Verification of Causality in the Frame System based on the Topological Functioning Modelling

Vladislavs Nazaruks and Jānis Osis
Department of Applied Computer Science, Riga Technical University, Sētas iela 1, LV-1048, Riga, Latvia

Keywords: Verification, Knowledge Frames, Knowledge Base, Topological Functioning Model, Causal Dependencies.

Abstract: Causality is universal relations among phenomena (states, facts, elements, functions) in the system. Verification of causality in the knowledge frame system based on principles of the topological functioning modelling can help in discovering inconsistencies such as incompleteness, ambiguity or contradictions in knowledge on system's functioning. The method for such verification is presented in this paper. It is based on topological and functioning properties of the topological functioning model including the definition of continuous mapping between topological spaces. The method helps in discovering inconsistent combinations of cause-and-effect relations or a lack of them. Functional characteristics of the system involved in these relations are marked as doubtful. The results of verification require additional investigation by a software developer. A use of the proposed method can lead to more thorough system analysis before development of the solution.

1 INTRODUCTION

Causality is an important concept in conditional logic, in artificial intelligence, e. g. for planning tasks (Wobcke, 1994; Giunchiglia *et al.*, 2004), and in the framework of action systems (Giordano and Schwind, 2004). Causality is universal, there is no any system that has no causal dependencies among its constituents (Osis and Asnina, 2011; Nazaruka, 2017). Causality is represented as a set of causal dependencies or causal implications.

The causal implication can exist between two types of assertions (Giordano and Schwind, 2004): (1) an action can cause a fact to become true, or (2) a fact can cause another fact. In the first case, the causal implication relates also to a state transition, so the caused fact "belongs" to the "next state". In the second case, the causal implication does not touch any state, the modifications occur in the same state.

Verification of causal dependencies can help in discovering functional, behavioural and structural inconsistencies in models of domains of different types, e. g. business, mechanical, biological etc. Models can be informal, semi-formal and formal. But all of them can contain inconsistencies with the modelled domain. Formal models allow discovering these inconsistencies. For domain modelling we suggest using advantages provided by the Topological Functioning Model – formal, but "light-weight" model that can be transformed to most-used UML (Unified Modelling Language) diagrams (Donins *et al.*, 2011, 2012; Donins, 2012b).

Causal dependencies in the TFM are called cause-and-effect relations or topological relations, since they represent topology on a set of system's functional characteristics (Osis and Asnina, 2011). Cause-and-effect relations transformed into elements of the design model are control flows, data flows and transitions among states (Osis and Asnina, 2008; Donins *et al.*, 2012; Asnina and Ovchinnikova, 2015).

Cause-and-effect relations and characteristics of the validity of the TFM – the central model in the topological functioning modelling – can be used for verification of BPMN diagrams (Nazaruka *et al.*, 2016). The approach for validation of causality in knowledge specified in the TFM using execution model simulation has been discussed in (Ovchinnikova and Nazaruka, 2016).

The construction of the TFM is based on procedural and declarative knowledge. The more suitable way for storing knowledge is a knowledge base. The base that incorporate TFM principles (Nazaruks and Osis, 2017) contains generable and manually added knowledge. Verification of knowledge includes generation and validation of the topological space and the corresponding TFM that

Nazaruks, V. and Osis, J.
Verification of Causality in the Frame System based on the Topological Functioning Modelling.
DOI: 10.5220/0006817905130521
In *Proceedings of the 13th International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE 2018)*, pages 513-521
ISBN: 978-989-758-300-1

allow discovering incompleteness and contradictions in the added knowledge.

The goal of the current research is to develop method for verification of the topology in the knowledge kept in the frame system (Section 3.2) using validity characteristics of the TFM.

The paper organization is the following. Section 2 gives summary of related work. Section 3 presents the necessary background of the TFM and the frame system based on its constructs and characteristics. Section 4 gives a description of the proposed method, which is illustrated in Section 5. Section 6 summarizes the main results and concludes the paper.

2 RELATED WORK

The next step after creation of the knowledge base is its verification and validation (Santos and Dinh, 2008). Verification is dedicated to find anomality in gathered knowledge. Rule-based knowledge bases can contain such anomality (or inconsistencies) as redundancy, subsuming rules, contradictory rules, inconsistent rules, incompleteness, missing rules, recursive rules, and unused attributes and values (Sarkar and Ramaswamy, 2000; Mach-Król and Michalik, 2015).

Fuzzy inference rules that are developed by a group of domain experts can contain all the mentioned inconsistencies (Skorupski, 2015). The author suggests an approach for their verification by using the expert-group evaluation and then assessing of the obtained results. The idea is that semantical analysis should be done by the experts themselves, the system only evaluates their decision and generates more certain rules automatically. Then the generated rules are compared with the rules provided by the experts. The idea that is proposed in our research is similar in some degree, i. e. when we compare the generated knowledge with the knowledge obtained from domain experts or documents.

Another way is a usage of Petri Nets, for example, for verification of frame-rule hybrid expert systems (Tadj and Laroussi, 2006). Here, Petri Nets are used as a reference model created for analysis of the markings graphs. The list of inconsistencies can be extended with hypotactic rules, conflict rule chains, dead ends and unreachable goals (Wu *et al.*, 2005). For discovering them, Coloured Petri Nets can be applied (Wu *et al.*, 2005). Verification can be based also on principles of data integrity, e. g. the consistency check can be done using SQL query method (Liu and Jiang, 2010).

Another knowledge verification method is based on using ontologies. For example, ontologies can be used to verify computer-based knowledge sharing between departments of a manufacturer enterprise (Anjum *et al.*, 2013) or for verification of planned and real plans (Zhong *et al.*, 2015).

Anjum *et al.* (2013) note that the necessary functionality of the enterprise system is to deal with several types of incompatibilities and heterogeneities between sets of knowledge located in independently developed computer-based knowledge management systems. Authors' idea is to match two domain ontologies and in such a way to verify the knowledge. As the authors mention, there are foundational and domain ontologies. Foundational ontologies such as WordNet are used as a common vocabulary for knowledge bases, while domain ontology have heterogeneous nature and must be semantically and syntactically verified against one common vocabulary. The proposed approach deals with parent-child relationships in domain ontologies, but it requires further elaboration for more complex ontological structures. However, the presented idea of semantical and syntactical verification as well as searching for similarities and differences is close to the idea presented in this paper.

Description logic can be used for keeping knowledge of model variants and their dependencies as well as for verifying inconsistencies in those knowledge (Asadi *et al.*, 2016).

For verification of very large complex knowledge bases, a decomposition to the smaller information-related based can be used (Sarkar and Ramaswamy, 2000). In this approach, first, the decomposed parts are verified using the "directed hypergraph approach", and then dependencies among those parts are analysed based on "ordered polytree structures".

Summarizing, verification of knowledge is a comparison of actual and gathered, prescribed, or planned knowledge. Inconsistencies can be found in the gathered knowledge and then they are verified by known-to-be-sound ontologies, sets of rules or experts opinions. Inconsistencies can be found in actual knowledges and then they can be either compared with some common sound base or modelled and verified by model checking techniques.

In our method, we apply verification by models, i. e. verifying topological and functioning properties of the TFM.

3 BACKGROUND

3.1 The Topological Functioning Model in Brief

The TFM is a formal model which describes the functionality of a system. Its fundamentals are published in (Osis, 1969; Osis and Asnina, 2011). The TFM can be presented in a form of a topological space (X, Θ), where X is a finite set of functional features (characteristics) of the system, and Θ is a topology set on X.

A functional feature is "a characteristic of the system (in its general sense) that is designed [for] and necessary to achieve some system's goal" (Osis and Asnina, 2011). It can be specified by a unique tuple (1), where:

$$\langle A, \boldsymbol{R}, \boldsymbol{O}, \boldsymbol{PrCond}, \boldsymbol{PostCond}, \boldsymbol{Pr}, \boldsymbol{Ex}, S\rangle \qquad (1)$$

- A is an action linked with object $\boldsymbol{O}$,
- $\boldsymbol{R}$ is a result of the action A,
- $\boldsymbol{O}$ is an object (objects) that gets the result of the action or an object (objects) that is used in this action,
- $\boldsymbol{PrCond}$ is a set of preconditions or atomic business rules,
- $\boldsymbol{PostCond}$ is a set of postconditions or atomic business rules,
- $\boldsymbol{Pr}$ is a set of responsible entities (systems or subsystems) that provide or suggest action A with a set of certain objects $\boldsymbol{O}$,
- $\boldsymbol{Ex}$ is a set of responsible entities (systems or subsystems) that enact a concrete action A (Osis and Asnina, 2011; Nazaruka *et al.*, 2016).
- S is a label that indicates belonging of the functional feature to the system for which the TFM will be composed, namely, *inner* if belongs and *external* if does not.

The topology Θ is presented by cause-and-effect relations. A cause-and-effect relation is a causal implication (causal dependency) between two functional features. It is a binary relationship, where a cause triggers an effect without any intermediate functional feature (Osis, Asnina and Grave, 2008; Asnina and Osis, 2011). There are several specifications of cause-and-effect relations (Osis and Slihte, 2010; Donins, 2012a; Asnina and Ovchinnikova, 2015), but the common is that they are focused on assessment of the completeness of incoming and outgoing conditions, as well as on logical correctness.

A cause-and-effect relation is a topological relation T_{id} (2) between a cause functional feature X_c and an effect functional feature X_e, where at least one condition of L_{out} that is a set of X_c postconditions is equal to the at least one condition of L_{in} that is X_e preconditions (Donins, 2012a).

$$T_{id} = \langle id, X_c, X_e, L_{out}, L_{in}\rangle \qquad (2)$$

The TFM is valid when it satisfies topological and functioning properties (Osis and Asnina, 2011). The topological properties are: connectedness, neighbourhood, closure and continuous mapping. The functioning properties are: cause-and-effect relations, cycle structure, inputs and outputs.

Let us consider the basics of closure and continuous mapping (Osis and Asnina, 2011), since they are needed for understanding of the presented verification.

The *closure* is an operation of separation of the TFM of the *system* from its *topological space*. The topological space contains functional characteristics (set $\boldsymbol{Z} = \boldsymbol{N} \cup \boldsymbol{M}$, where $\boldsymbol{N}$ is a set of inner functional features of the system, and $\boldsymbol{M}$ is a set of external functional features to the system) and causal dependencies of the system itself and other systems that interact with it. The aim of the closure is to get boundaries of the system, or the set $\boldsymbol{X}$ that contains the union of all neighbourhoods of functional features in the set $\boldsymbol{N}$. As a result, those functional features that have no direct interaction with the system's functional features are cut off as well as cause-and-effect relations that have linked them with functional features that have been included into the TFM. Sometimes, this can lead to discovering inconsistencies in causal dependencies, e.g., determination of unwanted independent sub-systems, isolated vertices, and broken functional cycles.

The *continuous mapping* is a relation between topological spaces that states that "direction of topological model arcs must be kept as in a refined as in a simplified model" (Osis and Asnina, 2011, p. 29). The continuous mapping allows comparing topological spaces for resemblance and differences.

3.2 The Scheme of the Frame System

The frame system suggested here is based on (but not limited to) the elements necessary to generate the TFM without definition of logical operations among cause-and-effect relations. The system represents domain ontology but does not specify scripts or daemons for frame instance generation.

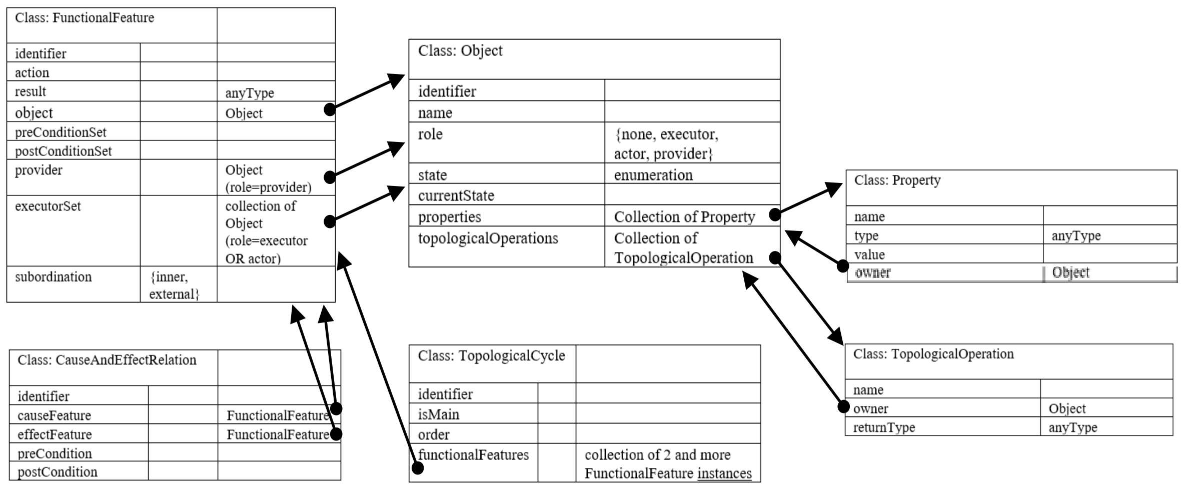

Figure 1: The schema of the frame system (Nazaruks, 2017).

The following frame classes are presented (Figure 1):

- Classes with manually added knowledge: FunctionalFeature, and Property;
- Classes with partial generation of knowledge: Object, and TopologicalCycle;
- Classes with complete generation of knowledge: CauseAndEffectRelation, and TopologicalOperation.

The frame instances purpose is the following (Nazaruks and Osis, 2017):

- CauseAndEffectRelation — for knowledge on cause-and-effect relations generated from instances of FunctionalFeature considering that the cause is predefined by using a precondition, while the effect is specified by using a postcondition;
- FunctionalFeature — for facts about the functional features;
- Object — for objects that participate in the execution of the functional feature;
- Property — for the domain object;
- TopologicalOperation — for knowledge about the operations that will implement actions of the functional features and are *generated* from FunctionalFeature data;
- TopologicalCycle — for facts about participation of functional features in cycles of functionality.

At the present, frame instances are filled in with data manually. These frames are core elements for further transformation (or it is better to say, generation) of analysis or design models depending from the knowledge on the software domain.

4 METHOD OF VERIFICATION

The idea of retrieving the topology by analysis of pre- and post- conditions has been proposed in Topological UML. There it has been applied for identification of logical relations among cause-and-effect relations (Donins, 2012a). This approach also requires human participation, since postcondition and precondition sets may be not indicated, thus semantics of logical conditions must be analysed properly.

The proposed method consists of the following steps:

- Step 1: verification of generated cause-and-effect relations corresponding to the specified pre- and postconditions in the specifications of functional features in the topological space.
- Step 2: separation of the TFM from the topological space.
- Step 3: verification of topological and functioning properties of the TFM.
- Step 4: verification of inconsistencies between the TFM and its original topological space.

Step 1.

Suppose we have a set of functional features $\boldsymbol{Z} = \{Z_i\}$ and a set of cause-and-effect (or topological) relations $\boldsymbol{T} = \{T_i\}$ among them, that were automatically generated by analysing the pre- and postconditions of the functional features. Let us define a function $C(\boldsymbol{L})$ which returns a set of common atomic parts of all the expressions in the set $\boldsymbol{L}$ (here we suppose that A and $\neg A$ are different atomic values). For example:

- if $\boldsymbol{L} = \{A \vee B \vee C; A \wedge D; A \vee D\}$ then $C(\boldsymbol{L}) = \{A\}$,
- if $\boldsymbol{L} = \{A \vee (B \wedge C); B \wedge C \wedge D\}$ then $C(\boldsymbol{L}) = \{B; C\}$,
- if $\boldsymbol{L} = \{A; \neg A \vee B\}$ then $C(\boldsymbol{L}) = \emptyset$.

First, we expand each cause-and-effect relation tuple T_i with an element C, where $C = C(\{L_{in}; L_{out}\})$ (3):

$$T'_{id} = \langle id, Z_c, Z_e, L_{out}, L_{in}, C\rangle \tag{3}$$

Second, for each functional feature Z_i we define used preconditions (4) and postconditions (5).

$$\text{precond}_{\text{used}}(Z_i) = \bigcup_{T'_i \in \boldsymbol{T}: Z_i = Z_e} C_{T'_i} \tag{4}$$

$$\text{postcond}_{\text{used}}(Z_i) = \bigcup_{T'_i \in \boldsymbol{T}: Z_i = Z_c} C_{T'_i} \tag{5}$$

Third, for each functional feature Z_i we calculate the differences D_{precond} between the sets of its preconditions and used preconditions (6), and D_{postcond} between the sets of its postconditions and used postconditions (7).

$$D_{\text{precond}}(Z_i) = \textbf{PrCond}_{Z_i} \setminus \text{precond}_{\text{used}}(Z_i) \tag{6}$$

$$D_{\text{postcond}}(Z_i) = \textbf{PostCond}_{Z_i} \setminus \setminus \text{postcond}_{used}(Z_i) \tag{7}$$

If for a specific functional feature Z_i the difference $D_{\text{precond}}(Z_i)$ or $D_{\text{postcond}}(Z_i)$ is not an empty set, then the corresponding functional feature is to be marked as possibly inconsistent.

Step 2.
The topological space is verified whether it contains inconsistencies such as isolated vertices, a lack of inputs, a lack of outputs, a lack of functioning cycles.

Step 3.
If the topological space is valid, then the closuring operation is executed over the set N of inner functional features of the system.

Otherwise, the list of inconsistencies that must be improved before the separation of the TFM is presented to the modeler or developer.

After improvement, the verification starts from Step 1.

Step 4.
The TFM is verified whether it contains inconsistencies such as isolated vertices, a lack of inputs, a lack of outputs, and a lack of functioning cycles. If the TFM is valid, the verification is successfully finished.

In case of incomplete knowledge of the domain functionality, more than one graph can be obtained after closuring. The graph can represent either a subsystem (if it has at least one functioning cycle), or a part of systems functionality (some process or a set of processes). In this case it is very important to indicate functional features that were cut off because of closuring. Comparing two topological spaces, the TFM and its original topological space, cut-off paths (chains of cause-and-effect relations) can be found and presented for the developer for a review.

Summarizing, the proposed method allows verifying incompleteness of conditions, functional characteristics of the system, and causal dependencies. At the present, the method does not support semantical verification of conditions and their combinations.

5 ILLUSTRATIVE EXAMPLE

Description of the business domain is as follows. “A criminal case is initiated by an investigator when a criminal act is stated. The criminal act may be stated when a criminal person has committed a criminal act and it was discovered or a victim or witness has submitted a claim about it. After the criminal case was initiated, the investigator conducts investigative actions. As the result of this, the indicted person is found. After the investigation is completed, the criminal case is sent to a prosecutor. If the criminal act is misdemeanour, the prosecutor can draw up a penal order. If the indicted person agrees with the accusation presented and the penalty the prosecutor offered, then the criminal case is terminated, and the convicted person serves the punishment. Otherwise, the prosecutor sent the case to the court. The criminal case is terminated, when the court adjudicates in the case.”

Having this knowledge about the system, a modeler can fill in frame instances for functional features and objects. The initial descriptions of functional features that contain the identifier, action, result and object, as well as providers and executors are the following:

- 1. Initiating [new] CriminalCase, {State Police}, {Investigator};

- 2. Commiting [new] CriminalAct, {}, {CriminalPerson}
- 3. Discovering [new] CriminalAct, {State Police}, {}
- 4. Submitting [new] ClaimOnCriminalAct, {State Police}, {victim, witness}
- 5. Stating [new] CriminalAct, {}, {}
- 6. Conducting investigativeActions CriminalCase, {State Police}, {}
- 7. Sending toProsecutor CriminalCase, {State Police}, {Investigator}
- 8. Drawing up penalOrder CriminalCase, {Prosecution Office}, {Prosecutor}
- 9. Terminating [] CriminalCase, {Prosecution Office}, {Prosecutor}
- 10. Serving [] Punishment, {Prisons Administration}, {ConvictedPerson}
- 11. Sending [to the] Court CriminalCase, {Prosecution Office}, {Prosecutor}
- 12. Adjudicating [] CriminalCase, {Court}, {}

The corresponding pre- and postconditions are illustrated in Table 1. Then, the verification can be started.

Step 1. Applying Step 1 of the verification method described in Section 4 gives the following results: the corresponding generated cause-and-effect relations (Table 2), a topological space of the functional features (Figure 2) and comparison of generated and specified pre- and postconditions (Table 3) that clearly show inconsistencies in knowledge of functional features 6, 7, 8, 9, and 11.

Table 1: Values of slots preConditionSet and postConditionSet of frame instances of FunctionalFeatures.

identifier	**preConditionSet**	**postConditionSet**
1	a criminal act is stated	a criminal case is initiated
2	—	a criminal act is committed
3	a criminal act is committed	a criminal act is discovered
4	a criminal act is committed	a claim about a criminal act is submitted
5	(a criminal act is committed) AND ((a criminal act is discovered) OR (a claim about a criminal act is submitted))	a criminal act is stated
6	a criminal case is initiated	an indicted person is found
7	an investigation is completed	a criminal case is sent to prosecutor
8	a criminal act is misdemeanour	a penal order is drawn
9	an indicted person agrees with the accusation presented and the penalty the prosecutor offered	a criminal case is terminated
10	a criminal case is terminated	—
11	NOT(an indicted person agrees with the accusation presented and the penalty the prosecutor offered)	(NOT (a criminal case is terminated)) AND (a criminal case is sent to the court)
12	(NOT (a criminal case is terminated)) AND (a criminal case is sent to the court)	a criminal case is terminated

Table 2: The generated frame instances of CauseAndEffectRelation.

id	Z_c	Z_e	L_{out}	L_{in}	C_{Z_i}
1-6	1	6	a criminal case is initiated	a criminal case is initiated	a criminal case is initiated
2-3	2	3	a criminal act is committed	a criminal act is committed	a criminal act is committed
2-4	2	4	a criminal act is committed	a criminal act is committed	a criminal act is committed
2-5	2	5	a criminal act is committed	And(a criminal act is committed, Or(a criminal act is discovered, a claim about a criminal act is submitted))	a criminal act is committed
3-5	3	5	a criminal act is discovered	And(a criminal act is committed, Or(a criminal act is discovered, a claim about a criminal act is submitted))	a criminal act is discovered
4-5	4	5	a claim about a criminal act is submitted	And(a criminal act is committed, Or(a criminal act is discovered, a claim about a criminal act is submitted))	a claim about a criminal act is submitted
5-1	5	1	a criminal act is stated	a criminal act is stated	a criminal act is stated
9-10	9	10	a criminal case is terminated	a criminal case is terminated	a criminal case is terminated
11-12	11	12	And(Not(a criminal case is terminated), a criminal case is sent to the court)	And(Not(a criminal case is terminated), a criminal case is sent to the court)	Not(a criminal case is terminated), a criminal case is sent to the court
12-10	12	10	a criminal case is terminated	a criminal case is terminated	a criminal case is terminated

Table 3: Comparison of pre- and postconditions of functional features.

id	$\text{precond}_{\text{used}}(Z_i)$	$\text{postcond}_{\text{used}}(Z_i)$	$D_{\text{precond}}(Z_i)$	$D_{\text{postcond}}(Z_i)$
1	{a criminal act is stated}	{a criminal case is initiated}	∅	∅
2	∅	{a criminal act is committed}	∅	∅
3	{a criminal act is committed}	{a criminal act is discovered}	∅	∅
4	{a criminal act is committed}	{a claim about a criminal act is submitted}	∅	∅
5	{a criminal act is committed, a criminal act is discovered, a claim about a criminal act is submitted}	{a criminal act is stated}	∅	∅
6	{a criminal case is initiated}	∅	∅	{an indicted person is found}
7	∅	∅	{an investigation is completed}	a criminal case is sent to prosecutor
8	∅	∅	{a criminal act is misdemeanour}	a penal order is drawn
9	∅	{a criminal case is terminated}	{an indicted person agrees with the accusation presented and the penalty the prosecutor offered}	∅
10	{a criminal case is terminated}	∅	∅	∅
11	∅	{Not(a criminal case is terminated), a criminal case is sent to the court}	{Not(an indicted person agrees with the accusation presented and the penalty the prosecutor offered)}	∅
12	{Not(a criminal case is terminated), a criminal case is sent to the court}	{a criminal case is terminated}	∅	∅

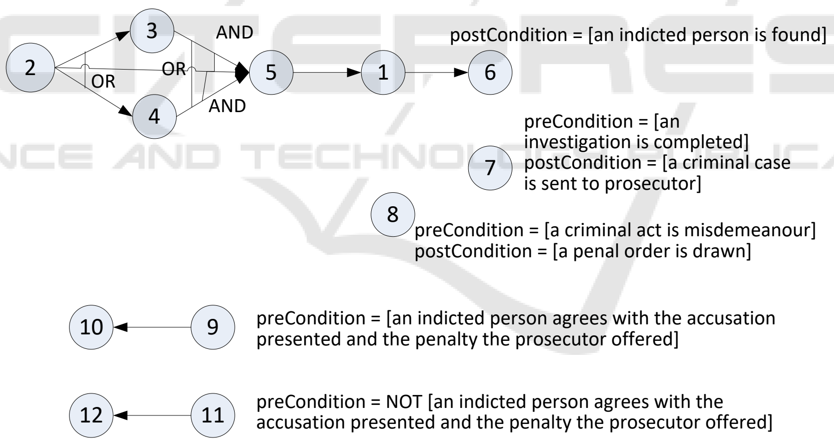
2
3
4
5
1
6
OR
OR
AND
AND
postCondition = [an indicted person is found]
7
preCondition = [an investigation is completed]
postCondition = [a criminal case is sent to prosecutor]
8
preCondition = [a criminal act is misdemeanour]
postCondition = [a penal order is drawn]
10
9
preCondition = [an indicted person agrees with the accusation presented and the penalty the prosecutor offered]
12
11
preCondition = NOT [an indicted person agrees with the accusation presented and the penalty the prosecutor offered]

Figure 2: The generated topological space of the system functionality.

Step 2. The verification of the topological space showed that it contains: a) isolated functional features 7 and 8, b) two isolated groups of functional features; c) functional features 6, 9, and 11 with pre- and postconditions that have no connections. So it is not clear what happens if an indicted person is found (functional feature 6), when an investigation is considered as complete and what happens when a criminal case is sent to a prosecutor (functional feature 7), how, when and who determines that a criminal case is misdemeanour and what actions should be done when a penal order is drawn (functional feature 8), as well as how "an indicted person agrees with the accusation presented and the penalty the prosecutor offered", who and when presents the accusation and when the penalty is offered by the prosecutor (functional features 9 and 11).

The result of the execution of this step is the end of the algorithm with presenting discovered inconsistencies to the modeler or developer.

6 CONCLUSIONS

Construction of knowledge bases either using an assistance of domain experts, or using automated solutions foresees verification of the obtained knowledge.

Inconsistencies can be in both declarative and procedural knowledge. They are incompleteness, redundancy, contradictions, recursion, etc. Verification must use valid knowledge such as common recognized ontologies, or transform the existing knowledge to formal models for further analysis. The proposed method allows verifying incompleteness among conditions and functional characteristics of the system based on the analysis of causal dependencies, first, in a topological space, second, in the formally separated topological functioning model, and, third, between topological spaces of the model and its original topological space. The results of the verification show all possible inconsistencies and require semantical analysis by a domain expert.

At the present, the method does not support any automatic semantical verification of conditions and their combinations.

If pre- and postconditions are kept in some rules representation format, then it would be possible to use existing verification techniques for such analysis. Analysis of inconsistencies in cause-and-effect relations leads to discovering incompleteness in system's functional characteristics. This leads to corrections not only in functional but also in structural elements of the system.

The aim of the future research is to implement semantical analysis of the inconsistencies in knowledge in the frame system.

REFERENCES

Anjum, N., Harding, J., Young, R., Case, K., Usman, Z. and Changoora, T., 2013. Verification of knowledge shared across design and manufacture using a foundation ontology, *International Journal of Production Research*, 51(22), pp. 6534–6552. doi: 10.1080/00207543.2013.798051.

Asadi, M., Gröner, G., Mohabbati, B. and Gašević, D., 2016. Goal-oriented modeling and verification of feature-oriented product lines, *Software & Systems Modeling*, 15(1), pp. 257–279. doi: 10.1007/s10270-014-0402-8.

Asnina, E. and Osis, J., 2011. Topological Functioning Model as a CIM-Business Model, in *Model-Driven Domain Analysis and Software Development*. Hershey, PA: IGI Global, pp. 40–64. doi: 10.4018/978-1-61692-874-2.ch003.

Asnina, E. and Ovchinnikova, V., 2015. Specification of Decision-making and Control Flow Branching in Topological Functioning Models of Systems, in *International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE), 2015*. Barcelona, Spain: SciTePress, pp. 364–373. doi: 10.5220/0005479903640373.

Donins, U., 2012a. Semantics of Logical Relations in Topological Functioning Model, in *Proceedings of the 7th International Conference on Evaluation of Novel Approaches to Software Engineering, Wrocław, Poland, 29-30 June, 2012*. SciTePress, pp. 217–223.

Donins, U., 2012b. *Topological Unified Modeling Language: Development and Application. PhD Thesis.* Riga Technical University.

Donins, U., Osis, J., Asnina, E. and Jansone, A., 2012. Formal analysis of objects state changes and transitions, in *ENASE 2012 - Proceedings of the 7th International Conference on Evaluation of Novel Approaches to Software Engineering*. Lisbon: SciTePress, pp. 249–256.

Donins, U., Osis, J., Slihte, A., Asnina, E. and Gulbis, B., 2011. Towards the refinement of topological class diagram as a platform independent model, in Čaplinskas, A., Dzemyda, G., Lupeikienė, A., and Vasilecas, O. (eds) *Proceedings of the 3rd International Workshop on Model-Driven Architecture and Modeling-Driven Software Development, MDA and MDSD 2011, in Conjunction with ENASE 2011*. Vilnius: Žara, pp. 79–88.

Giordano, L. and Schwind, C., 2004. Conditional logic of actions and causation, *Artificial Intelligence*. Elsevier, 157(1–2), pp. 239–279. doi: 10.1016/j.artint.2004.04.009.

Giunchiglia, E., Lee, J., Lifschitz, V., McCain, N. and Turner, H., 2004. Nonmonotonic causal theories, *Artificial Intelligence*. Elsevier, 153(1–2), pp. 49–104. doi: 10.1016/J.ARTINT.2002.12.001.

Liu, B. and Jiang, C., 2010. A knowledge base maintenance system for fault diagnosis expert system, in *The 2nd International Conference on Information Science and Engineering*. IEEE, pp. 21–24. doi: 10.1109/ICISE.2010.5690231.

Mach-Król, M. and Michalik, K., 2015. Validation and Verification of Temporal Knowledge as an Important Aspect of Implementing a Temporal Knowledge Base System Supporting Organizational Creativity, in *2015 Federated Conference on Computer Science and Information Systems (FedCSIS)*. IEEE, pp. 1315–1320. doi: 10.15439/2015F78.

Nazaruka, E., 2017. Meaning of Cause-and-effect Relations of the Topological Functioning Model in the UML Analysis Model, in *Proceedings of the 12th International Conference on Evaluation of Novel Approaches to Software Engineering*. SCITEPRESS - Science and Technology Publications, pp. 336–345. doi: 10.5220/0006384403360345.

Nazaruka, E., Ovchinnikova, V., Alksnis, G. and Sukovskis, U., 2016. Verification of BPMN Model Functional Completeness by using the Topological Functioning Model, in *Proceedings of the 11th International Conference on Evaluation of Novel Software Approaches to Software Engineering*. Portugal: SCITEPRESS - Science and and Technology Publications, pp. 349–358. doi: 10.5220/0005930903490358.

Nazaruks, V., 2017. The Knowledge Frame System based on Principles of Topological Functioning Model, *Applied Computer Systems*, 21, pp. 28–37. doi: 10.1515/acss-2017-0004.

Nazaruks, V. and Osis, J., 2017. Joint Usage of Frames and the Topological Functioning Model for Domain Knowledge Presentation and Analysis, in *Proceedings of the 12th International Conference on Evaluation of Novel Approaches to Software Engineering - Volume 1: MDI4SE*. Porto, Portugal: SCITEPRESS - Science and Technology Publications, pp. 379–390. doi: 10.5220/0006388903790390.

Osis, J., 1969. Topological Model of System Functioning (in Russian), *Automatics and Computer Science, J. of Academia of Siences*, (6), pp. 44–50.

Osis, J. and Asnina, E., 2008. Enterprise Modeling for Information System Development within MDA, in *Proceedings of the 41st Annual Hawaii International Conference on System Sciences (HICSS 2008)*. Waikoloa, USA: IEEE, pp. 490–490. doi: 10.1109/HICSS.2008.150.

Osis, J. and Asnina, E., 2011. Topological Modeling for Model-Driven Domain Analysis and Software Development : Functions and Architectures, in *Model-Driven Domain Analysis and Software Development: Architectures and Functions*. Hershey, PA: IGI Global, pp. 15–39. doi: 10.4018/978-1-61692-874-2.ch002.

Osis, J., Asnina, E. and Grave, A., 2008. Computation Independent Representation of the Problem Domain in MDA, *e-Informatica Software Engineering Journal*, 2(1), pp. 29–46. Available at: http://www.e-informatyka.pl/index.php/einformatica/volumes/volume-2008/issue-1/article-2/ (Accessed: 4 January 2018).

Osis, J. and Slihte, A., 2010. Transforming Textual Use Cases to a Computation Independent Model, in Osis, J. and Nikiforova, O. (eds) *Model-Driven Architecture and Modeling-Driven Software Development: ENASE 2010, 2ndMDA&MTDD Whs.* SciTePress, pp. 33–42.

Ovchinnikova, V. and Nazaruka, E., 2016. The Validation Possibility of Topological Functioning Model using the Cameo Simulation Toolkit, in *Proceedings of the 11th International Conference on Evaluation of Novel Software Approaches to Software Engineering*. SCITEPRESS - Science and and Technology Publications, pp. 327–336. doi: 10.5220/0005926003270336.

Santos, E. and Dinh, H. T., 2008. On automatic knowledge validation for Bayesian knowledge bases, *Data & Knowledge Engineering*. North-Holland: Elsevier, 64(1), pp. 218–241. doi: 10.1016/J.DATAK.2007.07.004.

Sarkar, S. and Ramaswamy, M., 2000. Knowledge Base Decomposition to Facilitate Verification', *Information Systems Research*, 11(3), pp. 260–283. doi: 10.1287/isre.11.3.260.12207.

Skorupski, J., 2015. Automatic verification of a knowledge base by using a multi-criteria group evaluation with application to security screening at an airport, *Knowledge-Based Systems*, 85, pp. 170–180. doi: 10.1016/j.knosys.2015.05.004.

Tadj, C. and Laroussi, T., 2006. Dynamic Verification of an Object-Rule Knowledge Base Using Colored Petri Nets, *Journal of systemics, Cybernetics and Informatics*. International Institute of Informatics and Cybernetics, 4(3), pp. 23–31.

Wobcke, W., 1994. A conditional logic for planning and plan recognition, in *Proceedings of ANZIIS '94 - Australian New Zealnd Intelligent Information Systems Conference*. IEEE, pp. 382–386. doi: 10.1109/ANZIIS.1994.396993.

Wu, Q., Zhou, C., Wu, J. and Wang, C., 2005. Study on Knowledge Base Verification Based on Petri Nets, in *2005 International Conference on Control and Automation*. IEEE, pp. 997–1001. doi: 10.1109/ICCA.2005.1528267.

Zhong, B. T., Ding, L. Y., Love, P. E. D. and Luo, H. B., 2015. An ontological approach for technical plan definition and verification in construction, *Automation in Construction*, 55, pp. 47–57. doi: 10.1016/j.autcon.2015.02.002.