group activities" is the following: *"The research group investigates issues in the field of interest. Once some valuable results are obtained, one or more members of the group prepare a paper as its authors. The completed paper is submitted to an appropriate conference by the responsible author. If the paper is accepted by the conference organizers, then the authors prepare a camera-ready paper in accordance with the obtained reviews. The responsible author submits the camera-ready paper to the conference, and presents it at the conference. If the paper is published, the responsible author records paper's bibliographical description in the authors' personal files. The presenter records his/her visit to the conference and the title of the paper in his/her personal file. Group members may attend conferences without accepted papers; these visits also are recorded in their personal files. Personal files of former group members are archived."*

The list of functional features for this problem domain obtained from the description (1-12), inferred during analysis of cause-and-effect relations in the first iteration (13-19) and the second iteration (20-22) is shown in Table 2.

After analysis of the suggested description, functional features from 1 to 12 together with corresponding cause-and-effect relations, which are illustrated in Figure 1(a), were defined. The obtained topological model has isolated vertices 10 and 12, and it does not have any functioning cycle. This model is not valid and must be refined. Figure 1(b) illustrates the refined model that completely satisfies topological and functional properties of the TFM. It was supplemented by functional features 13-19 and corresponding cause-and-effect relations, which specifies implicitly expressed knowledge about the domain (Figure 1(c), Table 2). The next step is to check completeness of causes for effect "firing" and valid combinations of causes as stated in Section 2.4. The necessity and sufficiency (T-true or F-false) of each relation are indicated above the arrows.

First, there is a list of insufficient cause-and-effect relations grouped by effects (14→1, 16→1, 18→1), (4→5, 2→5), (6→7, 13→7), (8→9, 17→9, 19→9), (7→11, 17→11), (13→10, 19→10, 17→10), 16→15, 15→18, (18→19, 12→19), (15→12, 17→12, 19→12). °

A combination of each pair of cause-and-effect relations (4→5, 2→5), (6→7, 13→7), (7→11, 17→11), and (18→19, 12→19) is sufficient when two causes occur, therefore these causes in each pair have a logical operator AND (Section 2.4).

Cause-and-effect relations 16→15 and 15→18 are not sufficient; this means that some causal relations were missed, or causes were not indicated in the model. In case of 16→15, starting a membership is not sufficient for ending a membership, there must be some reason. Existence of this reason is introduced by functional feature 22 and relation 22→15. The relation pair (16→15, 22→15) is necessary and sufficient if two causes are joined by a logical operator AND. In case of 15→18, ending a membership is not sufficient for renewing a membership, because a former member should ask to renew his/her membership. At the same time, starting a membership (feature 16) has a precondition that a candidate must not be a member of the group. Therefore, a new functional feature 21 "Appearance of a new member" is introduced and relations 21→18 (necessary, not sufficient) and 21→16 (necessary and sufficient) are set.

The logical operator AND is set for the pair (15→18, 21→18). The more complex cases are combinations of causes (14→1, 16→1, 18→1, 20→1), (8→9, 17→9, 19→9), and (13→10, 19→10, 17→10). For firing functional feature 1, the valid cause combination is ((20 OR 14) AND (16 XOR 18)). For firing functional feature 9, the valid cause combination is (8 AND (17 XOR 19)). For firing functional feature 10, the valid cause combination is (13 AND (17 XOR 19)). The resulting TFM is shown in Figure 1(c).

Table 2: The list of functional features, where the abbreviations are as follows: P - a person, RG – the research group, M – a member of the group, C – a conference, RA – a responsible author, CO – conference organizers, A – authors, Pr – a presenter, EE- the external environment.

Nr.	Name	PreCond	PostCond	Pr	Ex
1	Investigating an issue in the field of interest	issues	valuable results are obtained	RG	M
2	Preparing a new paper	valuable results are obtained	completed paper	RG	M
3	Submitting a new paper	completed paper		C	RA
4	Notifying the status of a paper		accepted paper OR not accepted paper	C	CO
5	Preparing a camera-ready paper	accepted paper	prepared camera-ready paper	RG	A
6	Submitting a camera-ready paper	prepared camera-ready paper	submitted camera-ready paper	C	RA
7	Presenting a camera-ready paper	submitted camera-ready paper and visited conference		C	Pr
8	Publishing a paper	submitted camera-ready paper	published paper	C	CO
9	Recording the bibliographical description of the paper in a personal file	published paper	all records are done	RG	RA
10	Recording the visit to the conference in a personal file	visited conference		RG	Pr, M
11	Recording the title of the paper in a personal file	presented paper		RG	Pr
12	Archiving a personal file	former group member		RG	RG
13	Visiting a conference		visited conference	C	M, Pr
14	Identifying the issues in a paper	not accepted paper	issues	RG	A
15	Ending a membership in the research group	current member	former member	RG	M
16	Starting a membership in the research group	not a member	new member	RG	P
17	Creating a personal file	new member	new personal file	RG	M
18	Renewing a membership in the research group	former member	renewed member	RG	M
19	Restoring a personal file	renewed member	restored personal file	RG	M
20	Existence of an issue in the field		issues	EE	EE
21	Appearance of a new member		a former member OR not a member	RG	P
22	Appearance of a membership finishing reason		reason to end the group	M	M

4 RELATED WORK

There are many works on causality and system thinking in the humanities, and only several of them are in the computer science. We would like to highlight only a few of them, which, by our opinion,

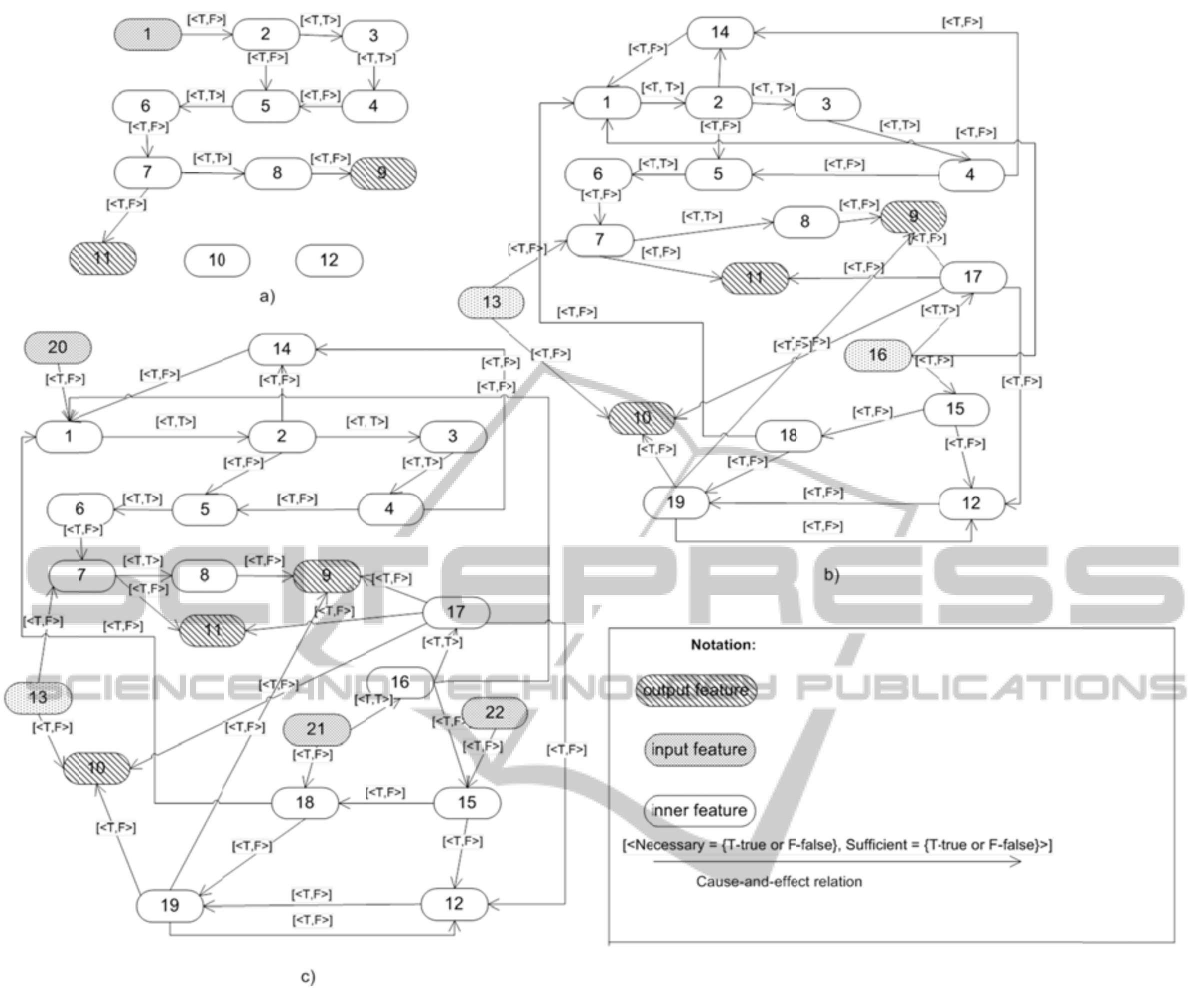
a)
b)
c)
Notation:
output feature
input feature
inner feature
[<Necessary = {T-true or F-false}, Sufficient = {T-true or F-false}>]
Cause-and-effect relation

Figure 1: A topological model of the system functioning: (a) obtained from the description, (b) inferred by using expert's knowledge about the system, (c) supplemented by using the cause-and-effect analysis.

are more interesting for understanding the place of our proposition and future research.

Authors in (Bider et al., 2011) discuss a model for managing enterprise agility, i.e., "property of an enterprise to function in the highly dynamic world". This model consists of three layers – enterprise assests, sensors, and business process instances. The authors joined two viewpoints on the system. The first viewpoint comes from system thinking, where the system is regarded as a whole that maintains its existence through constant interaction between its parts and a bigger whole, an external environment. The second viewpoint comes from a Business Process Management perspective, where a system is regarded as a number of repeatable business processes. A business process instance (BPI) is regarded as a system that is produced by a business process type (BPT) within an enterprise system. BPIs layer deals with this notions. Each BPT has a sensor, which discovers the need for a BPI. System thinking is applied for strategic BPT in order to find the best places to make changes in the whole organization. Both a sensor and a BPI are systems. Comparing this model with the TFM, we can conclude that the TFM has one layer which includes all BPIs and does not extract separate layers of assets and sensors. Sensors as systems are distributed over input and output functional features and cause-and-effect relations. Usually, functional characteristics of management are omitted in the TFM, because this model has another field of application. Discussion on causality of relationships among system objects leads authors to define declarative rules (Khomyakov and Bider, 2000). These rules constrain chaos within the system, and breaking these rules forces the system to restore them, i.e., forces the system to function. Definition of these rules is based on expert knowledge, and the rules may be composed. It is similar to our proposition, where logical relations among causes and causal relations among causes and effects can be considered as such rules.

Theoretical foundations of causality of relationships are well described by Chris Taylor (Taylor, 1993). Speaking about causation of temporal events (that is close to our discussion), the author defined several sets – a set of world (system) elements, a set of world states, a set of events (which are regarded as transition from one world state to another), and a set of worlds, which contains all possible (lawful) worlds for each state in the set of states. In other words, the author defines all possible transitions from a state to another related state. And these transitions also have logical relations – conjunction, disjunction, and negation. In this case, the author considers a counterfactual analysis. In our proposition we use a law-based analysis, i.e., we do not consider "possible worlds" for the event.

5 CONCLUSIONS

Application of the TFM together with careful analysis of causal relations among functional characteristics of the system allows investigating the system and its surrounding environment. The result is explicitly specified knowledge about stimulus (inputs) and reactions (outputs) of the system, its functioning cycles, and more complete understanding of collaboration among system's functional characteristics, namely, well-specified information about conductors, resources, control flows, activities, objects, and results.

In case of a very large system and a complex domain, the TFM provides a mathematical means for abstraction – continuous mapping between graphs. Functional features in a refined model may be mapped to one functional feature in a more abstract (simpler) model, while keeping all cause-and-effect relations with other functional features, which were defined in the refined model. Thus, at higher levels of abstraction cause-and-effects relations among large system fragments (or functional components) will be analyzed. But at lower levels of abstraction, analysis of cause-and-effect relations within those fragments will be conducted. Certainly, this work must be iterative, because changes in the model at any level of abstraction may have impact on the model at other, lower and higher, levels of abstraction.

As a computation independent model, the TFM can be used as an input specification for automated transformations to the more detailed computation independent and initial platform-independent models– traceability models, business process models, use case models, class diagrams, and object interaction diagrams. Work on formalization of mappings from TFM to these models has been referred in Introduction. Additionally, the TFM as an input specification must be properly verified before transformation to other models. Future research direction is TFM verification by model checking approaches, e.g., Colored Petri Nets.

REFERENCES

Asnina, E., & Osis, J. (2010). Computation independent models: bridging problem and solution domains. *Proceedings of the 2nd InternationalWorkshop on Model-Driven Architecture and Modeling Theory-Driven Development MDA & MTDD 2010, In conjunction with ENASE 2010, Athens, Greece, July 2010* (pp. 23-32). Portugal: SciTePress.

Asnina, E., & Osis, J. (2011). Topological Functioning Model as a CIM-Business Model. In J. Osis, & E. Asnina, *Model-Driven Domain Analysis and Software Development: Architectures and Functions* (pp. 40-64). Hershey, New York, USA: IGI Global.

Basener, W. (2006). *Topology and Its Applications.* New Jersey, USA: John Wiley and Sons, Inc.

Bider, I., Bellinger, G., & Perjons, E. (2011). Modeling an Agile Enterprise: Reconciling Systems and Process Thinking. *Proceedings of PoEM 2011, LNBIP , 92*, pp. 238-252.

Cummins, D. (1995). Naive theories and causal deduction. *Memory and Cognition , 23*, pp. 646-658.

Donins, U., Osis, J., Slihte, A., Asnina, E., & Gulbis, B. (2011). Towards the Refinement of Topological Class Diagram as a Platform Independent Model. *Model-Driven Architecture and Modeling-Driven Software Development: ENASE 2011, 3rd Whs. MDA&MDSD*, (pp. 79 - 88).

Drack, M., & Apfalter, W. (2007). Is Paul A. Weiss' and Ludwig von Bertalanffy's System Thinking Still Valid Today? *Systems Research and Behavioral Science , 24* (5), pp. 537-546.

Khomyakov, M., & Bider, I. (2000). Achieving Workflow Flexibility through Taming the Chaos. *OOIS 2000 - 6th international conference on object oriented information systems* (pp. 85-92). Springer.

Laszlo, A., & Krippner, S. (1998). Chapter 3 Systems theories: Their origins, foundations, and development. In J. Scott Jordan, *Advances in Psychology* (Vol. 126, pp. 47-74). North-Holland.

Lavagno, L., Grant, E. M., & Selic, B. (2004). *UML for Real: Design of Embedded Real-Time Systems.* Springer.

Lee, B., & Miller, J. (2004). Multi-project Software Engineering Analysis Using Systems Thinking.

Software Process Improvement and Practice (9), pp. 173–214.

Miller, J., & Mukerji, J. (Eds.). (2003, May 1). *MDA Guide Version 1.0.* Retrieved January 15, 2010, from http://www.omg.org/mda/

Mingers, J., & White, L. (2010). A review of the recent contribution of systems thinking to operational research and management science. *European Journal of Operational Research , 207*, 1147–1161.

Osis, J. (2006). Formal Computation Independent Model within the MDA Life Cycle. (P. Loucopoulos, & K. Lyytinen, Eds.) *International transactions on system science and applications , 1* (2), pp. 159-166.

Osis, J. (2004). Software development with topological model in the framework of MDA. *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. 1*, pp. 211-220. Riga, Latvia: RTU.

Osis, J. (1969). Topological Model of System Functioning. *Automatics and Computer Science, J. of Acad. of Sc.* (6), 44-50.

Osis, J., & Asnina, E. (2008). A Business Model to Make Software Development Less Intuitive. *Proceedings of 2008 International Conference on Innovation in Sofware Engineering (ISE 2008). December 10-12, 2008, Vienna, Austria* (pp. 1240-1245). IEEE Computer Society Publishing.

Osis, J., & Asnina, E. (2011 a). Derivation of Use Cases from the Topological Computation Independent Business Model. In J. Osis, & E. Asnina, *Model-Driven Domain Analysis and Software Development: Architectures and Functions* (pp. 65-89). Hershey, New York, USA: IGI Global.

Osis, J., & Asnina, E. (2011 b). Is Modeling a Treatment for the Weakness of Software Engineering? In J. Osis, & E. Asnina, *Model-Driven Domain Analysis and Software Development: Architectures and Functions* (pp. 1-14). Hershey - New York, USA: IGI Global.

Osis, J., & Asnina, E. (2011 c). *Model-Driven Domain Analysis and Software Development: Architectures and Functions.* Hershey, New York, USA: IGI Global.

Osis, J., & Asnina, E. (2011 d). Topological Modeling for Model-Driven Domain Analysis and Software Development. In J. Osis, & E. Asnina, *Model-Driven Domain Analysis and Software Development: Architectures and Functions* (pp. 15-39). Hershey, New York, USA: IGI Global.

Osis, J., & Donins, U. (2010). Formalization of the UML Class Diagrams. In *Evaluation of Novel Approaches to Software Engineering* (pp. 180-192). Berlin: Springer-Verlag.

Osis, J., Asnina, E., & Grave, A. (2007 a). Computation Independent Modeling within the MDA. *Proceedings of IEEE International Conference on Software, Science, Technology & Engineering (SwSTE07), 30-31 October 2007, Herzlia, Israel* (pp. 22-34). IEEE Computer Society, Conference Publishing Services (CPS).

Osis, J., Asnina, E., & Grave, A. (2008 a). Computation Independent Representation of the Problem Domain in MDA. *e-Informatica Software Engineering Journal , 2* (1), 29-46.

Osis, J., Asnina, E., & Grave, A. (2007 b). Formal Computation Independent Model of the Problem Domain within the MDA. *Information Systems and Formal Models, Proceedings of the 10th International Conference ISIM'07* (pp. 47-54). Opava, Czech Republic: Silesian University.

Osis, J., Asnina, E., & Grave, A. (2008 b). Formal Problem Domain Modeling within MDA. In *Communications in Computer and Information Science (CCIS). Software and Data Technologies* (pp. 387-398). Berlin: Springer-Verlag.

Slihte, A., Osis, J., & Donins, U. (2011). Knowledge Integration for Domain Modeling. *Model-Driven Architecture and Modeling-Driven Software Development: ENASE 2011, 3rd Whs. MDA&MDSD*, (pp. 46 - 56).

Taylor, C. (1993). *A Formal Logical Analysis of Causal Relations, DPhil Thesis.* University of Sussex.