Chapter 2
Topological Modeling for Model-Driven Domain Analysis and Software Development: Functions and Architectures

Janis Osis
Riga Technical University, Latvia

Erika Asnina
Riga Technical University, Latvia

ABSTRACT
Model-driven software development has all chances to turn software development into software engineering. But this requires not only mature methodologies but also engineering models. An engineering model should satisfy five key characteristics, namely, abstraction, understandability, accuracy, predictiveness and inexpensiveness. This chapter discusses capabilities of a Topological Functioning Model (TFM) as such an engineering model for the purposes of domain analysis and software development in common. The TFM has functional and topological properties. The functional properties are cause-effect relations, cycle structure, inputs, and outputs. The topological properties are connectedness, closure, neighborhood, and continuous mapping. Thanks to its formal mathematical foundations, the TFM completely satisfies the mentioned characteristics of engineering models that is illustrated in the chapter.

INTRODUCTION
There are many ways how to describe semantics. In software development during the so called problem domain analysis mostly informal approaches and languages are used. There are several causes, and one of more important is that the problem domain itself is not well determined. Thus, developers explore the problem domain by 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 holistic and more formal representation.

Indeed, the question about possibility of formal description of semantics still exists. The important property of diagrams used in software development is that they must provide very precise or even formal sense. In (Diskin, Kadish, Piessens, &
Johnson, 2000) authors wrote that if some diagram $D$ exists, then it has some sense $M(D)$. This sense must be described in precise (it would be ideal, if mathematical) statements. This description makes some precise specification $S_D$ that has the precise semantic $M(S_D)$. Therefore, $M(S_D)$ is abstraction of formal intuitive sense $M(D)$ and $S_D$ may be considered as some (internal) logic specification, that is hidden in the diagram $D$. They pointed that this described approach is typical (or desired) for diagrams used in software engineering.

There are many intuitive worlds that can provide sense $M$ in the real world. The authors pointed out that in spite of that when one starts to think about $M$ formal description, this huge amount is being narrowed until several domains of mathematical constructs, e.g. set theory, type theory, high-order predicate logic or category theory, etc. All these languages are universal and expressive. Therefore, any formal semantic (some specification $S_D$) can be specified in any formal language of them.

But if we want to consider software development as an engineering discipline, then we need to take into account that not every formal language can be accepted for domain analysis by using models. Bran Selic (2003) enumerated five key characteristics that a useful and effective engineering model needs to satisfy. They are abstraction, understandability, accuracy, predictiveness and inexpensiveness (Selic, 2003):

- **Abstraction** is the most important characteristic. Usually, abstraction is the only available means of dealing with complex functionality and the structure of the system.
- **Understandability** makes the abstracted model expressive, reducing the amount of intellectual effort needed for model understanding.
- **Accuracy** makes the model useful. The model has to provide realistic representation of the modeled system features of interest.
- The model should be able to be used to predict the modeled system interesting implicit properties either through experimentation or through some formal analysis. In this case, the mathematical model is much better at predicting.
- The model should be **inexpensive**, so that it must be significantly cheaper in constructing and analysis than the modeled system.

Thus, an engineering model suitable for model-driven engineering is a formal mathematical model that supports abstraction and predictiveness, and at the same time it is understandable for all stakeholders (depending on the context), and cheap enough to be used for industrial tasks. As Unified Modeling Language (UML) developers’ and academic practice shows, one of such models could be Petri Nets and their derivations with some limitations in using. As our practice shows (and that what we demonstrate in this chapter), another one could be topological models of systems.

The chapter is organized as follows. The next section discusses advantages and weaknesses of Petri nets, and the background of topological models. Main properties and capabilities of the topological model, which support an engineering viewpoint on system functionality and structure, are described in section “Domain analysis with topological modeling”. The general framework of application of the topological model of system functioning for model-driven engineering is illustrated in section “Model-driven software development with topological modeling”.

## BACKGROUND

### Past and Present of Petri Nets

There is one formal mathematical graphical model accepted in software development for analysis of
activity flows and system states. These are Petri nets, especially Colored Petri Nets (CPNs). Basic concepts and practical use of the CPNs are discussed in (Jensen, 1997). To achieve application of Petri nets for modeling industrial tasks, their authors have combined the strength of Petri nets with the strength of programming languages. Petri nets provide the primitives for describing synchronization of concurrent processes, while a programming language provides the primitives for defining data types (color sets) and manipulating data values (Kristensen, Christensen, & Jensen, 1998). CPNs in modeling concurrency, synchronization, and communication in systems have undoubted advantages. CPNs and Petri nets are, however, also applicable more generally for modeling systems where concurrency and communication are key characteristics. Examples of this are business process/workflow modeling and manufacturing systems (Kristensen, Jurgensen, & Jensen, 2004). CPN notation is short. High-level CPNs support control and modularization of large system models. CPN models are descriptions of systems, thus they could be used as both specifications and presentation of systems. Moreover, the process of specification creation and analysis enables understanding of the modeled system. Model behavior can be analyzed by CPN model simulation or formal methods for graph analysis. And last, there are computer tools for creating, simulating and formal analyzing CPNs.

However, nevertheless development of Petri nets goes intensively during last forty years, this kind of nets have significant weaknesses. The cause is that a use of Petri nets as self-sufficient model is not quite convenient. In typical industrial applications, a CPN diagram consists of 10-100 pages (modules) with varying complexity (Kristensen, Christensen, & Jensen, 1998). First, a CPN model should be well-modularized, because only minor changes do not affect a structure of the CPN. However, if it is necessary to introduce large changes, then it will be quite hard work. Second, although a collection of Petri net constructs as well CPN constructs is relatively small, the way of its visualization is very important in case of large system design, since sometimes common representation of states and transitions takes modeler’s attention off one or another.

Past and Present of the Topological Model of System Functioning

Another formal mathematical model is a Topological Model of System Functioning (TFM). In this chapter we will use interchangeably several terms that defines the TFM, namely, a topological model, a topological model of system functioning and a topological functioning model. The TFM is completely an engineering model in Selic’s sense, and it was successfully used for system analysis from 1970s. The topological modeling of system functioning (TFM) was developed at Riga Technical University. First time its theoretic foundations were represented by Janis Osis in (Osis, 1969; Osis, 1972). Its application in different areas is being developed today as well.

A large number of high-quality diagnostic algorithms and methods based on this theory were developed (Osis, Gefandbein, Markovitch, & Novozhilova, 1991). Besides that, in 1970 a new approach of creation of the system theory foundations was suggested (Osis, 1970). One perspective direction that was begun in 1973 and published in (Osis, 1973), wherein the relation to the microprogramming was affected, is software engineering.

Since 1970 topological modeling was successfully applied for medicine problem solving by Zigurds Markovitch (Markovitcha & Markovitch, 1970). His work continued with mathematical model composition from mini-models of elements (Markovitch & Rekners, 1998) and expert knowledge (Markovitch & Stalidzans, 2000). Continuing topological modeling introducing into medicine, Zigurds Markovitch and Ieva Markovitcha successfully illustrated topological modeling
application for therapy selection (Markovitch & Markovitcha, 2000).

The topological modeling development was continued also by Janis Grundspenkis since the same 1970s. At the beginning his work was related to investigations in the field of cycle hierarchies for the purpose of rational diagnostic algorithm development (Grundspenkis, 1974) and (Grundspenkis & Blumbergs, 1981). The most recent interests are related to model-driven knowledge systematic acquisition and these solutions joining with knowledge management (Grundspenkis, 1996; Grundspenkis, 1997; Grundspenkis, 2004).

Besides that, the topological functioning model is being developed also in the field of the object-oriented analysis. The recent works are described in several sources: topological modeling application for business process modeling and simulation (Osis, Sukovskis, & Teilans, 1997), for the purposes of biological systems modeling (Osis & Beghi, 1997), the topological functioning model application in the software development for mechatronic and embedded systems (Osis, 2003), the topological functioning modeling application to introduce more formalism into the Model Driven Architecture framework and problem domain analysis, which grounds development of topological modeling language (Osis, 2004; Asnina, 2006a; Asnina, 2006b; Osis & Asnina, 2008; Osis, Asnina, & Grave, 2008; Asnina, 2009). We have to mention that unfortunately only publications after 1992 are available in English.

**DOMAIN ANALYSIS WITH TOPOLOGICAL MODELING**

The topological modeling is a modeling approach that uses a formal mathematical model to specify and analyze characteristics of a system. Before discussing what the topological modeling is, we need to refresh in minds what “topology” means.

**Figure 1. Two topological axioms**

\[ X \in Q ; \emptyset \in Q ; \]
\[ \forall \eta \left( \bigcup_{\eta} A_{\eta} \in Q \right) ; \forall \eta \left( \bigcap_{\eta=1} A_{\eta} \in Q \right) \]

**Mathematical Foundations**

**Topology** is a mathematical study of the properties that are preserved through deformations, twistings, and stretchings of objects. Tearing of objects is not allowed. Topology can be used to abstract the inherent connectivity of objects while ignoring their detailed form. For example, network topology used to describe connections of network nodes is commonly known application of topology studies in network modeling.

The “objects” of topology are often formally defined as topological spaces. A topological space is a set \( X \) together with a collection of open subsets \( A \). These subsets form topology \( Q \) in the set \( X \). In other words, topology in the set \( X \) is any system \( Q \) of open sets \( A \) of the set \( X \). The system \( Q \) has to satisfy two topological axioms (see Figure 1) stated by Andrei N. Kolmogorov (Kolmogorov, 1999).

The axiom (a) in Figure 1 states that \( X \) is a finite closed set of elements of the system under consideration with some certain topology \( Q \) among those elements, and this topology \( Q \) is set or defined, indeed. The axiom (b) in Figure 1 states that different systems (unions and intersections) of a finite number of open sets \( A \) of the set \( X \) belongs to the defined topology \( Q \).

In order to study problems related to topology graph theory is used. In the mathematics and computer science, graph theory studies the properties of graphs. Informally, a graph is a set of objects called vertices (or nodes) connected by links called arcs (or oriented edges). Usually, a graph is designed as a set of vertices connected by edges. A graph structure can be extended by assigning a weight to each edge, or by making the edges to the graph directional, technically...
called a digraph. A digraph with weighted edges is called a network.

Many applications of graph theory exist in the field of network analysis. They can be divided into two categories. The first is analysis for determination of structural properties of networks, and the second is analysis for discovering measurable quantities within networks.

Topological spaces in network analysis usually describe structural properties of systems, i.e., networks. However, the “subject” of the topological space may be also dynamical properties of systems, e.g. system functionality. Therefore, as theoretic foundations presented by Janis Osis (1969) state, a topological model of system functioning can be represented in the form of the topological space \((X, Q)\), where \(X\) is a finite set of functional features of the system under consideration, and \(Q\) is topology in the form of a directed graph. This model occurs through the acquisition of experts’ knowledge about the complex systems, verbal descriptions, and other documents concerning the structure and functioning.

Although the topological model of system functioning specifies functionality of the system, it also captures knowledge about the structure of the system. The topological modeling of functioning offers the formal, compact and comprehensive way to transform problem domain functioning processes (dynamics) into a structure (statics), i.e., information mapping from the model of functioning to the class diagram at the conceptual level. The main idea is that the functionality determines the structure of the planned system. Each system has the purpose of its existence. It relates as to living organisms (including human beings) as to mechanical, social and business systems. Each system functions in order to fulfill its purpose. In order to function successfully, the system needs to have a proper structure. Certainly, ways towards the purpose can differ, and the system is forced to achieve different goals which can also be changed. This means that the system should be able to change (remove, add, modify) its functions. Thus the structure of the system also may be changed according to these changed functions. However, it is clear that main functions of the system should be persistent in order for this system to be able to fulfill the purpose of its existence. This means that the structure of the system depends on its functionality.

**Abstract Topological Model of System Functioning**

Let us assume that an abstract finite closed set \(X = \{a, b, c, \ldots, z\}\) defined in topology \(Q\) is given. It is known that the simplest way of representation of topology \(Q\) can be direct listing of those subsets, which we consider as open subsets in \(X\). It is possible, for example, to set binary relations in the form of some hypothetic rules, which connect two points of the set \(X\). Then for any two different points, \(a\) and \(b\), which belong to \(X\), only one of four statements of binary relations (see Figure 2) will be true.

A binary relation can be represented in the form of an arc of a directed graph (digraph). In

**Figure 2. Four statements which are possible for two different elements in \(X\)**

```
\begin{tabular}{ccc}
\textbf{a)} & \textbf{b)} & \textbf{c)} & \textbf{d)} \\
\begin{tikzpicture}[node distance = 1.5cm, auto]
  \node (a) at (0,0) {a};
  \node (b) [right of=a] {b};
  \path[->] (a) edge node [left] {a R b} (b);
\end{tikzpicture} & \begin{tikzpicture}[node distance = 1.5cm, auto]
  \node (a) at (0,0) {a};
  \node (b) [right of=a] {b};
  \path[->] (a) edge node [left] {a R b} (b);
\end{tikzpicture} & \begin{tikzpicture}[node distance = 1.5cm, auto]
  \node (a) at (0,0) {a};
  \node (b) [right of=a] {b};
  \path[->] (a) edge node [left] {a R b} (b);
\end{tikzpicture} & \begin{tikzpicture}[node distance = 1.5cm, auto]
  \node (a) at (0,0) {a};
  \node (b) [right of=a] {b};
  \path[->] (a) edge node [left] {a R b} (b);
\end{tikzpicture} \\
\text{\textit{Denotation:}} & \text{\textit{Denotation:}} & \text{\textit{Denotation:}} & \text{\textit{Denotation:}} \\
R & R^* & a R b, b R a & a R b, b R^* a \\
\end{tabular}
```
Figure 2 digraphs that are corresponding to those four possible binary relations are illustrated above each statement.

An abstract topological model of system functioning visually can be represented as a directed graph \( G(X, U) \), where \( X \) is a finite closed set of elements with some certain topology \( Q \) among them, and \( U \) is a set of arcs that illustrates this topology. The set \( X \) is a union of a subset \( N \) of features of the system itself and a subset \( M \) of features of other systems, constituting the surrounding environment interacting with the system under consideration. This theoretical statement is explained below in more detail in Subsection “Separation of a topological model of system functioning”.

Elements of the Topological Model of System Functioning

Functional Features of the System

When we speak about functioning, we need to discuss what a concept “function” means. A “function” is the natural purpose (of something) or the duty (of a person). “Functional” means designed for or capable of a particular function or a use. This means that a functional feature is a characteristic of the system (in its general sense) that is designed and necessary to achieve some system’s goal. Functional features must be defined in accordance with the verbs (actions) defined in the description of the system. The unique nature of each functional feature defined in (Asnina, 2006b) is extended in this chapter.

Each functional feature is a unique tuple \( <A, R, O, PrCond, PostCond, Pr, Ex> \), where:

- \( A \) is an action linked with an object;
- \( R \) is a result of that action (it is an optional element);
- \( O \) 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 \( PrCond = \{ c_1, ..., c_i \} \), where \( c_i \) is a precondition or an atomic business rule (it is an optional element);
- \( PostCond \) is a set \( PostCond = \{ c_1, ..., c_i \} \), where \( c_i \) is a post-condition or an atomic business rule (it is an optional element);
- \( Pr \) 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.

A name of the functional feature could be expressed in the following form:

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

For example, the name of the functional feature “evaluating the condition of a print” contains description of the action “evaluate”, result “condition”, and object “print”.

In case if a result of the action cannot be evaluated, a name of the functional feature could be written down in the following form:

\[ <action>-ing a(n) <object> \]

For example, a name of the functional feature “servicing a reader” indicates at the action “service” and the object of this action “reader”.

Topological Structure

The topology is represented by cause-and-effect relations between functional features. Graphically, the cause-and-effect relations are represented as arcs between vertices of a directed graph, which are oriented from a cause vertex to an effect vertex. By far, the cause-and-effect relations between functional features can be recorded in the form of an adjacency (incident) matrix, which is a \( Z \) by \( Z \) matrix where \( Z \) is a number of functional features represented as nodes in the topological space. If there is an arc from some vertex \( x \) to some vertex
y, then the element $M_{x,y}$ is 1, otherwise it is 0. This makes it easier to find sub-graphs, it is vitally important for analyzing subsystems.

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. Since identification of cause-and-effect relations is rather intuitive work, the essence of cause-and-effect relations needs to be described.

The connection between a cause and an effect is represented by a certain conditional expression, the causal implication. It is characterized by the nature or business rules not by logic rules. They are such concepts as ontological necessity, probability etc. In causal connections “something is allowed to go wrong”, whereas logical statements allow no exceptions. Using this property of the cause-and-effect relations a logical sequence, wherein the execution of the precondition guarantees the execution of the action, can be omitted; this means that even if a cause is executed, it is allowed that the corresponding effect will not be generated because of some functional damage.

Cause-and-effect relations have a time dimension, since a cause chronologically precedes and generates an effect. The concept of generating is necessary in order to distinguish a cause-and-effect relation from the simple consequence that is not causal. Causes may be sufficient or necessary (in other words, complete or partial) for generating an effect (Farlex Inc., 2009). A sufficient (complete) cause generates its effect ever, in any conditions. On the other hand, a necessary cause (partial) only promotes its effect generating and this effect is realized only if this partial cause joins other conditions. However, most cause-and-effect relationships involve multiple factors. Sometimes there are factors in series. Sometimes there are factors in parallel. In case of the topological functioning model, it is assumed that a deal is always with necessary causes as the functionality of the system has its known and unknown risks at the time of analysis. And the last, the causality is universal. This means that there is no such a problem domain without causes and effects. The person can see nothing, but a cause or an effect exists.

As already mentioned, a structure of cause-and-effect relations can form a causal chain (in series or in parallel). The causal chain begins with an initial cause and follows with series of intermediate actions or events to a final effect. Though one link may not be as important or as strong like the other ones, they are all necessary to the chain. If just one of these intermediate causes is absent, then the final effect would not be reached. As an example, the situation of payment in restaurant can be considered. A causal chain of this case can be the following: ordering in a restaurant, then eating the order, making a decision about payment, paying the bill. If the decision about payment is not made, the last action (paying the bill) will not be generated. Additionally, even if you change something, you cannot remove the effect without removing or changing the cause.

Separation of a Topological Model of System Functioning

As stated in the beginning of this section, “Abstract topological model of system functioning”, a topological space of the system is represented by the expression (a) in Figure 3, where $Z$ is a set of system functional features, $N$ is a set of inner functional features of the system, and $M$ is a set of functional features of other systems, constituting the environment affecting the system or that is affected by the system. Inner functional features and external affecting functional features are stated by domain experts. However, there is a way that allows checking of experts’ statements. In common case, external affecting functional features should be inputs and outputs in the model. For example, in the topological space presented in Figure 3 (c) the set $Z = \{a, b, c, d, e, f, g\}$ that is a union of $M = \{a, d, f, g\}$ and $N = \{b, c, e\}$ as
Topological Modeling for Model-Driven Domain Analysis and Software Development

Figure 3. Separation of the topological model of abstract system functioning

(a) \( Z = M \cup N \)
(b) \( X = [N] = \bigcup_{i=1}^{k} X_i \)
(c) The set of system’s inner functional features
\( \mathcal{N} = \{b, c, e\} \)
The set of external system’s functional features
\( \mathcal{M} = \{a, d, f, g\} \)
The set of functional features in the topological space
\( Z = \{a, b, c, d, e, f, g\} \)
The set of functional features in the topological model
\( X = \{a, b, c, d, e, f\} \)

stated by the expression (a) in the same figure. Besides that this topological space also satisfies axioms in Figure 1.

Separation of a topological model of system functioning from the topological space, i.e., formal definition of the set \( X \), can be done by using the closure operation over the set \( \mathcal{N} \) as illustrated by the expression (b) in Figure 3, where \( X_i \) is an adherence point of the set \( \mathcal{N} \); and \( k \) is a number of adherence points of \( \mathcal{N} \), i.e. capacity of \( X \). In order to execute this operation, definitions of neighborhood and an adherence point need to be done.

Definition 1: The neighborhood of the vertex \( x \) in a graph is a set of all the vertices adjacent to \( x \) and the vertex \( x \) itself. It is assumed here that all the vertices adjacent to \( x \) lie at the one step distance from \( x \). This means that only direct successors of \( x \) are taken into accounts.

Definition 2: An adherence point of the set \( \mathcal{N} \) is a point, each neighborhood of which includes at least one point from the set \( \mathcal{N} \).

For example, in case of the topological space in Figure 3 (c), the neighborhood of the vertex \( e \) is a set of vertices \( e \) (the vertex itself), \( b, d, \) and \( f \). The vertex \( c \) is not included in this neighborhood, because it lies at a distance of one step back.

When we need to define the set \( X \), we should define neighborhoods for all vertices in the topological space and then to unite those neighborhoods, which contain at least one point that belongs to \( \mathcal{N} \). Taking as an example the topological space in Figure 3 (c), sets of neighborhoods are as follows:

- The neighborhood of the vertex \( a \) is a set \( X_a = \{a, b\} \);
- The neighborhood of the vertex \( b \) is a set \( X_b = \{b, c\} \);
- The neighborhood of the vertex \( c \) is a set \( X_c = \{c, e\} \);
- The neighborhood of the vertex \( d \) is a set \( X_d = \{d, g\} \);
- The neighborhood of the vertex \( e \) is a set \( X_e = \{e, b, d, f\} \);
- The neighborhood of the vertex \( f \) is a set \( X_f = \{f\} \);
- The neighborhood of the vertex \( g \) is a set \( X_g = \{g\} \).

Vertices names, which belong to the set \( \mathcal{N} \), are denoted with \textbf{bold}. Hence, we have four from seven neighborhoods, which contain vertices from the set \( \mathcal{N} \). They are neighborhoods of vertices \( a, b, c, \) and \( e \). Therefore, the set \( X \) is a union of neighborhoods of those vertices, or in other words a union of adherence points \( X_a, X_b, X_c, \) and \( X_e \). Note that the neighborhood of the vertex \( a \) also belongs to this union, although the vertex \( a \) itself belongs to the set \( \mathcal{M} \), not to the set \( \mathcal{N} \). The result of the union is the set \( X = \{a, b, c, d, e, f\} \). Functional feature \( g \) is out of the scope of this abstract system.

In case of studying complex systems it is essential to investigate systems division into series of...
subsystems. Selection of the topological subspace of the subsystem from the topological space of the system also is formulated as closing over the subset of own properties of the subsystem, thus this amounts taking into consideration the closures of subsets of $N$.

Practically, studying topological and functional properties of topological models of system functioning (discussed below) reduces to checking on those properties of the according topological digraph. The formalized statements given below provide the control of correctness in the process of model construction (Osis, 2004). Examples which illustrate topological and functional properties of topological models of system functioning are presented in the next subsection.

**Common Topological and Functional Properties of Systems**

A topological functioning model has topological and functional properties. According to (Osis, 1969), the topological properties are connectedness, neighborhoods, closure, and continuous mapping; and the functional properties are cause-and-effect relations, cycle structure, inputs and outputs. Statements described below formalize these model properties.

The topological and functional properties are illustrated by a few examples. One of them is a quite simplified description of the part of library’s work but allows us to demonstrate the represented theory. Let us assume that we have the following description:

“The library invites people to come. The Advertising Company gives the informational support for the library. When a reader comes, he is serviced by a librarian. Each month the librarian evaluates the condition of used prints. If a print is damaged, then it is either restored by the Restoration Company or removed by the Liquidating Company. The removed prints are liquidated by the Liquidating Company, while the library continues to use the restored prints. Library’s Fund Company gives the financial support that is based on annual library’s reports. The Library’s Fund Company itself is credited by its partners. Library’s fund gets this financial support. The library distribute the obtained income among paying salaries to employees, restoring prints, removing damaged prints and purchasing prints. Each three months the library purchases prints which are published by publishing houses in order to service their readers. Before each purchase, the library evaluates readers’ requirements as well the condition of library’s prints. The library gives the information support by fee.”

**Topological Properties**

**Connectedness**

The first topological property is *connectedness*.

**Statement 1:** A topological space, which represents functioning of the business or technical system, must be connected.

Statement 1 defines that the topological space of the functioning system must be connected. This means that the digraph cannot have isolated vertices. It defines also formal semantics of concepts of subsystems and independent systems.

- **Corollary 1-1:** The topological digraph $G(X, U)$ of functioning of the system cannot include any isolated vertices.
- **Corollary 1-2:** Every business and technical system is a subsystem of the environment.

As mentioned, functional features are determined from the verbal description of the system given above. Table 1 enumerates defined functional features of the topological space according with the unique tuple $<A, R, O, PrCond, PostCond, Pr, Ex>$ described hereinafter (an exception is the last column that describes to which system a functional feature belongs). They correspond to the graph vertex labels in Figure 4 (a). For example,
Topological Modeling for Model-Driven Domain Analysis and Software Development

Table 1. Functional features of the library

<table>
<thead>
<tr>
<th>Label</th>
<th>Name</th>
<th>Preconditions</th>
<th>Postconditions</th>
<th>Providers</th>
<th>Executors</th>
<th>Subordination</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>action-ing action-ing the result an object</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>Publishing</td>
<td>PrCnd</td>
<td>PostCond</td>
<td>Pr</td>
<td>Ex</td>
<td>N/M</td>
</tr>
<tr>
<td>a</td>
<td>Publishing a print</td>
<td></td>
<td>PH PH</td>
<td>PH</td>
<td>PH</td>
<td>M</td>
</tr>
<tr>
<td>b</td>
<td>Conming a reader</td>
<td>Lb</td>
<td>R</td>
<td>N</td>
<td></td>
<td></td>
</tr>
<tr>
<td>c</td>
<td>Purchasing a print</td>
<td>PH</td>
<td>Lb</td>
<td>N</td>
<td></td>
<td></td>
</tr>
<tr>
<td>d</td>
<td>Sericing a reader</td>
<td>income</td>
<td>Lb</td>
<td>Lb</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td>e</td>
<td>Evaluating the condition of a print each month damaged or undamaged condition</td>
<td>Lb</td>
<td>Lb</td>
<td>N</td>
<td></td>
<td></td>
</tr>
<tr>
<td>f</td>
<td>Evaluating the require-ments of a reader reader’s require-ments</td>
<td>Lb</td>
<td>Lb</td>
<td>N</td>
<td></td>
<td></td>
</tr>
<tr>
<td>g</td>
<td>Resoring a print</td>
<td>if a print is damaged</td>
<td>RC</td>
<td>Lb</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td>h</td>
<td>Distributing Income</td>
<td></td>
<td>Lb</td>
<td>Lb</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td>i</td>
<td>Rmving a damaged print</td>
<td>if a print is damaged</td>
<td>LC</td>
<td>Lb</td>
<td>M</td>
<td></td>
</tr>
<tr>
<td>j</td>
<td>Givng financial support</td>
<td>review of library’s annual report income</td>
<td>LF</td>
<td>LF</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td>k</td>
<td>Payng the salary to an employee</td>
<td></td>
<td>Lb</td>
<td>Lb</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td>l</td>
<td>Givng informational support</td>
<td></td>
<td>Lb</td>
<td>Lb</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td>m</td>
<td>Creting a report</td>
<td>deadline of annual report submission</td>
<td>Lb</td>
<td>Lb</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td>n</td>
<td>Liquidating a print</td>
<td>if a print is removed</td>
<td>LC</td>
<td>LC</td>
<td>M</td>
<td></td>
</tr>
<tr>
<td>o</td>
<td>Invting a man</td>
<td></td>
<td>AC</td>
<td>AC</td>
<td>M</td>
<td></td>
</tr>
<tr>
<td>p</td>
<td>Creting library’s fund company</td>
<td></td>
<td>Cr</td>
<td>FC</td>
<td>M</td>
<td></td>
</tr>
</tbody>
</table>

the vertex c graphically denotes the functional feature “c: Purchasing a print”. Letters used for denoting providers and executors specifies the following entities: Lb - the library, Cr - a creditor, R - a reader, LF – Library’s Fund Company, AC – an advertising company, RC –a company that restores prints, LC – a company that liquiates prints, and PH – a publishing house. In the last column, N denotes inner (system) functional features, and M denotes functional features of the external environment.

Neighborhoods

According to expression (b) in Figure 3, we should identify neighborhoods of functional features of N. According to Table 1 the set $N = \{b, c, d, e, f, g, h, j, k, l, m\}$, and the set $M = \{a, i, o, n, p\}$. The set $X$ of system properties is obtained by uniting adherence points of $N$. The list of neighborhoods is as follows:

$X_a = \{a, c\}$,  
$X_b = \{b, d\}$,  
$X_c = \{c, d\}$,
Topological Modeling for Model-Driven Domain Analysis and Software Development

\[ X_d = \{d, f, h, c\}, \]
\[ X_e = \{e, g, i\}, \]
\[ X_f = \{f, c\}, \]
\[ X_g = \{g, d\}, \]
\[ X_h = \{h, c, g, i, l, m\}, \]
\[ X_i = \{i, n\}, \]
\[ X_j = \{j, h\}, \]
\[ X_k = \{k, d\}, \]
\[ X_l = \{l, b\}, \]
\[ X_m = \{m, j\}, \]
\[ X_n = \{n\}, \]
\[ X_o = \{o, b\}, \]
\[ X_p = \{p, j\}. \]

Labels of vertices denoted by bold are labels of functional features which belong to the set \( N \).

Closure

In order to get the set \( X \) we should perform closure over set \( N \), i.e., we should unite neighborhoods \( X'_s \), \( X'_d \), \( X'_e \), \( X'_f \), \( X'_g \), \( X'_h \), \( X'_i \), \( X'_j \), \( X'_k \), \( X'_l \), \( X'_m \), and \( X'_n \). Thus, the set of system functional features is \( X = \{a, b, c, d, e, f, g, h, i, j, k, l, m, o, p\} \). It includes also functionality from the set \( M \), i.e., vertices \( a, i, o \) and \( p \). The vertex \( n \) is out of the system’s boundary. In such a way, the system boundary is formally (mathematically) defined.

Figure 4 (a) illustrates a topological model of library functioning that is constructed from the description given above. Elements that are in the external environment, for example “n: Liquidating a print” (the vertex \( n \)) or “p: Crediting a fund company” (the vertex \( p \)), do not belong to the inner system characteristics; however, they must

---

Figure 4. The topological model of library functioning

---

a) The valid topological model of the library
b) The invalid topological model of the library
c) The topological model of library’s subsystem functioning
d) The refined topological model of the library
Topological Modeling for Model-Driven Domain Analysis and Software Development

Table 2. The TFM represented as the incident matrix X

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
<th>e</th>
<th>f</th>
<th>g</th>
<th>h</th>
<th>i</th>
<th>j</th>
<th>k</th>
<th>l</th>
<th>m</th>
<th>o</th>
<th>p</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>b</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>c</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>d</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>e</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>f</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>g</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>h</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>i</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>j</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>k</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>l</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>m</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>o</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>p</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

be represented in the topological functioning model since the system must communicate with other systems in order to perform its functions. Those vertices in Figure 4 which belong to M are shadowed.

Figure 4 (b) demonstrates the invalid topological model. First, because it contains three isolated vertices, namely, a, o and p as well as the vertex n that is out of the system scope. Second, the topological model is invalid due to a lack of the cause-and-effect relation from “k: Paying salaries to employees” to “Servicing a reader”. In the environment, the real world, employees work for money, and if this relation is absent, then this means that the business system, i.e., the library, is invalid or incompliant subsystem of the environment.

Functional Properties

Cause-and-Effect Relations
Table 2 shows the topology as an incident matrix $X_{(a,b,c,d,...,p)}$, where 1 denotes that a cause-and-effect relation between functional features exist. We believe that representation by matrix is not convenient. First, cycles are not so evident then it is in graphical or list forms. Identification of cycles requires additional analysis of the matrix.

Second, in case of large sizes of the TFM, the large number of null elements in matrices impractically uses computer resources.

Cycle Structure
In point of fact of functioning the common thing for all systems (technical, business and biological) should be an oriented cycle (a directed closed path). This similarity of technical and biological systems was recognized, because the main cycle in essence represents a shape of the main feedback. Therefore, in every topological model of system functioning there must be at least one directed closed loop (i.e. a directed closed path). Usually it is even expanded hierarchy of cycles. As stated in (Osis, 1969), this property of the model enables analyzing similarities and differences (that is not less important than similarity analysis) of functioning systems.
In Figure 4 (a) the model reflects the following cycles. The main cycle of the system model is c-d-h-c, because the system will not be able to function if it does not offer services to clients, or it has no new prints (books and magazines become unreadable as time goes by), or it has no money. The system can function without other characteristics but, perhaps, not so efficiently. This cycle represents the main feedback. The first order subcycle is a cycle that has at least one vertex shared with the main cycle. The second order subcycle is a cycle that has at least one vertex shared with the first order subcycle, and so on. Cycles can form hierarchies.

The first order subcycles are as follows:

- The subcycle c-d-f-c represents the way the system analyzes its work taking into consideration clients’ desires.
- The subcycle h-k-d-h represents a subsystem of money relationships among the organization and its employees.
- The subcycle h-m-j-h represents system relationships with financial support organizations. The library needs to report information about its work in order to get some financial support from these organizations.
- The subcycle d-e-g-d represents functionality that is necessary to perform when the used print condition requires restoration.
- The subcycle d-h-g-d represents the financial support necessity when a used print needs restoration.
- The subcycle d-h-l-b-d describes activities that let readers know any necessary information.

For comparison, the list of incident tuples for the same topological model in Figure 4 (a) is as follows:

\(<a, c>, [c, d], [d, h], [h, c]>, (c, e), (d, f), (h, c), (d, h, g), (h, c), (d, h, i), (l, b, d), (o, b, p, j), (j, h, m, j), (e, i, h, i), (i, n)\).

Where:

- squared brackets “[sublist]” denote a sublist of arcs, which form the main functioning cycle, e.g. the cycle [c, d], [d, h], [h, c];
- simple brackets “(sublist)” denote a sublist of arcs, which form a subcycle, e.g. (c, d, f), (h, c);
- “label” denotes functional features that belong to the main functioning cycle, e.g. vertex c;
- “label1 label2” denotes an arc of the higher rang that is presented in the given subcycle, e.g. arc cd.

This clear representation allows tracing back a hierarchy of the cycle structure also if the TFM description is given in the list form.

**Inputs and Outputs**

The topological model must have input vertices as well as output vertices. This enables to identify what functionality from the external environment and inside the system generates other functional properties of the system. For example, the constructed topological model has the input vertices \(\{a, p, o\}\) and the output vertex \(i\). Formal definition of the TFM by using the closing operation over a set is considered hereinafter.

Input and output functional features can be identified by analyzing values in rows and columns. Namely, input functional features of the TFM have all nulls in columns, and output functional features of the TFM have all nulls in rows (see Table 2 row \(i\), and columns \(a, o,\) and \(p\)).

**Statement 2.** If a vertex \(\alpha\) of the neighborhood \(X_\alpha = \{a, b\}\) belongs to the set \(M\) and the neighbor-
hood $X_\alpha$ contains a vertex $\beta$ of the set $N$, then the vertex $\alpha$ is a TFM input vertex.

For example, the neighborhood of vertex $p$ is $X_p = \{p, j\}$, hence $p$ is an input vertex. For input vertex columns in the incident matrix $X$ all values are nulls (see Table 2).

**Statement 3.** If a vertex $w$ of the set $M$ belongs to a neighborhood of some vertex of the set $N$, then the vertex $w$ is a TFM output vertex.

For example, vertex $i$ belongs to $M$ and the neighborhoods of vertices $h$ and $e$ of the set $N$, namely, $X_e = \{e, g, i\}$ and $X_h = \{h, c, k, g, i, l, m\}$. Thus, vertex $i$ is an output vertex.

**Corollary 2/3-1.** If a vertex $\gamma$ that belongs to the set $M$ can be found only in those neighborhoods, all vertices of which belong to the set $M$, then $\gamma$ does not belong to the set $X$ and is not reflected in the graph representing the TFM.

For example, the vertex $n$ can be found only in the neighborhoods $X_i = \{i, n\}$ and $X_n = \{n\}$, where all vertices (here: $i$ and $n$) in these neighborhoods belong to $M$. Therefore, $n$ is not depicted in the topological models in Figure 4 (a) and (d).

**Decomposition in Subsystems**

According to Corollary 1-2, there is no difference between a system and a subsystem. Every subsystem is a system regarding to its environment. Therefore, in case of subsystems, the environment is a system that in turn is separated from the larger environment.

If we would like to separate a topological model of the subsystem, we should do the same activities. For example, if we need to create a subsystem that provides qualitative services for readers. Let us assume that an expert has defined that functionality described by functional features $d$ and $e$ should be included in this subsystem. Thus, the set of system inner functional features is $N_{\text{subsystem}} = \{d, e\}$. The set of functional features $M_{\text{subsystem}}$ is the set difference of $X$ and $N_{\text{subsystem}}$, i.e., $M_{\text{subsystem}} = X \setminus N_{\text{subsystem}} = \{a, b, c, f, g, h, i, j, k, l, m, o, p\}$. By using the closure over $N_{\text{subsystem}}$, we get the following neighborhoods:

- $X_a = \{a, c\}$
- $X_b = \{b, d\}$
- $X_c = \{c, d\}$
- $X_d = \{d, f, h, e\}$
- $X_e = \{e, g, i\}$
- $X_f = \{f, c\}$
- $X_g = \{g, d\}$
- $X_h = \{h, c, k, g, i, l, m\}$
- $X_i = \{i\}$
- $X_j = \{j, h\}$
- $X_k = \{k, d\}$
- $X_l = \{l, b\}$
- $X_m = \{m, j\}$
- $X_o = \{o, b\}$, and
- $X_p = \{p, j\}$.

Labels of vertices denoted by bold are labels of functional features which belong to the set $N_{\text{subsystem}}$. In order to get the set $X_{\text{subsystem}}$ we should unite neighborhoods $X_b$, $X_c$, $X_d$, $X_e$, $X_g$, and $X_k$. The set of subsystem functional features is $X_{\text{subsystem}} = \{b, c, d, e, f, g, h, i, k\}$, and the topological model of this subsystem with shadowed vertices, which denotes external functional features, is illustrated in Figure 4 (c). This model is valid, because it has not isolated vertices, and has its main functioning cycle $d-e-g-d$ that represents functionality that is necessary to perform when the used print condition requires restoration.

According to Statement 2, input functional features of the TFM of the subsystem are $b$, $c$, $g$, $h$, $k$. The reason is neighborhoods $X_b = \{b, d\}$, $X_c = \{c, d\}$, $X_g = \{g, d\}$, and $X_k = \{k, d\}$. According to Statement 3, output functional features of the TFM of the subsystem are $f$, $h$, $e$, and $i$. The reason is neighborhoods $X_f = \{d, f, h, e\}$, and $X_i = \{e, g, i\}$. Functional feature $g$ belongs as to inputs as to outputs, this means that it is an inner functional
There is a feature that was not recognized by the experts before and it should belong to the set \( N \). Hence, inputs are \( b, c, \) and \( k \), and outputs are \( f, h \) and \( i \).

The model shows that experts’ definition of subsystem functionality (functional features “\( d: \) servicing a reader” and “\( e: \) evaluating the condition of a print”) was incomplete, because they did not include the functional feature “\( g: \) Restoring a print”. However, this is one of the essential functions of the subsystem, because the bad condition of prints can be a cause of decreasing a count of readers as well as a cause of unnecessary costs for purchasing new copies of prints. Fortunately, the topological model in graph representation shows such particularities evidently in comparison with matrices or lists. The essential nature of the functional feature “\( g: \) Restoring a print” is evident from its belonging to the first-order functioning cycle \( d-e-g-d \) that is a main functioning cycle for this subsystem (if we look at this subsystem independently from the library). The subsystem cannot operate without at least one functioning cycle. Moreover, experts, matrices and lists cannot formally point to this incompleteness, only the TFM enables discovering this important feature of the subsystem-to-be during formal separation of the subsystem-to-be from its environment.

Besides that, the topological model enables us to see that such a subsystem will have multiple communication points with other subsystems, since this model has multiple inputs and outputs. Figure 4(c) illustrates that in order to service readers qualitatively the subsystem should purchase new prints or new copies of liquidated prints (functional feature “\( c: \) purchasing a print”), extend or at least not decrease a number of readers (functional feature “\( b: \) coming a reader”), satisfy employees’ desires (functional feature “\( k: \) paying the salary to an employee”). Besides that the subsystem should provide information to the external environment about the condition of used prints (functional feature “\( f: \) evaluating the condition of a print”), income got by servicing readers (functional feature “\( h: \) distributing income”), and damaged prints (functional feature “\( i: \) removing a damaged print”). These cause-and-effect relations from input to inner functional features and from inner to output functional features illustrates related functionality that extend the initial set of subsystem functions.

Certainly, subsystems should have a minimum of communication points with other systems. And our assumed subsystem is not good from the viewpoint of creating components (modules), but at the same time this example is good for illustrating the principle of creation of subsystems and discovering its incompleteness. We should note, again, that definition of subsystems is rather more evident in topological digraph representation than in matrices or lists. Graphical representation allows seeing possible subsystems straighter than matrices and lists do.

Continuous Mapping

Continuous mapping is the last topological property that can be presented only now – after overview of all functional properties of the TFM.

A functional feature of the system can be considered as a more detailed subset of specialized features. The model with any complexity can be abstracted to the simpler one and vice versa. This means that continuous mapping of topological models is realized. Continuous mapping states that direction of topological model arcs must be kept as in a refined as in a simplified model. Moreover, a lack of knowledge about the system can be filled up with knowledge that is obtained from the continuous mapping of the same type model to the system model under consideration. This property may be very useful for development of product lines. Because core functions of the systems in product lines are the same, optional functionality may be explicitly represented in corresponding TFMs.

- Statement 2: If some more detailed functioning system is formed by substitution of a subset of specialized properties for some
Topological Modeling for Model-Driven Domain Analysis and Software Development

functional property, then continuous mapping exists between a detailed model and a simplified parent topological model of the same system.

- **Proof:** The continuous mapping of the topological space of the detailed system \( T^* \) into the topological space of the simple parent system \( T \) will take place, if neighbourhoods of \( T^* \) will map into neighbourhoods of \( T \).

Let us assume that the contrary is the case, i.e., that neighbourhoods of \( T^* \) are not mapped into neighborhoods of \( T \). This means that \( T^* \) possesses other essential topological properties. Therefore, either the mode of functioning of the detailed system will be different, or the detailed system will cease to exist at all. It follows that new essential functional features are being added but it contradicts to the premise of the statement. It’s easy to prove also the converse statement.

- **Corollary 2-1:** In the topological digraph \( G^* (X^*, U^*) \), the direction of arcs, which join the specialized point subset nodes with other nodes, is determined by the direction of the arcs, which join the replaced point with the corresponding nodes of the digraph \( G (X, U) \).

- **Corollary 2-2:** A data lack that sometimes arises during the composing of a topological model of functioning can be filled up by data that are obtained, when models of the same type systems are being continuously mapped to the model of the system under study.

For example, let us consider the topological model in Figure 4 (a). This model can be continuously mapped to a more detail model, e.g., to such a model like in Figure 4 (d), where the vertex “c: purchasing a print” is specialized to two vertices “c1: purchasing a book” and “c2: purchasing a magazine”. In order to prove the continuous mapping between models in Figure 4 (a) and (d), neighbourhoods of the detailed system in Figure 4 (d) must map into neighborhoods of the system in Figure 4 (a). Considering list of neighborhoods of those two models, the mapping between all neighborhoods is one-to-one except the following ones:

- \( X^*_a = \{a, c_1, c_2\} \) is mapped into \( X_a = \{a, c\} \), as \( c_1 \) belongs to \( c \), and \( c_2 \) belongs to \( c \);
- \( X^*_{c_1} = \{c_1, d\} \) is mapped into \( X_{c} = \{c, d\} \), as \( c_1 \) belongs to \( c \);
- \( X^*_{c_2} = \{c_2, d\} \) is mapped into \( X_{c} = \{c, d\} \), as \( c_2 \) belongs to \( c \);
- \( X^*_f = \{f, c_1, c_2\} \) is mapped into \( X_f = \{f, c\} \), as \( c_1 \) belongs to \( c \), and \( c_2 \) belongs to \( c \);
- \( X^*_h = \{h, c_1, c_2, k, g, i, l, m\} \) is mapped into \( X_h = \{h, c, k, g, i, l, m\} \), as \( c_1 \) belongs to \( c \), and \( c_2 \) belongs to \( c \).

But also these neighborhoods of the detailed system (denoted by “*”) are mapped into the neighborhoods of the system. Thus, the topological functioning model of the detailed system is continuously mapped into the topological functioning model of the system. The continuous mapping between models means that if something changes in one model, the same changes must be taken into account in the other one. Moreover, all arcs that belong to refined vertices have the same direction as before.

The topological model in Figure 4 (d) illustrates how the main functioning cycle \( c-d-h-c \) is divided into two main cycles – \( c_1-d-h-c_1 \) and \( c_2-d-h-c_2 \). These are two duplicating cycles. The library can operate if it provides continuously replenished collections of magazines, books or both.

By the way, continuous mapping serves as a very useful mechanism for handling changes and discovering similarities and differences between systems.
Similarities and Differences between Systems

Speaking about analysis of similarities and differences between functionality of systems, let us consider the topological model of functioning represented in Figure 5, the meaning of its vertices is given in Table 3.

The simplified topological model of functioning shown in Figure 5 represents the basic functional features of a technical system, diesel engine, and a live organism. This model seems to be very much simplified; nevertheless it captures the most important feature of all functioning systems (technical, biological, social, etc.). All functioning systems can be characterized by their cycle structure. Every functioning is possible only if there is at least one directed cycle in the topological model (a closed path in the graph). In case of the digraph shown in Figure 5, there are three cycles in it: $a-b-c-d-f-g-a$, $c-d-f-c$, and $b-c-d-f-b$. We shall define the cycle $a-b-c-d-f-g-a$ as a main one, in contrast to other two, which are first order sub cycles and are main cycles for subsystems.

The analogy between live systems and technical systems was noted earlier on the level of control. The feedback circuits in control systems and live organisms are similar. However, topological models permit to consider otherwise the cybernetic resemblance of functioning systems. Similarity of systems can be studied in details up to level, which is determinate beforehand. Let us notice in particular that both resemblance and difference between live organisms and technical devices can be studied with the help of topological models.

**MODEL-DRIVEN SOFTWARE DEVELOPMENT WITH TOPOLOGICAL MODELING**

Organizations are complex man-made systems that deal with many concepts such as people, business processes and information systems, technology, etc. Understanding how these complex systems operate or function is very important in order to introduce software solution into organization’s framework.

According to (IEEE, 2000) architecture is the fundamental organization of a system embodied in its components, their relationships to each other and to the environment and the principles guiding its design and evolution. The fundamental

---

**Table 3. Basic functional features of systems**

<table>
<thead>
<tr>
<th>Label</th>
<th>Diesel engine</th>
<th>Live organism</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>Contacting the diesel fuel in a tank</td>
<td>Contacting food</td>
</tr>
<tr>
<td>b</td>
<td>Taking the fuel in by a pump</td>
<td>Swallowing food</td>
</tr>
<tr>
<td>c</td>
<td>Burning up fuel</td>
<td>Digesting food</td>
</tr>
<tr>
<td>d</td>
<td>Transferring energy to crankshaft</td>
<td>Absorbing food by cells</td>
</tr>
<tr>
<td>e</td>
<td>Flowing used up gases out</td>
<td>Throwing waste out</td>
</tr>
<tr>
<td>f</td>
<td>Distributing energy</td>
<td>Distributing energy</td>
</tr>
<tr>
<td>g</td>
<td>Moving to a filling station</td>
<td>Moving to food</td>
</tr>
<tr>
<td>h</td>
<td>Supplying the fuel from a tank</td>
<td>Procuring food from surroundings</td>
</tr>
</tbody>
</table>
organization means essential, unifying concepts and principles of the system. The term “system” depicts applications, systems, platforms, systems of systems, enterprises, product lines, etc. And the environment depends on the context of the system – it can be developmental, operational, programmatic, and so on. Architecture is always applied to the certain universe of discourse, where a problem must be solved. Views on the system are parts of an architectural description of the system, i.e., products which document the architecture. An architectural view is a representation of a whole system from the perspective of a set of concerns (IEEE, 2000).

The topological functioning model is an engineering model with simple notation but solid foundations and modeling capabilities. It combines functional and structural views on the system within one formal holistic representation of the system. How these views can be used in software development is discussed hereinafter.

The initial stages of traditional software development are requirements gathering and system analysis that sometimes includes also business modeling. The objective of these initial stages is to find the solution which solves the stated problem and complies with client’s desires, i.e. understanding of what software must do in order to satisfy client’s needs.

Requirements are to be specified as complete as possible, describing external behavior of software to be built. Therefore after determining what functionality of the system in the real world exists, the next step is to determine those functional parts which will benefit from automation. This means that the real world system - the system “as is” is additionally constrained by client’s requirements for the software. These requirements will change the real world system, if are implemented and introduced, thus will modify the system “as is” to the system “to be”, where the software is a part of the system “to be” (Figure 6).

As mentioned in the previous sections, the TFM is an engineering model that can be successfully used for domain modeling and analysis based on formal mathematical constructions and mechanisms. And this self-sufficient engineering model of the domain (“as is” and “to be”) is a solid ground for further development of software.

The suggested process that relates domain analysis with the TFM and further construction of software models used in software development is illustrated in Figure 7. It starts from modeling and analyzing the problem domain with the TFM (see Figure 7-1a). At the high level of abstraction the TFM can be used as a simplified model for software developers – business and system analytics, but it has one significant advantage – it is formal and has mathematical foundations and thus may be formally refined to a more complex model (or models) of subsystems.

The TFM of the system “as is” is transformed into the system “to be” by using mappings from functional requirements as in Figure 7(2). As
Topological Modeling for Model-Driven Domain Analysis and Software Development

*Figure 7. Software development started with topological modeling of systems*

mentioned in (Osis, Asnina, & Grave, 2008), mappings describe the case of new non-existing functionality (one-to-zero), functionality that will not be implemented in the solution (zero-to-one), and functionality that exists and will be implemented in the solution (one-to-one, one-to-many, and many-to-one).

**One to One** is used if requirement $A$ completely specifies what will be implemented in accordance with functional feature $B$.

**Many to One** is used if requirements $(A_1, A_2, ..., A_n)$ overlap the specification of what will be implemented in accordance with functional feature $B$. In case of the covering requirements, their specification should be refined. Another case is a *disjoint (component) relation* that is used if requirements $(A_1, A_2, ..., A_n)$ together completely specify functional feature $B$ and do not overlap each other.

**One to Many** is used if a part of requirement $A$ completely specifies some functional feature $B$. This means that requirement $A$ completely specifies a set of functional features $(B_1, ..., B_n)$. For this purpose a relation called *separating family of functions* is used. This case can occur if: a) the requirement joins several requirements and can be split up or b) functional features are more detailed than the requirement.

**One to Zero** is used when one requirement specifies some new, invalid (impossible) or undefined functionality. New functionality requires definition of possible changes in the problem domain functioning. Undefined functionality forces more careful investigation of the problem and solution domains. In turn, invalid or impossible functionality requires identification of additional constraints in the problem domain. The discovered constrains can cause modifying or even removing those requirements.

**Zero to One** is applied if specification does not contain any requirement corresponding to the defined feature. This means that it could be a missed requirement, and therefore it could be left unimplemented in the application. Thus, it is mandatory to take a decision about the implementation of the discovered functionality together with the client.

The result of this activity is both checked requirements and TFM, which describes a needed and possible functionality of the system and the
environment it operates within. The mappings modify the TFM by introducing new functions and changing the existing one, thus composing the TFM of the system “to be” as in Figure 7(3). This composed TFM will illustrates both functions that will be implemented in the solution and functions that will not be implemented in the solution but still will exist in the problem domain (manual activities).

After mapping, the TFM of the system “to be” can be transformed into the behavioral and structural models of the solution as in Figure 7 (4 and 5). These behavioral and structural models are viewpoints on the system which describe particular concerns about the role of the system and may belong to different existing architectures.

For example, wide and long known John Zachman’s framework for information systems architecture (Sowa & Zachman, 1992) is a set of architectures structured according with perspectives of development. According with this framework, the topological modeling of the system is allocated in context and business model perspectives of the first dimension and answers the following questions of the second dimension: data (“what?”), function (“how?”), network (“where?”), people (“who?”), time (“when?”), and motivation (“why?”). The data perspective is shown as a list of objects, subsystems and external systems that are used in system functioning. The function perspective is shown as functional features and cause-and-effect relations among them. The network perspective is shown as a place where each functional feature is allocated, i.e., within the system or in the external environment. The people perspective is shown by an entity that provides each functional feature. The time perspective is shown as cause and effect sequences, because the effect may occur only if a cause generated it. The motivation perspective is shown by functioning cycles, inputs and outputs in the topological functioning model. Therefore, it holds all the necessary information from the business model perspective for construction of models according to the more detailed perspective, namely, the information system model from the designer’s viewpoint.

Let us take for another example the 4+1 View Model of Architecture suggested by Philippe Kruchten in1995 (Kruchten, 1995). The View Model describes software architecture by five related views. One of these views, a scenario view, is related with other four views, namely, logical, process, development and physical. The topological functioning model is a holistic representation of all possible scenarios of the system operation. It is a base for constructing the logical view that specifies design object model, the process view that specifies time and synchronization aspects, and also the development view that specifies the system by modules and subsystems and communications among them.

Therefore, these solution models will be in conformity with the problem domain, and client’s requirements will be thoroughly checked and inspected. Some of those aspects are highlighted and discussed in more detail in other chapters of the book.

The Metamodel of the TFM for System Modeling

The main purpose of metamodeling is definition of modeling languages. Besides that, metamodeling is also used for defining domain specific languages and for transforming artifacts among various languages. Thus metamodeling supports conformance among tools and languages. The metamodeling language used for this purpose is a Meta-Object Facility (MOF) standard developed by the Object Management Group (2006). The MOF is implemented within several commercial and open-source tools.

The topological model of system functioning has a concrete and abstract syntax and construction rules (which we can call also as well-formedness rules). Thus, the TFM is a modeling language.
The standalone metamodel of the TFM that defines abstract syntax and construction constraints is represented in Figure 8. The metamodel is described at the MOF metalevel M2 (the level of metamodels), and represents the topological functioning model as an instance of the metaclass \textit{TopologicalFunctioningModel} that includes at least two functional features of the metaclass \textit{FunctionalFeature}.

Instances of functional features \textit{FunctionalFeature} can be joined in functional feature sets (the metaclass \textit{FunctionalFeatureSet}). This means that a functional feature represented in a TFM can visualize a functional feature set. One functional feature can contain only one set and one functional feature can belong only to the one set. A functional feature can be subordinated to a system itself or to an external system (enumeration \textit{Subordination}). Each functional feature contains an instance of the object (the metaclass \textit{FunctionalFeatureObject}), an instance of the result (the metaclass \textit{FunctionalFeatureResult}), and an instance of an action (the metaclass \textit{FunctionalFeatureAction}).

The combination of these three instances together with preconditions (the metaclass \textit{Precondition}), postconditions (the metaclass \textit{Postcondition}) and instances of responsible entities (the metaclass \textit{ResponsibleEntity}) associated with the functional feature must be unique within the model instance.

Functional features should form functioning cycles (the metaclass \textit{Cycle}) of different order. Only one cycle can be the main one. Cause-and-effect relations connect functional features. A cause functional feature must have at least one effect. An effect functional feature must have at least one cause.

Functional features are related to functional requirements (the metaclass \textit{FunctionalRequirement}) via correspondence (the metaclass \textit{Correspondence}). The correspondence generally is many-to-many. The correspondence can be complete or incomplete, overlapping or disjoint. The corresponding functional requirements are associated with the functioning cycle, whose order establishes their benefit value (the type \textit{Benefit}).
FUTURE RESEARCH DIRECTIONS

As mentioned in Section “Background”, application of the topological model of system functioning is quite wide. At the present we research application of topological modeling for development of information systems within model-driven development paradigm. The initial results are presented in this book in the following chapters. Namely, application of the topological model of system functioning as a holistic business model at the high level of abstraction, formal determination of use cases grounded on the TFM, and formal determination of a class diagram at the conceptual level from the TFM.

Future research directions are related to developing a tool that supports construction of the TFM and its transformations to different architectural views (or dimensions) on the system, and refinement of a technique of TFM application in practice.

CONCLUSION

In this chapter we have discussed mathematical foundations of the topological model of system functioning and formalism of the modeling approach based on this model, and have explained applications of this model with examples. The topological model of systems functioning is an engineering model, i.e., it supports understandability, accuracy, predictiveness and inexpensiveness. Comparing with Petri nets, the topological functioning model can be used as a self-sufficient model for modeling functionality of businesses, mechanical systems and even living organisms. The main advantage of the topological model is its holistic representation of the system that allows analyzing the system as a whole, not by fragmental presentation of system’s parts. Analysis of cause-and-effect relations allows identifying missing or weakly defined functionality as well identification of subsystems could be more evident, because the TFM reflects all communication points of assumed subsystems. Therefore, we can start analysis and development of systems or subsystems seeing their operation in the whole together with their interactions with the external environment.

REFERENCES


KEY TERMS AND DEFINITIONS

**Cause-and-Effect Relation:** A relation between two functional features of the system where appearance and execution of one functional feature is caused (generated) by the other functional feature without any intermediate functionality.

**Continuous Mapping:** A mathematical relation between two topological directed graphs that preserve their functional and structural properties.

**Cycle Structure:** A closed directed path that is formed by cause-and-effect relations between TFM functional features. Cycle structure describes functioning cycles of the system. The system must have at least one functioning cycle. Cycle structures may form hierarchies.

**Functioning Cycle:** A closed path of cause-and-effect relations among functional features in the topological functioning model. A main cycle joins functional features which are vitally important for the system operation. Functioning cycles may form hierarchies.
**Metamodel:** A model that specifies abstract and concrete syntax of the modeling language as well as well-formedness rules. A metamodel is specified in terms of a reflexive metametamodelling language, e.g. OMG’s Meta-Object Facility.

**Model of System Architecture:** A set of viewpoints on behavioral and structural characteristics of the system under consideration from different dimensions and perspectives.

**System’s Function:** The natural purpose of the system or its part.

**System’s Functional Feature:** A functional characteristic of the system that is needed to reach system’s functioning goal.

**Topological Functioning Model:** A model based on system theory and mathematics that specifies functionality and structure of a complex system as a connected topological space of system’s functional characteristics and cause-and-effect relations among them.