Joint Usage of Frames and the Topological Functioning Model for Domain Knowledge Presentation and Analysis

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

Keywords: System Modelling, Frames, Knowledge Base, Topological Functioning Model.

Abstract: Joint usage of frames and Topological Functioning Model (TFM) provides proper analysis of knowledge in the domain under study. The main issue in domain knowledge analysis is completeness of discovered knowledge. Formal representation of the knowledge in frames allows automated construction and validation of the TFM, thus allowing to discover white places in knowledge. Analysing TFM metamodels, the structure of the frame system for generation of the TFM is proposed. The frame system leads to highlighting structural knowledge, while validation of the generated TFM shows white places in behavioural knowledge. Validation of the TFM does not guarantee the complete identity of obtained knowledge to the domain, since the knowledge is based on expert opinions. Thus, analysis of the problem domain is shifted from the separate investigation of dynamic and structural aspects of the system to holistic understanding of domain phenomena. The presented results should be refined if other derived models are added.

1 INTRODUCTION

Complex business domain logic that is necessary for software development can be and must be discovered during systems analysis and specified by models as accurate as possible. The goal of systems analysis is not only to discover some limited knowledge about the domain and requirements to the software, but also to make implicit domain knowledge clear and unambiguous, thus raising the probability of its completeness.

The more formal is a model, the higher is its accuracy. The suggested technique uses formal models for knowledge representation, modelling and analysis, namely a Topological Functioning Model (TFM) and knowledge frames. But the question is how both these formal means can be jointly used to lead to the more complete discovering of knowledge.

There are many formats for knowledge representation, e. g. knowledge frames, ontology, product rules, first-order predicate logic, high-order predicate logic, fuzzy logic, modal logic, etc. Every format has its own limitations and benefits (Okafor & Osuagwu 2007). Some of them (e. g. ontology, logic-based structures) are dedicated to representation of declarative domain knowledge, while other (e. g. product rules) are dedicated to representation of procedural domain knowledge.

Due to evolution of web technologies, the ontology has gained the greater popularity. There are many domain ontologies, knowledge of which can be used, and can enhance other knowledge bases.

From one point of view, the ontology can supplement the TFM well, since the model is focused more on the functionality and domain object participation in it than on the structural relationships among objects. However, the construction of the TFM requires analysis of domain information and extraction of both (procedural and declarative) knowledge, but here the functional view on the system is the primary one. During transformation of the TFM into software analysis and design models, this view must be changed. The functional view on the system must be assigned to the structural view on the system. The ontology, compared to the knowledge frames, cannot hold procedural knowledge together with the declarative one. Therefore, the proposed approach foresees the use of the knowledge frames. However, it does not mean that it cannot use ontologies for some specific tasks.

The research describes the related work (Section 2), the common vision of the proposed technique (Section 3) of a joint use of the frame system and the TFM (Section 4), and illustrates it by a small example

Nazaruks, V. and Osis, J.
Joint Usage of Frames and the Topological Functioning Model for Domain Knowledge Presentation and Analysis.
DOI: 10.5220/0006388903790390
In *Proceedings of the 12th International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE 2017)*, pages 379-390
ISBN: 978-989-758-250-9

(Section 5). Conclusions (Section 6) highlight some benefits and limitations of the proposition.

2 RELATED WORK

The most common way of entering knowledge into the frame system is manual, i. e. a knowledge engineer enters facts and assertions about the domain based on results of interviews with domain experts and other information about the domain (Beltrán-Ferruz et al. 2004; Shiue et al. 2008; Bimba et al. 2016; Detwiler et al. 2016). Only in case of text analysers automated entering is applied, but the amount of human participation in this process is not clear (Grigorova & Nikolov 2007; Xue et al. 2010; Xue et al. 2012; Corcoglioniti et al. 2016).

Frame-based representation of declarative and procedural knowledge has a wide application, but the last decade tendency is health care and biomedicine (mostly for ontologies of terms) (Beltrán-Ferruz et al. 2004; Bimba et al. 2016), forecasting (Kim et al. 2008; Bimba et al. 2016) and text/natural language processing (Kramer & Kaindl 2004; Marinov 2004; Marinov 2008; Gennari et al. 2005; Tettamanzi 2006; Grigorova & Nikolov 2007; Xue et al. 2010; Xue et al. 2012; Rector 2013; Sim & Brouse 2014; Bimba et al. 2016; Al-Saqqar et al. 2016).

Limitations mentioned by authors are inadequate representation of knowledge (Kramer & Kaindl 2004), greater expressiveness that can lead pure ontologies to the loss of information in case of transformation into them (Gennari et al. 2005; Bimba et al. 2016; Detwiler et al. 2016), necessity to work with the completely known characteristics and static knowledge domain (Grigorova & Nikolov 2007), representation of the procedural *knowledge as programming code inside frames* (Grigorova & Nikolov 2007), and the fact that complex structures can decrease the performance of the system inference and execution (Shiue et al. 2008; Xue et al. 2010).

There could be integration with other knowledge representation systems such as product rules and business constraints (Hernández & Serrano 2001), OWL (Hernández & Serrano 2001; Corcoglioniti et al. 2016; Detwiler et al. 2016), fuzzy logic (Tettamanzi 2006), and modal logic (Al-Saqqar et al. 2016).

The related work illustrates that frame systems are still in use. There are made optimistic attempts to adapt this knowledge representation format to new technologies, which allows integrating frame-based knowledge systems with already existing ontologies and other knowledge representation techniques.

This means that frame systems can be applied also for our purpose considering enumerated limitations and possibilities.

3 JOINT USAGE OF FRAMES AND TFM

The TFM is a formal model which describes the functioning of a system. Its fundamentals are published in (Osis 1969). The TFM can be specified as a topological space (X, Θ), where X is a finite set of functional features of the system under consideration, and Θ is a topology 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 & Asnina 2011). It can be specified by a unique tuple $\langle A, R, O, PrCond, PostCond, Pr, Ex \rangle$, where:

- A is an action linked with an object,
- R is a result of the action A,
- O is an object (objects) that gets the result of the action or an object (objects) that is used in this action,
- $PrCond$ is a set of preconditions or atomic business rules,
- $PostCond$ is a set of postconditions or atomic business rules,
- Pr is a set of responsible entities (systems or subsystems) that provide or suggest an action with a set of certain objects,
- Ex is a set of responsible entities (systems or subsystems) that enact a concrete action (Osis & Asnina 2011), (Nazaruka et al. 2016).

A TFM is *valid* when it satisfies topological and functioning properties (Osis & 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. The possibility of validation of the TFM using execution model simulation is discussed in (Ovchinnikova & Nazaruka 2016), where decision making is based on results presented in (Asnina & Ovchinnikova 2015).

In this research, three approaches for complex system modelling are considered, namely TFM4MDA, TopUML and IDM.

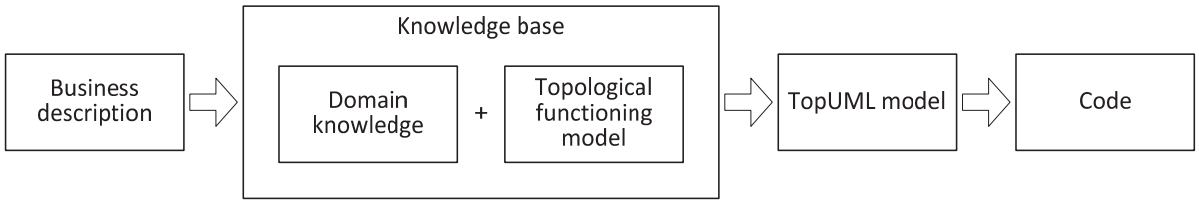

Figure 1: General vision of joint usage of TFM and frame system.

The Topological Functioning Modelling for Model Driven Architecture (TFM4MDA) approach defined in (Osis & Asnina 2008b) is intended for problem domain analysis and modelling in the context of MDA. It makes possible to use a formal TFM as a Computation Independent Model (CIM).

The TopUML approach proposed in (Donins 2012) combines TFM and its formalism with elements and diagrams of the TopUML modelling language, which is a specially developed profile of the Unified Modelling Language (UML). The TopUML modelling method "covers modelling and specification of systems in computation independent and platform independent viewpoints" (Donins 2012).

The Integrated Domain Modelling (IDM) approach is proposed in (Slihte 2015); the goal of this approach is "to provide an efficient way to acquire a domain model based on declarative and procedural domain knowledge". The approach "suggests using common system analysis and artificial intelligence practices to capture the domain knowledge and then transform these into a corresponding domain model" (Slihte 2015).

Figure 1 illustrates the general vision of how the TFM and frame-based knowledge base can be used for software development. The main idea is to represent knowledge of the domain under study as a knowledge frame system. The system holds also knowledge that is specific to the TFM, such as causal dependencies, cycles, inputs, and outputs. Using this knowledge base, the TFM can be generated automatically. The TFM itself and other knowledge in the frame system serve as a source for constructing a design model in TopUML. The TopUML model is planned to be transformed to the source code of the software. Reverse engineering from UML sequence diagrams to the TFM is considered in (Ovchinnikova & Asnina 2015).

The presence of the knowledge base lets store the knowledge in a structured and formal manner, while allowing to check it for inconsistencies.

The closure of the topological space over a set of system's inner functional features, as well as the TFM representation as a graph, could allow a modeller to find these inconsistencies. The joint use of the frame system and the TFM for this purpose is illustrated in Figure 2.

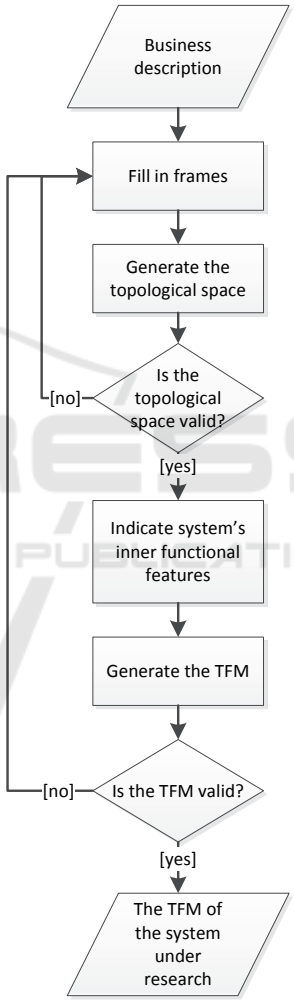

Figure 2: The process of checking the inconsistencies.

The idea is that generation and further validation of the TFM in compliance with its metamodel would allow seeing places where the business knowledge is implicit. The valid topological space is a requirement for generating the TFM. However, it does not guarantee that the generated TFM will be valid. For example, let us consider the situation shown in Figure

3. Here, the topological space is valid (Figure 3, part A): all functional features are connected, and it does have a cycle, inputs and outputs. After the closure of the set of system's inner functional features (vertices labelled with S), the TFM has been separated into two parts. This indicates lost cause and effect relations between system's functional features. The graphical representation allows seeing the possibly related functional features (Figure 3, bold-lined vertices in part B).

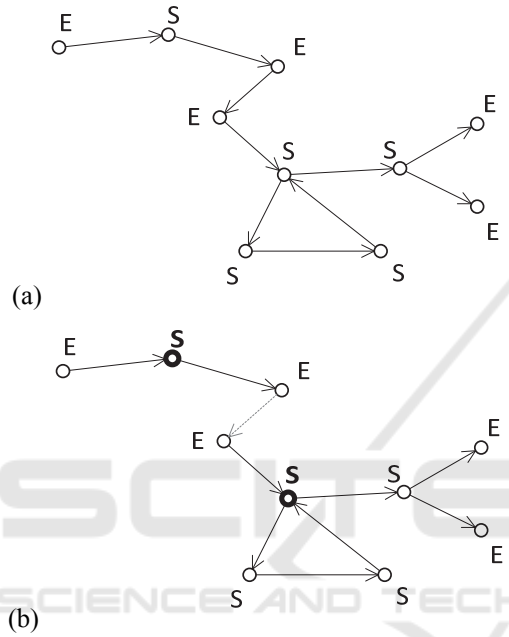

Figure 3: The invalid abstract TFM generated from the valid topological space.

4 DOMAIN KNOWLEDGE PRESENTATION

To understand what knowledge is to be kept in frame system, let us consider what is needed for construction of the TFM. First, let us compare knowledge units that are kept in TFM from its metamodel perspective, and then from viewpoint of functional feature definitions.

4.1 Comparison of TFM Metamodels

At the present, there are three TFM metamodels, each developed for a concrete TFM application: TFM4MDA (Asnina 2006; Osis et al. 2007), TopUML (Osis & Donins 2010b; Donins 2012) and IDM approach (Osis et al. 2012; Slihte 2015). The analysis of them showed the following common elements (Table 1): the topological functioning model (row 1), the functional feature (row 2), the cycle (row 4), the actor (row 9), the logical relationship (row 21), and the topological relationship (row 22), as well as two enumerations "Subordination" (row 14) for functional feature subordination within the system and "LogicalOperation (row 20) for specification of logical operation on the set of topological relationships. Other elements are specific for each approach. In case of TFM4MDA they are required for transformation from TFM to use case specifications (rows 9–13), in case of TopUML for transformation to TopUML diagrams (rows 15–19), and in case of IDM approach the TFM is a target of the transformation (therefore, the metamodel of use cases, the source of the transformation, is not considered here).

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

4.2 Comparison of Functional Feature Definitions

The definition of a functional feature of the TFM in the form of a *n*-ple has been introduced and elaborated by several authors, its historical aspects are discussed in more detail in (Solomencevs 2016). Here we want to understand what knowledge and for what artefacts functional feature's *n*-ple may contain. The results of this analysis will be applied to the frame system to separate knowledge that are necessary to the pure TFM, and knowledge that can be related to the previous one to infer some additional knowledge, e. g. necessary for generating software analytical diagrams, or for checking software functional requirements for incompleteness.

Table 2 illustrates elements of *n*-ples and corresponding artefacts. Elements that are necessary for the pure TFM are A, R, O, **PrCond**, E, **PostCond**, S, Pr; and will be presented in the suggested frame system. The bold font indicates sets of elements. Other elements may be generated from these. Artefacts such as the TDM and analytical diagrams use all the elements presented.

Table 1: Comparison of metaclasses and datatypes of the three TFM metamodels (Asnina 2006; Donins 2012; Slihte 2015), where domains are denoted as P for the problem domain and S for the solution domain, as well as artefacts are TFM for the topological functioning model, REQ for functional requirements, and APL for application analytical diagrams.

N.	TFM4MDA metamodel 2006	TFM metamodel 2012	TFM metamodel 2015	Domain	Artefacts
1	TFMTopologicalFunctioningModel	TopologicalFunctioningModel	TFM	P, S	TFM
2	TFMFunctionalFeature	FunctionalFeature	FunctionalFeature	P, S	TFM
3	TFMFunctionalFeatureSet			P, S	TFM
4	TFMCycle	TopologicalCycle	Cycle	P, S	TFM
5	TFMCorrespondence			P, S	TFM, REQ
6	TFMUserGoal			S	REQ
7	TFMUserSystemGoal			S	REQ
8	TFMUserBusinessGoal			S	REQ
9	TFMUserRole	Actor	Actor	P, S	TFM, REQ
10	TFMBusinessActor			S	REQ
11	TFMBusinessWorker			S	REQ
12	TFMFunctionalRequirement			S	REQ
13	Enumeration “Benefit”			S	REQ
14	Enumeration “Subordination”	Enumeration “Subordination”	Enumeration “Subordination”	P, S	TFM
15		ActionResult		S	APL
16		Class		S	APL
17		State		S	APL
18		Condition		S	APL
19		TopologicalOperation		S	TFM, APL
20		Enumeration “RelationType”	Enumeration “LogicalOperation”	P, S	TFM
21		LogicalRelationship	LogicalRelationship	P, S	TFM
22		TopologicalRelationship	Topological Relationship	P, S	TFM

Object O has the responsibility to execute action A to get the outcome — result R. Therefore, there is a single object type that could represent also a set of objects (e.g. a collection).

Result R of action A can be defined as “a thing that is caused or produced by something else; a consequence or outcome” (Oxford University Press 2013), “expressed in client-valued terms” (Luca 2002). Therefore, in the frame system, the result can be represented by any type. This statement is not in contradiction with the assumption that the result can be an attribute of a class or another class made in (Donins 2012). Thus, topological operation Op of class Cl (equals to object O) should be named as a union of names of action A and result R. For example, in case of functional feature “Calculating the maximum per year for sales”, the topological operation of object collection Sales will be *Sales::calculateMaximumPerYear()* and result R will be derived attribute *Sales::/maximumPerYear* of some numerical type.

Creating explicit assignments between frames of functional features and objects gives the possibility to look at the functionality of the system from the structural viewpoint, as well as to consider causal dependencies among objects from the functioning viewpoint.

4.3 The Frame System

According to the information defined in previous sections, the following frame classes have been developed:

- CauseAndEffectRelation (Table 3) — for knowledge on cause-and-effect relations generated from instances of frame FunctionalFeature;
- FunctionalFeature (Table 4) — for facts about the functional features;
- Object (Table 5) — for objects that participate in the functional feature execution;

Table 2: Use of elements of *n*-ples of a functional feature for artefacts that are denoted as TFM for topological functioning models, TCD for topological class diagrams, SD for state diagrams, CD for communication diagrams, SeD for sequence diagrams, AD for activity diagrams, TUCD for topological use case diagram.

N.	Element	Description	References	Artefacts
1	A	The action	(Osis & Asnina 2008b; Osis & Donins 2010a; Donins et al. 2012; Slihte 2015)	TFM, AD, SD *necessary to TCD
2	R	The result of action A	(Osis & Asnina 2008b; Osis & Donins 2010a; Donins et al. 2012; Slihte 2015)	TFM, TCD
3	O	The object that gets result R or that is used in action A	(Osis & Asnina 2008b; Osis & Donins 2010a; Donins et al. 2012; Slihte 2015)	TFM *necessary to TCD
4	**PrCond**	The set of **pre**conditions or atomic business rules	(Osis & Asnina 2008b; Osis & Donins 2010a; Donins et al. 2012; Slihte 2015)	TFM, AD, SD
5	E	The entity that is responsible for execution of action A	(Osis & Asnina 2008b; Osis & Donins 2010a; Donins et al. 2012; Slihte 2015)	TFM, TUCD
6	**PostCond**	The set of **post**conditions	(Osis & Asnina 2008a; Osis & Donins 2010a; Donins et al. 2012; Slihte 2015)	TFM
7	S	Subordination of the functional feature, i.e. its location in relation to the system under the study: inner or external	(Osis & Asnina 2008a; Donins et al. 2012; Slihte 2015)	TFM
8	Cl	The class which will represent object O in static viewpoint of the system	(Osis & Donins 2010a; Donins et al. 2012)	TCD, CD, SeD *is equal to object O
9	Op	The operation of the class Cl	(Osis & Donins 2010a; Donins et al. 2012)	TCD, CD, SeD *is equal to action A with its result R
10	St	The new state of object O after execution of action A	(Donins et al. 2012)	SD
11	Es	The indicator that execution of action A can be automated	(Donins et al. 2012)	SD
13	Pr	A set of responsible entities that provide or suggest action A with or for object O	(Osis & Asnina 2011)	TFM

- Property (Table 6) — for domain object properties;
- TopologicalOperation (Table 7) — for knowledge about the operations that will implement actions of the functional features; the instances of this frame class are to be generated based on the values of frame FunctionalFeature values;
- TopologicalCycle (Table 8) — for holding facts about functional feature participation in cycles of functionality.

Each frame class is identified by its name, it contains slots, fillers and facets. Now, frame classes hold only static knowledge on the domain, i. e. ontology. Further, integration with the product (business) rules must be implemented.

There are several frame classes which slot values must be set and updated automatically. The first one is CauseAndEffectRelation, which slot values are generated based on the facts that the cause is predefined by using a precondition, while the effect is specified by using a postcondition (Donins 2012). So, the frame instance is created and filled in when there is a case of the equal precondition and a postcondition.

Table 3: Frame class for cause-and-effect relations (generated).

Class: CauseAndEffectRelation		
identifier		
causeFeature		FunctionalFeature
effectFeature		FunctionalFeature
preCondition		
postCondition		

Table 4: Frame class for functional features (filled in).

Class: FunctionalFeature		
identifier		
action		
result		anyType
object		Object
preConditionSet		
postConditionSet		
provider		Object (role=provider)
executorSet		collection of Object (role=executor OR actor)
subordination	{inner, external}	

Table 5: Frame class for objects (filled in and generated).

Class: Object	
identifier	
name	
role	{none, executor, actor, provider}
state	enumeration
currentState	
properties	Collection of Property
topologicalOperations	Collection of TopologicalOperation

Table 6: Frame class for properties (filled in).

Class: Property	
name	
type	anyType
value	

Table 7: Frame class for topological operations (generated).

Class: TopologicalOperation	
name	
owner	Object
returnType	anyType

Table 8: Frame class for topological cycles (filled in and partially generated).

Class: TopologicalCycle		
identifier		
isMain		
order		
functionalFeatures		collection of 2 and more FunctionalFeature instances

The second one is frame TopologicalOperation, where the *name* must be set as a union of values of slots *action* and *result* of FunctionalFeature. The slot *owner* gets his value based on the value of slot *object* in FunctionalFeature frame, but the slot *returnType* by a type of the value of slot *result*.

Frames whose slot values are partially generated are Object (the value of slot *topologicalOperation*), and TopologicalCycle (the value of slot *functionalFeatures*). The latter is possible since in all the metamodels the topological cycle is represented as a set of functional features that are involved in some closed path.

Identifiers in all frames are to be also automatically generated.

This frame system represents only facts about the domain, not scripts or daemons.

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 sends the case to the court. The criminal case is terminated when the court adjudicates in the case. The Chief of Department assigns an investigator to a stated criminal act, and then the investigator initiates a new criminal case. The decision is based on the availability of investigators, since each investigator informs Chief when the criminal case is sent to the prosecutor.”

Having this knowledge about the system, a modeler can fill in frame instances for functional features and objects (see Table 9 and Table 10). The corresponding generated topological space of the system functional features is shown in Figure 4. The topological space is valid, it contains no isolated vertices, has inputs, outputs, and a cycle structure.

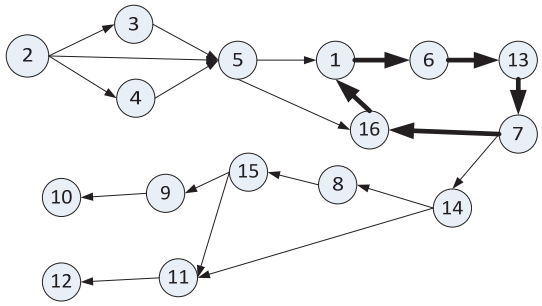

Figure 4. The topological space obtained from the facts in the frame system.

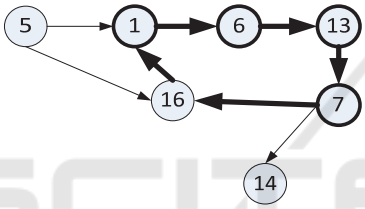

Figure 5. The TFM obtained after closuring.

The next step is to indicate system's inner functional features. Let us imagine that we need to generate the TFM of the investigator work. Then the inner functional features are 1, 6, 7, and 13. The value of slot "subordination" of frame instances of them should be set to "inner". After the closuring, the corresponding TFM (Figure 5) is obtained.

Though the model is small, it is valid, namely, it does contain input (functional feature 5), output (functional feature 14), the functioning cycle (1-6-13-7-16-1), and has no isolated functional features.

6 CONCLUSIONS

In this paper, three existing TFM metamodels were compared with each other, and common elements were discovered. At the present, the frame system contains only static information; some frame instances can be filled in with fully or partially generated knowledge. This structure is sufficient to generate TFM from it; however, the generated TFM does not represent any logical operations on cause-and-effect relations, since now the suggested frame system does not hold the required constructs.

A joint use of the knowledge frame system and the TFM has the following benefits. First, the nature of the knowledge frame system as a closed system does not allow infering ambiguous statements, i. e. if something is not stated as true in the system, then this means that it is false. In case if this proposition does not correspond to the real phenomena, it would lead to further investigation of the domain phenomena and rules. Second, the structure of the frame system is similar to the object-oriented way of thinking. This allows software developers to operate with elements of the frame system in the known way. Third, principles that lie in the topological functioning modelling lead to the more complete discovering of knowledge. The sequential validation of the topological space and the topological model of the system functioning could help in discovering potential "holes" in the presented knowledge. Fourth, The TFM supplemented with the declarative and procedural knowledge from the frame system serves as the formal root view on the system at the very beginning of the development.

However, there are limitations, too. One of the difficulties in development of frame instances of functional features is to define the proper client-valued result of the action and the corresponding object. Another one is handling business rules and logical operators on cause-and-effect relations. The third one is the implementation of the frame system and a use of proper inference engine. The fourth, integration with already existing ontologies that use an open world paradigm, thus allowing to infer potentially untrue statements.

The future research will be related to extending the frame system with procedural knowledge, i. e. logical operations on cause-and-effect relations, scripts that will set and update values of the corresponding slots with facts about the domain, and business rules definitions separately from the frame classes to provide its greater flexibility and maintainability.

Table 9: Frame instances for FunctionalFeatures.

identifier	action	result	object	preConditionSet	postConditionSet	provider	executor-Set
1.	Initiating	[new]	Criminal Case	a criminal act is stated AND an investigator is assigned	*a criminal case is initiated*	State Police	Investigator
2.	Committing	[new]	Criminal Act		*a criminal act is committed*		Criminal-Person
3.	Discovering	[new]	Criminal Act	a criminal act is committed	*a criminal act is discovered*	State Police	
4.	Submitting	[new]	ClaimOn Criminal Act	a criminal act is committed	*a claim about a criminal act is submitted*	State Police	victim, witness
5.	Stating	[new]	Criminal Act	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.	Conducting	investigative Actions	Criminal Case	a criminal case is initiated	*an indicted person is found*	State Police	
7.	Sending	toProsecutor	Criminal Case	an investigation is completed	*a criminal case is sent to prosecutor*	State Police	Investigator
8.	Drawing up	penalOrder	Criminal Case	a criminal act is misdemeanour OR a criminal act is average gravity	*a penal order is drawn*	Prosecution Office	Prosecutor
9.	Terminating		Criminal Case	an indicted person agrees with the penal order	*a criminal case is terminated*	Prosecution Office	Prosecutor
10.	Serving		Punishment	a criminal case is terminated		Prisons Administration	Convicted-Person
11.	Sending	to the-Court	Criminal Case	NOT(an indicted person agrees with the penal order) OR a criminal act is grave	*NOT (a criminal case is terminated) AND a criminal case is sent to the court*	Prosecution Office	Prosecutor
12.	Adjudicating		Criminal Case	*NOT (a criminal case is terminated) AND a criminal case is sent to the court*	*a criminal case is terminated*	Court	
13.	Completing	investigative Actions	Criminal Case	an indicted person is found	*an investigation is complete*	State Police	Investigator
14.	Assessing	gravety	Criminal Act	a criminal case is sent to prosecutor	*a criminal act is misdemeanour OR a criminal act is average gravity OR a criminal act is grave*	Prosecution Office	Prosecutor
15.	Signing	agreement	Penal-Order	a penal order is drawn	*an indicted person agrees with the penal order OR NOT(an indicted person agrees with the penal order)*	Prosecution Office	Indicted-Person
16.	Assigning	investigator	Criminal Case	a criminal case is sent to prosecutor OR a criminal act is stated	*an investigator is assigned*	State Police	Chef of Department

Table 10: Frame instances for Object.

identifier	name	role	state	currentState	properties	topologicalOperations
1	CriminalCase	none	isInitiated, isTermina-ted			initiate[new](); conductInvestigative-Actions(); sendToProsecutor(); drawUpPenalOrder(); terminate(); sendToCourt(); adjudicate(); assignInvestigator()
2	Investigator	executor/actor				
3	CriminalAct	none	isStated, isCommit-ted, isMisde-meanour		gravety = {misdemeanour, average, grave}	commitNew(); discoverNew(); stateNew(); assessGravity()
4	CriminalPerson	executor/actor				
5	ClaimOnCriminalAct	none	isSubmitted			submitNew()
6	InvestigativeActions	none				
7	Investigation	none	isComple-ted			
8	Prosecutor	executor/actor				
9	Victim	executor/actor				
10	Witness	executor/actor				
11	IndictedPerson	none				
12	PenalOrder	none	isDrawn		accusation, penalty	signAgreement()
13	Accusation		isPresented			
14	Penalty		isOffered			
13	Punishment	none				serve()
14	ConvictedPerson	executor/actor				
15	Court	none				
16	ChefOfDepartment	executor/actor				

REFERENCES

Al-Saqqar, F., Bentahar, J. & Sultan, K., 2016. On the soundness, completeness and applicability of the logic of knowledge and communicative commitments in multi-agent systems. *Expert Systems with Applications*, 43, pp.223–236. Available at: http://dl.acm.org/citation.cfm?id=2840535 [Accessed December 17, 2016].

Asnina, E., 2006. *Formalization of Problem Domain Modeling within Model Driven Architecture*. Riga Technical University.

Asnina, E. & Ovchinnikova, V., 2015. Specification of decision-making and control flow branching in Topological Functioning Models of systems. In *ENASE 2015 - Proceedings of the 10th International Conference on Evaluation of Novel Approaches to Software Engineering*. Lisbon: SciTePress, pp. 364–373. Available at: https://www.scopus.com/inward/record.uri?eid=2-s2.0-84933575203&partnerID=40&md5=8d49b05925e0362f60cb6f1528714891.

Beltrán-Ferruz, P.J., González-Calero, P.A. & Gervás, P., 2004. Converting Mikrokosmos frames into description logics. *Proceeedings of the Workshop on NLP and XML (NLPXML-2004): RDF/RDFS and OWL in Language Technology*, pp.35–42.

Bimba, A.T. et al., 2016. Towards knowledge modeling and manipulation technologies: A survey. *International Journal of Information Management*, 36(6), pp.857–871. Available at: http://linkinghub.elsevier.com/retrieve/pii/S026840121630336X [Accessed December 17, 2016].

Corcoglioniti, F., Rospocher, M. & Aprosio, A.P., 2016. A 2-phase frame-based knowledge extraction framework. In *Proceedings of the 31st Annual ACM Symposium on Applied Computing - SAC '16*. New York, New York, USA: ACM Press, pp. 354–361. Available at: http://dl.acm.org/citation.cfm?doid=2851613.2851845 [Accessed December 17, 2016].

Detwiler, L.T., Mejino, J.L.V. & Brinkley, J.F., 2016. From frames to OWL2: Converting the Foundational Model of Anatomy. *Artificial Intelligence in Medicine*, 69, pp.12–21. Available at: http://www.aiimjournal.com/article/S0933-3657(16)30152-X/abstract [Accessed December 17, 2016].

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

Donins, U. et al., 2012. Using functional characteristics to analyze state changes of objects. In *CEUR Workshop Proceedings. Databases and Information Systems. Tenth International Baltic Conference on Databases and Information Systems: Local Proceedings, Materials of Doctoral Consortium*. Vilnius: Žara, pp. 94–106.

Gennari, J.H., Mork, P. & Li, H., 2005. Knowledge transformations between frame systems and RDB systems. In *Proceedings of the 3rd international conference on Knowledge capture - K-CAP '05*. New York, New York, USA: ACM Press, p. 197. Available at: http://portal.acm.org/citation.cfm?doid=1088622.1088666 [Accessed December 17, 2016].

Grigorova, D. & Nikolov, N., 2007. Knowledge representation in systems with natural language interface. In *Proceedings of the 2007 international conference on Computer systems and technologies - CompSysTech '07*. New York, New York, USA: ACM Press, p. 1. Available at: http://portal.acm.org/citation.cfm?doid=1330598.1330670 [Accessed December 17, 2016].

Hernández, J.Z. & Serrano, J.M., 2001. Knowledge-based models for emergency management systems. *Expert Systems with Applications*, 20(2), pp.173–186. Available at: http://linkinghub.elsevier.com/retrieve/pii/S0957417400000579 [Accessed December 17, 2016].

Kim, K. et al., 2008. A frame-based probabilistic framework for spoken dialog management using dialog examples. In *Proceedings of the 9th SIGdial Workshop on Discourse and Dialogue*. Columbus, Ohio: Association for Computational Linguistics, Stroudsburg, PA, USA, pp. 120–127.

Kramer, S. & Kaindl, H., 2004. Coupling and cohesion metrics for knowledge-based systems using frames and rules. *ACM Transactions on Software Engineering and Methodology*, 13(3), pp.332–358. Available at: http://portal.acm.org/citation.cfm?doid=1027092.1027094 [Accessed December 17, 2016].

Luca, J. De, 2002. Feature Driven Development. *Feature Driven Development Processes*. Available at: http://www.featuredrivendevelopment.com/node/449 [Accessed January 8, 2017].

Marinov, M., 2008. Using frames for knowledge representation in a CORBA-based distributed environment. *Knowledge-Based Systems*, 21(5), pp.391–397. Available at: http://linkinghub.elsevier.com/retrieve/pii/S0950705108000154 [Accessed December 17, 2016].

Marinov, M., 2004. Using XML to represent knowledge by frames. In *Proceedings of the 5th international conference on Computer systems and technologies - CompSysTech '04*. New York, New York, USA: ACM Press, p. 1. Available at: http://portal.acm.org/citation.cfm?doid=1050330.1050350 [Accessed December 17, 2016].

Nazaruka, E. et al., 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. Available at: http://www.scitepress.org/DigitalLibrary/Link.aspx?doi=10.5220/0005930903490358 [Accessed February 20, 2017].

Okafor, E.C. & Osuagwu, C.C., 2007. Issues in Structuring the Knowledge-base of Expert Systems. *Journal of Knowledge Management*, 5(3), pp.313–322.

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. & Asnina, E., 2008a. A Business Model to Make Software Development Less Intuitive. In *2008 International Conference on Computational Intelligence for Modelling Control & Automation*. IEEE, pp. 1240–1245. Available at: http://ieeexplore.ieee.org/document/5172803/.

Osis, J. & Asnina, E., 2008b. 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. Available at: http://ieeexplore.ieee.org/document/4439190/.

Osis, J. & Asnina, E., 2011. Topological Modeling for Model-Driven Domain Analysis and Software Development : Functions and Architectures. In *Model-Driven Domain Analysis and Software Development*. Hershey, PA: IGI Global, pp. 15–39.

Osis, J., Asnina, E. & Grave, A., 2007. Formal computation independent model of the problem domain within the MDA. In *CEUR Workshop Proceedings*.

Osis, J. & Donins, U., 2010a. Formalization of the UML Class Diagrams. In *Evaluation of Novel Approaches to Software Engineering. 3rd and 4th International Conferences, ENASE 2008/2009, Funchal, Madeira, Portugal, May 4-7, 2008 / Milan, Italy, May 9-10, 2009. Revised Selected Papers*. Springer Berlin Heidelberg, pp. 180–192. Available at: http://link.springer.com/10.1007/978-3-642-14819-4_13 [Accessed January 8, 2017].

Osis, J. & Donins, U., 2010b. Platform Independent Model Development by Means of Topological Class Diagrams. In *Model-Driven Architecture and Modeling Theory-Driven Development : Proceedings of the 2nd International Workshop on Model-Driven Architecture and Modeling Theory-Driven Development (MDA & MTDD 2010)*. Lisbon: SciTePress, pp. 13–22.

Osis, J., Slihte, A. & Jansone, A., 2012. Using Use Cases for Domain Modeling. In *Proceedings of the 7th International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE 2012), Poland, Wrocław, 29-30 June*. Lisbon: SciTePress, pp. 224–231.

Ovchinnikova, V. & Asnina, E., 2015. The algorithm of transformation from UML sequence diagrams to the Topological Functioning Model. In *ENASE 2015 - Proceedings of the 10th International Conference on*

Evaluation of Novel Approaches to Software Engineering. Lisbon: SciTePress, pp. 377–384.

Ovchinnikova, V. & 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. Available at: http://www.scitepress.org/DigitalLibrary/Link.aspx?doi=10.5220/0005926003270336 [Accessed February 22, 2017].

Oxford University Press, 2013. Oxford Dictionaries. *Oxford Dictionaries*. Available at: https://en.oxforddictionaries.com/definition/result [Accessed January 8, 2017].

Rector, A., 2013. Axioms and templates: distinctions and transformations amongst ontologies, frames, and information models. In *Proceedings of the seventh international conference on Knowledge capture - K-CAP '13*. New York, New York, USA: ACM Press, p. 73. Available at: http://dl.acm.org/citation.cfm?doid=2479832.2479840 [Accessed December 17, 2016].

Shiue, W., Li, S.-T. & Chen, K.-J., 2008. A frame knowledge system for managing financial decision knowledge. *Expert Systems with Applications*, 35(3), pp.1068–1079. Available at: http://linkinghub.elsevier.com/retrieve/pii/S0957417407003247 [Accessed December 17, 2016].

Sim, W.W. & Brouse, P., 2014. Towards an Ontology-based Persona-driven Requirements and Knowledge Engineering. *Procedia Computer Science*, 36, pp.314–321. Available at: http://www.sciencedirect.com/science/article/pii/S1877050914013489 [Accessed October 7, 2016].

Slihte, A., 2015. *The Integrated Domain Modeling: an Approach & Toolset for Acquiring a Topological Functioning Model*. Riga Technical University.

Solomencevs, A., 2016. Topological Functioning Model for Software Development within MDA (Survey). In *Proceedings of the 11th International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE 2016)*. SciTePress, pp. 315–326.

Tettamanzi, A.G.B., 2006. A Fuzzy Frame-Based Knowledge Representation Formalism. In *Di Gesú V., Masulli F., Petrosino A. (eds) Fuzzy Logic and Applications. WILF 2003. Lecture Notes in Computer Science, vol 2955*. Springer Berlin Heidelberg, pp. 55–62. Available at: http://link.springer.com/10.1007/10983652_8 [Accessed December 17, 2016].

Xue, Y., Ghenniwa, H.H. & Shen, W., 2010. A Frame-based Ontological view Specification Language. In *The 14th International Conference on Computer Supported Cooperative Work in Design*. IEEE, pp. 228–233. Available at: http://ieeexplore.ieee.org/document/5471972/ [Accessed December 17, 2016].

Xue, Y., Ghenniwa, H.H. & Shen, W., 2012. Frame-based ontological view for semantic integration. *Journal of Network and Computer Applications*, 35(1), pp.121–131. Available at: http://linkinghub.elsevier.com/retrieve/pii/S1084804511000488 [Accessed December 17, 2016].