# DEVELOPMENT OF INFORMATIONS SYSTEM USING METHOD OF GRADUAL TRANSFORMATIONS

**M. Pícka**

*Czech University of Life Sciences, Faculty of Economics and Management, Department of Information Engineering, Prague, Czech Republic*

The objective of this article is to show a new way of depicting of design method model. The model is gradually created by adding new elements to the existing model. At the beginning there are only input elements in the model. A new element is created by a transformation of existing elements in the model. Admissible transformations are defined by methodology. We are able to produce a diagram that captures progress of the information system development. This helps us keep the model consistent, well documented and helps us find affected elements when the model is changed. With these principles we are able to construct a model of design method. This model is created by concepts and admissible transformations between them. These principles are applicable to every design method. Examples are shown on BORM and UP methodology.

BORM methodology; Unified Process; model; transformation; software engineering; method's model

## INTRODUCTION

Information system (IS) development can be carried out by a variety of methods. The methods provide different constructs for working with real-world terms. The goal of analysis is to record a certain part of reality and depict it in an appropriate model as accurately as possible. This model (or a set of models) becomes the foundation of the IS design. In the end the design then leads to implementation. We are focused to different aspects of the future system during individual phases of development.

The creator of an IS generally works by sequentially adding new elements to the model. However, analytical and design methods usually used cannot record the relationships between the elements being added and the model created so-far. Methods usually do not explicitly contain relationships among their terms, either.

Nowadays, information systems are usually designed using methodologies that do not help maintain relationships among the individual gradually added elements of the model much. Big mental jumps among loosely bound elements (documents, diagrams) are usual in methodologies – e.g. in methodologies based upon UML there exist huge gaps between the use-case diagrams, the activity diagrams, sequences and classes. This forces the analyst to fill those gaps in his mind, which increases the demands on analyst's experience of the modelled branch on one hand. On the other hand, those transitions are undocumented, because methodologies do not provide ways how to record them – we cannot say why and how a certain element got into the model. That leads to a consistency loss among those elements. Model typically contains elements that are either useless or even false. A solution may be to construct an IS in a sequence of small steps that follow each other, such that the analyst does not lose the context.

The goal of this article is to show a new way of correctness and consistency assurance during IS design using successive transformation of elements in the model from entry elements, created according to the task, to elements making the appropriate IS model. The article also shows a model of methodology, that illustrates time dependencies between elements of methodology.

## DEFINITIONS OF TERMS

For better understanding of the following text, we will define new or not usually used terms that we will use:

*Concept* – is an entity with which we work in the method (or methodology). Examples of concepts are: class, package, use-case, function, scenario, state, activity, etc. Concepts are contained in model's metamodel.

*Transition between the concepts* – it is a possible transformation of (several) concepts to new concepts, which is allowed in the method. Those transitions define links in the method.

*Model of admissible transitions in the method* (or shortly the model of the method) – is a model depicting all concepts of the method and mutual transitions allowed by the method. This model is expressed by the Concept Transition Diagram.

*Element is an instance of concept* – is a representation of concrete, further indivisible parts of the IS model. Elements are stored in a repository of the model. Examples of elements include a concrete class, a method, a function, a scenario, etc. A new element of the model is created by a transformation of existing elements in the model. Specific elements are so-called input elements that are introduced to the model from the outside, typically as a consequence of an analysis of the task and requirements.

*Transformation between elements* – is a process that generates new elements. It is an instance of a concept transition. The transformation between elements is a process in which new elements in the model are created from the existing ones. The transformation between elements is specified by its appropriate transition. All transformations performed are logged in the repository of the model.

*Element transformation log* – is a layer of the model which depicts all transformations performed in the model. This log records the "pedigree" of all elements in the model (i.e. relations of predecessor-successor type in the model). It is expressed by an appropriate diagram.

## SUCCESSIVE CONSTRUCTION OF INFORMATION SYSTEM

Successive modelling of information system in small steps (for context and relevance assurance) can be seen as successive adding of new elements to the existing model. For correctness assurance we propose to abide the following rules.

- Every new element added to the model of the information system must have sense.
- Every element, except of the entry ones, is created by one transformation.
- A transformation transforms several (one or more) elements to one or more successor elements.
- So-called entry elements exist in the model. They have no predecessor in the model and were created directly from the specification. It is necessary to pay big attention to their correctness, because the whole IS design that follows depends on them (when the foundation is weak, the whole building is such).

If those rules are followed, a new layer of model is constructed along with the model. The layer will show which elements originated from which elements and will record transformations among them (the pedigree of all elements in the model will be available). If the origins of all elements are recorded, a powerful tool for relevance checking is obtained.

### Characteristics of the Element Transformations Log

The layer of model that logs the transformations according to those rules has the following characteristics:

- Every new element must be created by a relevant (for given moment and given elements) transformation from the elements already present in the model (predecessor-successor relation).
- Transformation is allowed if all input elements exist.
- The IS model forms a directed acyclic graph where the nodes are the elements and the edges depict the transformations (after a small and distinct transformation).
- There is a partial ordering among the elements.

These characteristics imply the range of possible application of this approach, e.g. for checking or defining all elements, that are affected by a change of some element. In Fig. 1 there is an example of a part of a car service IS designed in a BORM methodology. More about the construction of the element transformation log is in (P í c k a , 2006).

### Uses of the Element Transformation Log

If the model of an IS is created according to the principles mentioned above, it will be able:

- To assure consistency of the information system being developed.
- To better document the process of creating the IS.
- To control the created model using the rules mentioned above.
- To localize changes – when a change occurs in an existing project, it is possible to easily identify elements that will be affected by this change – they are those that originated from the changed element by a transformation chain.

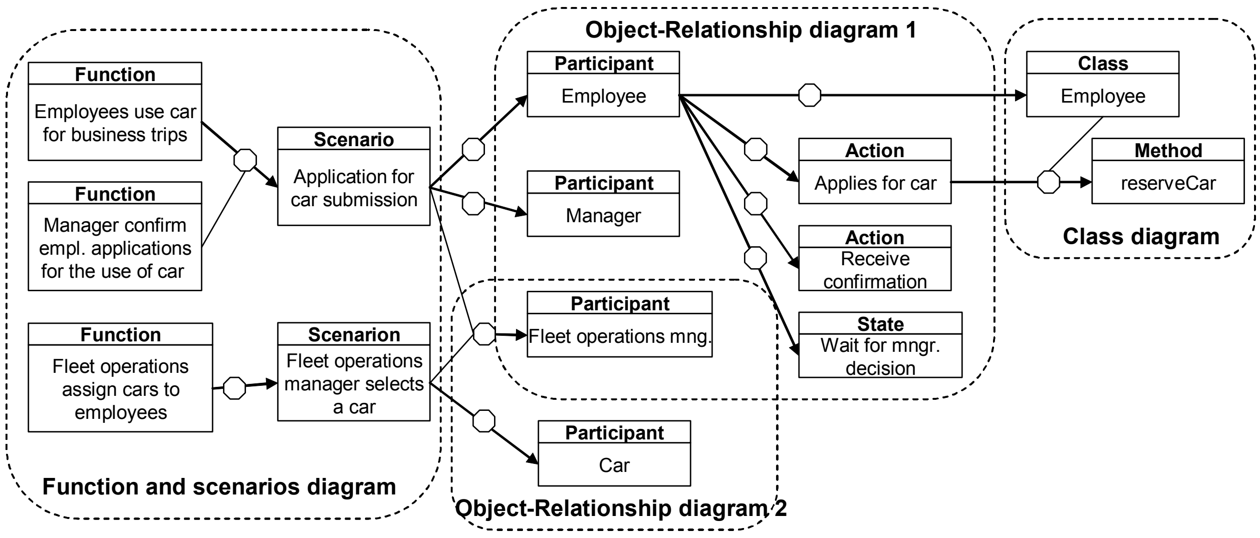
Function and scenarios diagram
Function
Employees use car for business trips
Function
Manager confirm empl. applications for the use of car
Function
Fleet operations assign cars to employees
Scenario
Application for car submission
Scenario
Fleet operations manager selects a car
Object-Relationship diagram 1
Participant
Employee
Participant
Manager
Action
Applies for car
Action
Receive confirmation
State
Wait for mngr. decision
Object-Relationship diagram 2
Participant
Fleet operations mng.
Participant
Car
Class diagram
Class
Employee
Method
reserveCar

Fig. 1. An example of elements transformation log for car fleet (in BORM)

Construction of model element transformations without additional knowledge is tough in finding the predecessors of the element and it is very laborious. There is a danger that the IS creator will stop following the mentioned rules, or they will carry them out just formally.

## CONCEPT TRANSITION MODEL

During the IS design, the construction of the element transition log helps us to just a limited extent. The above mentioned rules just tell us that we cannot add new elements arbitrarily – every newly added element shall have its predecessor. This forces the designer to think about the context of every newly added element and it decreases the probability of errors in design. However, the designer is not advised as to by which transformation a new element is created. So, during the design of IS it would be worth knowing, which elements can occur in a given context. To this end, we need to specify admissible transformations.

The creation of new elements is driven by the method of analysis and design of the information system. The method specifies which transformations can be in used in a given context and which new elements can be created. So we need to depict the terms used in the method and the possible transitions between them. We need to create a "data-flow" model of the method. We named this model the Concept Transition Model.

Unfortunately, in the methods used for analysis and design of ISs those transitions are not explicitly specified. For their depicting we need a new apparatus. It is described in following paragraphs.

### An Example of Transition Model

For illustrative reasons we will first show an example of model of transitions between the concepts of the model. For simplicity, we choose the transformation between the Chen entity-relationship diagram and the physical model of a relational database. This transformation is well-known and is often used. Almost every CASE tool used for relational database modelling does it automatically. Let us remind how it is done:

1. Transform all entities to tables.
2. If a relationship between entities is binary and of 1:N type without attributes, then transform the relationship to a new attribute (foreign key) and add it to the attributes of the table on the N-side. If the relationship is of 1:1 type, add foreign key to one of the tables.
3. Otherwise transform the relationship to a table. Add foreign keys pointing to the related tables to the attributes.
4. Transform remaining attributes of entities and relationships to attributes in the tables.

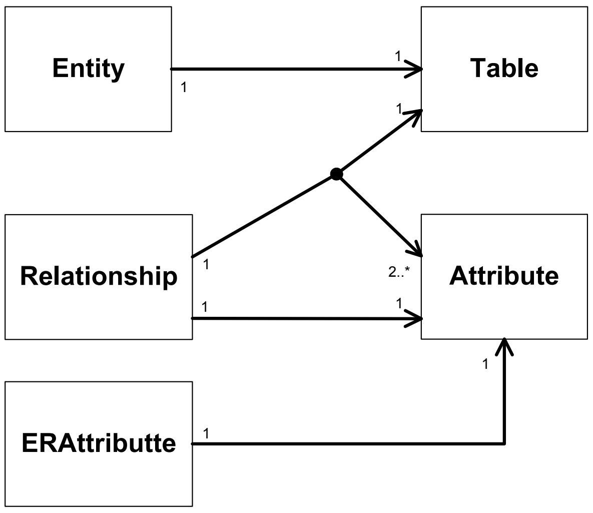

Fig. 2. Concept transitions diagram for ER to physical database model

This word-description is depicted using the diagram of concept transition in Fig. 2. It can be seen that (one) entity transforms into (one) table. A relationship can transform either into an attribute (foreign key) or into a table with two or more (according to the relationship's level) foreign keys (attributes). Attributes of entities and relationships transform to attributes of tables.

The above described word-description is better expressed by an algorithm, but a diagram better depicts relationships and possibilities in the transformation. This transformation can be done automatically, because we know the correct algorithm (see e.g. G o d o l l a , L i n d o w, 2003). However, this is not typical in methodologies of analysis and design. We typically know the relations between concepts of the methodology, but the concrete realisations of these relations are chosen by analyst according to their experience.

## DIAGRAM OF CONCEPT TRANSITION

The possible relations in the model of concept transition can be optimally shown using a metamodel. This metamodel is in Fig. 3. The metamodel depicts the possible relations between concepts and transitions. Transitions between concepts can be generally of M:N type – i.e. one concept may originate by a transition from more concepts and during one transition there can originate more concepts at once. Both relations between a concept and a transition have their cardinality. This is depicted by an association class Cardinality in the metamodel.

Just the relations between elements in the model are defined by metamodel. To be able to define the structure of a diagram, we need a graphical representation of each element in the diagram. In Fig. 4 there is graphical representation of elements of diagram of concepts' transition. A concept is depicted by a rectangle with its name and transition by a circle. A name of the transition can be noted by it. A transition is connected with its input concepts by links and with its output concepts by arrows from transition to concept. Cardinalities are noted by input and concepts.

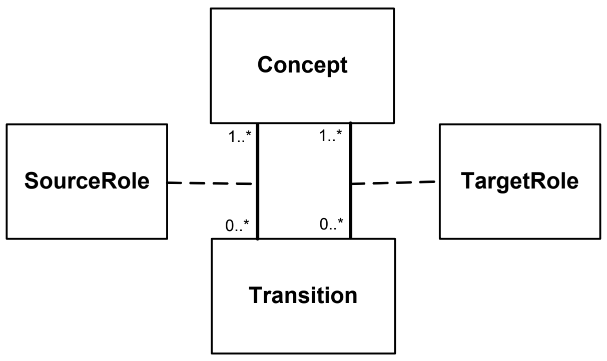

Fig. 3. Metamodel of concept transitions

## USAGE OF THE DIAGRAM OF CONCEPT TRANSITION

With the method of IS analysis and design recorded by the model of concept transition it is possible:

- To manage the development process – in every moment it is possible to say, which transformations the method allows (it is possible for instance to create a CASE tool capable of possible transformations offers). We know which and how many elements can originate in the next step of method.
- To check, whether the IS model matches the used method. It is possible to control, whether the element added to the model matches the model. The Craft-CASE modelling tool supports such control (see Craft. CASE homepage).
- Such record helps in performing some transformations automatically or semi-automatically. It is necessary to add the algorithm that defines the transformation.
- To depict the process of a method – this model can be used for defining relations in a method and this can be used for instance for easier understanding of relations inside the method, for method teaching, etc.
- To control and improve methods – by having all concepts and transitions defined, it is possible to control, whether transition between elements is not too rough (e.g. it doesn't transform directly to final classes, in the extreme) or too fine.

## BORM

BORM (Business Objects Relational Modelling) – see M e r u n k a et al. (2003) – is an object-oriented method of IS analysis and design. It focuses on processes running inside the modelled system, on their revealing, analysis and following modelling. BORM is an interactive method and is based on spiral model of system design. One of the main rules in BORM is depicting of its terms using sequential transformations.

A process model is in BORM depicted as a set of mutually communicating final automata. Those automata represent business objects. After modelling all processes using diagrams of processes a process model is created. In this moment, a lot of BORM-based projects end – BORM is often used just for process analysis, e.g. for reengineering processes purpose.

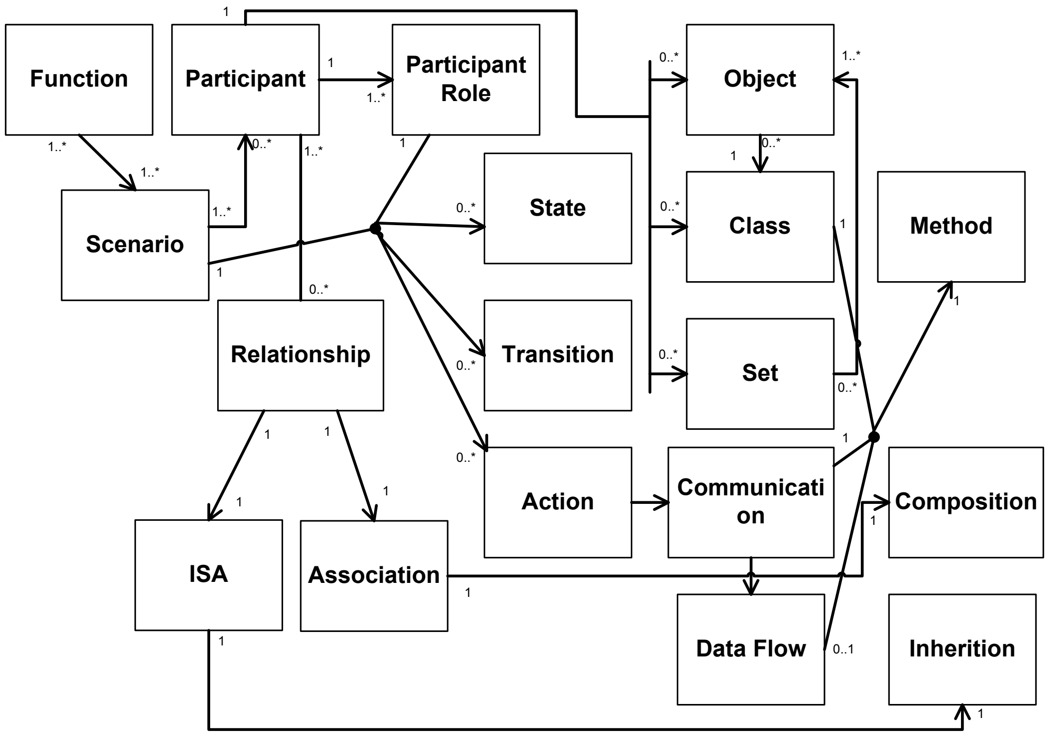

Fig. 4. Diagram of concept transitions of BORM methodology

## Process Diagram in BORM – Object Relationship Diagram (ORD)

As the next step the transformation of the process model into conceptual model may follow. That is a virtually ideal model created with no attention to implementation environment. Its graphical language uses a simplified UML class-diagram (see OMG, 2002).

The last step is a transformation of the conceptual model into the implementation environment and the ensuing implementation. This transformation may be considerably easy – with usage of a pure object-oriented language (e.g. Smalltalk) and with usage of object databases. It may be not so easy with the use of relational databases, because of the bigger semantic gap. The advantage of this approach is that the model remains implementation-independent as long as possible; implementation decisions are made during the final stage.

## BORM and the Concept Transformation

The basic idea of BORM methodology is based on transitions between its concepts. So demonstrate these principles is easy and straightforward (see Fig. 4).

## THE UNIFIED PROCESS

The Unified Software Development Process (USDP) methodology, known better under its shortened name Unified Process (UP) is one of many object-oriented methodologies based upon the UML language. This methodology comes directly from the authors of UML (Booch, Jacobson, Rumbought – see Jacobson et al., 1999) and is (together with its derivatives – e.g. RUP, see Kruchten, 2004) the most commonly used iterative methodology.

## UP and the Concepts Transition

To implement the ideas of sequential transformations during the IS design for the UP methodology is not as easy and straightforward as in the case of BORM. One of the problems is that the methodology itself consists of many alternative methods. For the use of concept transitions we must deal with individual methods and develop the overall way through the methodology from them. In UP it is the smartest to construct transition diagrams in each work procedure.

The next problem is that transitions between concepts are not explicitly defined in the methodology. The diagram of concept transition for the work procedure of finding actors and use-cases is in Fig. 5.

## CONCLUSIONS

The model of concept transition allows to view methods of IS analysis and design from a new perspective. It gives an apparatus for formalizing relations between concepts in the model and their successiveness. The model helps gain a better understanding of a method. The fact that relations inside this method are well defined improves the method's manageability and the possibilities to improve it.

During the IS development, by using the model of concept transition we get several advantages. The model can be used for managing the development process, for control of the method usage and for depicting the method's process.

Existing CASE tools support some ideas of the model of concept transitions, e.g. CraftCASE modelling tool performs checks, whether the added element conforms to the method. To further improve the quality of analyst's work, it would be a great contribution to implement complex

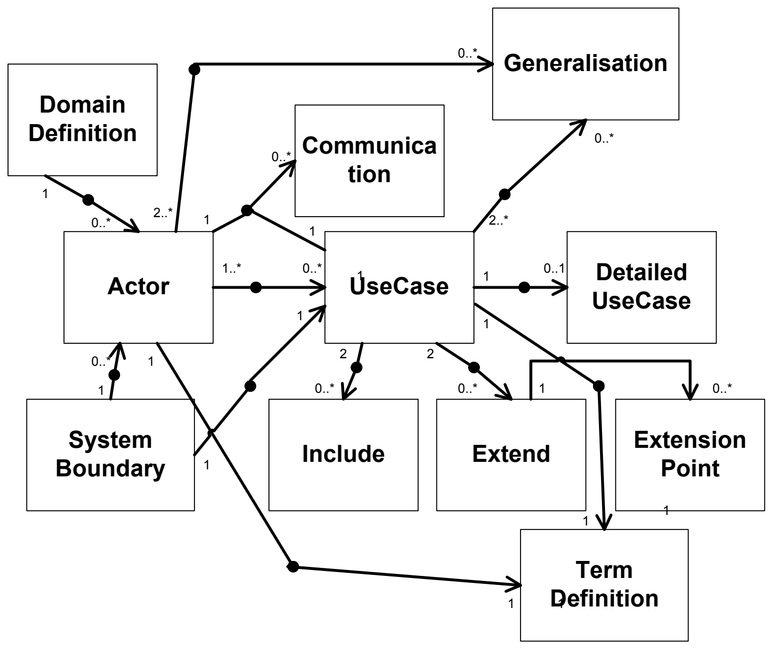

Fig. 5. Diagram of concept transitions of Use Cases

support for the concept transitions model. A CASE tool could thus better lead an analyst through the process of analysis, give him hints, check and record his steps.

## REFERENCES

GODOLLA, M. – LINDOW, A.: Transforming Data Model with UML. In: Knowledge Transformation for Semantic Web. Amsterdam, IOS Press 2003.

JACOBSON, I. – BOOCH, G. – RUMBAUGH, J.: The Unified Software Development Process. Addisson Wesley Profesional 1999.

KRUCHTEN, P.: The Rational Unified Process: An Introduction. 3rd ed. Addison Wesley Professional 2004.

LIU, L. – ROUSSEV, R. – KNOTT, R. – MERUNKA, V. – POLAK, J.: Management of the Object-Oriented Development Process. Part 15: BORM Methodology. University of Akron. University of Virgin Islands, 2005.

MERUNKA, V. – KNOTT, R. – POLAK, J.: The BORM Methodology: a third generation fully object-oriented methodology. In: Knowledge-Based Systems. New York, Elsevier Science International 2003.

PÍCKA, M.: Metamodeling and Development of Information Systems. In: Zeměd. Ekon., *50*, 2004 (2).

OMG. OMG Unified Modeling Language Specification – version 1.5. 2002.

PÍCKA, M.: Gradual modelling of information systems – Model of Method Expressed as Transitions between Concepts. In: Proc. 8th Int. Conf. on Enterprise Information Systems – ICEIS 2006. INSTICC – Institute for Systems and Technologies of Information and Communication, 2006.

Craft.CASE homepage. http://www.craftcase.com.

Received for publication on January 14, 2008
Accepted for publication on March 3, 2008

PÍCKA, M. (Česká zemědělská univerzita, Fakulta provozně ekonomická, katedra informačního inženýrství, Praha, Česká republika):

**Vývoj informačního systému za pomocí metody postupných transformací.**

Scientia Agric. Bohem., *39*, 2008: 61–66.

Článek popisuje a demonstruje základní principy metody postupných transformací informačního systému. Princip této metody vychází z toho, že model informačního systému vzniká přidáváním nových prvků do něj na základě již v modelu existujících prvků pomocí transformace. Tyto principy lze shrnout do těchto bodů:

- Každý nový prvek, přidávaný do modelu informačního systému, musí mít smysl.
- Každý nový prvek v modelu, kromě vstupních, vzniká právě jednou transformací.
- Transformace transformuje několik (jeden nebo více) prvků na několik (jeden nebo více) následnických prvků.
- V modelu existují tzv. vstupní prvky, které v modelu nemají žádného předchůdce a vznikly přímo ze zadání.

Proces tvorby takto vytvářeného informačního systému jsme schopni zachytit pomocí diagramu transformací mezi prvky. Pokud budeme takto konstruovat informační systém, potom nám to přinese tyto výhody:

- Budeme moci lépe dokumentovat průběh vytváření informačního systému.
- Pomůže nám to zprůhlednit vztahy mezi prvky vyvíjeného informačního systému.
- Budeme moci kontrolovat vytvářený model pomocí výše uvedených pravidel.
- Budeme moci lépe lokalizovat změny – pokud nastane změna v už existujícím projektu, tak lze snadno určit všechny prvky, kterých se tato změna týká (jsou to všechny ty, které vznikly ze změněného prvku řetězem transformací).

Na takto vytvořený model se lze podívat z hlediska metamodelu. Tento metamodel se bude skládat z pojmů, což jsou entity, s kterými pracujeme v rámci metody (např. třída, funkce apod.), a z transformací mezi nimi. Tyto transformace mezi pojmy jsou dané použitou metodou návrhu informačního systému. Tak dostaneme model transformací pojmů metody. Tento model metody nám ukazuje všechny v metodě použitelné pojmy a způsoby jejich vzniku. Takto vytvořený model metody nám umožňuje:

- Řízení vývojového procesu metodou.
- Kontrolu vytvářeného modelu vůči metodě.
- Automatické či poloautomatické provádění transformací.
- Znázornění průběhu metody.
- Kontrolu a vylepšování metod.

Tento model jsme schopni vytvořit pro jakoukoliv metodu návrhu informačního systému. Článek to demonstruje na příkladu metody BORM a UP.

metodika BORM; Unifikovaný proces; model; transformace; softwarové inženýrství; model metody

*Contact Address:*

Ing. Marek Pícka, Česká zemědělská univerzita v Praze, Fakulta provozně ekonomická, katedra informačního inženýrství, Kamýcká 1076, 165 21 Praha 6-Suchdol, Česká republika, tel.: +420 224 383 244, e-mail: picka@pef.czu.cz