Modeling of Unified Process Metamodel Structure in Association-Oriented Database Metamodel: Flowing Resources and Transformation

Marek Krótkiewicz, Marcin Jodłowiec
Institute of Computer Science,
Opole University of Technology,
Poland

Wojciech Przemysław Hunek
Institute of Control Engineering,
Opole University of Technology,
Poland

Krystian Wojtkiewicz
Department of Information Systems,
Wroclaw University of Science
and Technology,
Poland

***Abstract*—The paper is devoted to presentation of *Unified Process Metamodel* as a tool to describe both structural and behavioral aspects of processes identified in a real world. The complexity of the description of the reality urges the use of sophisticated tool for modeling. Therefore *Association-Oriented Database Metamodel* was used to prepare database model that comply with the needs of proposed system. Since presentation of complete implementation of the *Unified Process Metamodel* would be rather wide, the paper presents and discusses only a minor section of the system, namely *Transformation* module that is used to change flowing resource set into new flowing resource.**

***Index Terms*—Process modeling, resources management, resource allocation, Association-Oriented Database Metamodel, Unified Process Metamodel.**

I. Introduction

There are various approaches toward process based description of reality depending on the fields i.e. production engineering, management, information technology or automatic control engineering. Each of those fields independently develop domain-specific solutions and focuses only on processes aspects important to them only [1]–[6]. However, the most important of all is the basic principle why process modeling is needed at all. Whichever of abovementioned fields is considered it is always a matter of organization understanding or the need for change of already implemented way of service. The process analysis, modeling and enhancement leads to at least one of following: reduced process cost, increased quality, reduced number of errors, reduced process throughput time, reduced training time of new employee, reduced number of support requests etc. Therefore, process modeling is considered to be one of the most important aspects of any organization, at any level of complexity, in terms of management, control and knowledge sharing. Proper modeling of the processes involves many orthogonal aspects such as the flow of the process, stakeholders involved, resources (artifacts) consumed or produced, requirements etc. Those aspects should not only be properly and unambiguously defined but also clearly presented to all parties involved in process usage. It is especially important if the artificial actors are involved. The computer systems in the aspect of management or knowledge base systems are just examples of such artificial actors' class.

978-1-5090-5795-5/17/$31.00

Since process approach has been used for decades now, there are i.a. two mainstream modeling techniques used, namely BPMN [7] and UML [8]. Both notations originate from IT approach to process management. BPMN has been focused on process modeling from the very beginning of its creation, while UML diagrams have been barely adopted for this tasks. The differences and similarities between those two approaches are a scene of the discussion for years [9]–[11]. Both approaches are changing over years, but still in most cases they are used parallel and supplement each other.

The main idea standing behind the solution presented in the paper, namely *Unified Process Metamodel* (UPM) is to introduce one coherent and complementary system covering all issues and approaches in the scope of process modeling. The solution consist of:

- theoretical basis (full identification of categories and relations between them),
- formal notation,
- modeling language and methodology,
- algorithms and processing methods.

The above-mentioned components are arranged in space consisting of material resources, time, finances and location dimensions [12], [13]. For proper understanding of the process the distinction on four layers has been proposed: *Data Layer* (DL), *Control Layer* (CL), *Flow Layer* (FL), *Infrastructure Layer* (IL) (Fig. 1). Each of the layers has components specific to its nature. At the same time, all layers combine with one another and form a coherent system. In particular, the *Control Layer* uses the *Data Layer* and cause that processes can be set in motion and carried out; the *Flow Layer* contains processes that use the *Infrastructure Layer*.

Process models in the UPM has structural and behavioral aspects. The structural aspect is defined by the internal structure of processes, their interrelationships and connections between elements included in processes. Moreover, components of the DL and the IL should be included in structures.

The behavioral aspect is represented by the components found in the *Control Layer* and is based on the concept of

Event Condition Action (ECA). In this concept, *action* is taken as a result of the occurrence of a given event provided that the condition for the performance of this action is performed.

The implementation of the UPM is not a trivial one, since it includes not only the construction of graph-based structures of process but also requires knowledge representation methods for the definition of key system elements e.g. *Flowing Resources*, *Infrastructure* etc. Therefore authors decided to use *Association-Oriented Database Metamodel* (AODB) [14], [15] as a basis for the definition of data structures of UPM implementation. However, the complexity of UPM is high, thus the authors decided to focus on one of the system aspects, namely transformation of flowing resources. In order to properly define the process of transformation the flowing resource approach presented in UPM has been provided in section II. In the following section authors elaborate on the *Transformation* (T) module that is responsible for transforming set of *Flowing Resources* (FR) into new resources. After that the data structure of UPM in AODB is provide along with the motivation.

II. Flowing Resource

The *Flowing Resource* (FR) in the process is generated in*Resource Sources* and absorbed by the *Resource Target*. FR is a (material or immaterial) entity of continuous or discrete nature that constitutes an input, intermediate or output element of the process modeling. The consumption capacity is the basic feature of the FR.

For example, a man-hour is FR which is used in the processing component whereas an employee as a *Static Resource* is not used and it cannot be sent[1] in the process. A *Message* is not the FR either, for example, due to the above-mentioned reasons, so it was recognized as a separate primitive notion of this metamodel.

The types of FR have specific taxonomies within specific processes. The *Resource Taxonomy Diagram* (RTD) contains the inheritance structure, i.e. so-called generalization-specialization of types of FR occurring in the description of the process.

It is deemed that abstract types of FR also exist, i.e. these which do not have specific occurrences (instances). A *Flowing Resource* defined as a liquid is an example of the above. There is no liquid as such; its specializations can take specific forms, e.g. fuel, water, milk. However, a liter of fuel can be FR. The *Resource Flow Rate* (RFR), e.g. liter of fuel per minute, is the parameter of the *Resource Source* (RS).

Flowing Resources can be of quantized nature, i.e. in spite of a specified unit in the form of e.g. liters, they can be aggregated in the form of elements of resources of specific capacity. For example, the resource "fuel" can be sent in the form of barrels as the smallest "pack" of this resource that can be taken into consideration. It does not affect the fact that liters can still be used as a basic unit. Therefore, it is necessary to create a unit conversion map so that it could be possible to express a given resource in a useful and convenient way for a given model.

Certain types of resources are used even though they do not take part in any process. This is the case of man-hours. Even if a given machine is not used for some time, its man-hours are lost forever, similarly to time. Resources of this type, e.g. man-hours, cannot be stored. The number of available resources in the form of man-hours of the machine decreases with time. It means that it is not possible not to use the machine for 7 hours and, then, to use all of its 8 man-hours in the eight hour. It is connected with the parameter of RS defined as the maximum available RFR which specifies how many units of a given resource can be used in a given period of time. Nevertheless, it can be limited by the *Resource Channel* (RC) by means of which it is supplied. This is the case of electricity. Usually there is not less and less electricity if its resources are not used, but it does not mean that it can be obtained from the source in any amount in a short time. In case of electricity, it corresponds to the concept described by the notion of "installed capacity" that limits the availability of the resource in time, i.e. maximum RFR.

The *Flowing Resource*, as a type, is formally defined as:

$$\sigma\left(FR\right) = \begin{pmatrix} id : ID, \\ unit : ID, \\ pack : double, \\ storage : bool \end{pmatrix} \tag{1}$$

where:

FR – *Flowing Resource* type,
id – identifier of FR,
$unit$ – basic measurement unit of the FR,
$pack$ – the smallest amount („pack") of the resource which can be used,
$storage$ – boolean value determining whether a given FR has the ability to store it.

Unlike the *Static Resource* (SR), FR does not have a direct representation in the form of a component. It is an element that "flows" through components.

A. Inputs and Outputs of Resources

Each of the modules that have inputs and outputs must have a certain specificity concerning the possibility of forcing the *flow of resources* (Fig. 2). If the flow of resources is not forced, the resource flow rate remains at the same level, otherwise it can be of:

- suction nature for inputs (In) – suction pump (SP),
- forcing nature for outputs (Out) – force pump (FP).

The force is determined by the value of the resource flow rate that is defined as the amount of *Flowing Resource* FR per time unit:

$$rfr = \frac{fr}{t} \tag{2}$$

where:

rfr – *resource flow rate*,

[1]The relocation of SR is an operation of completely different nature than the resource flow because it relates to the change in the process model structure and not in its functioning.

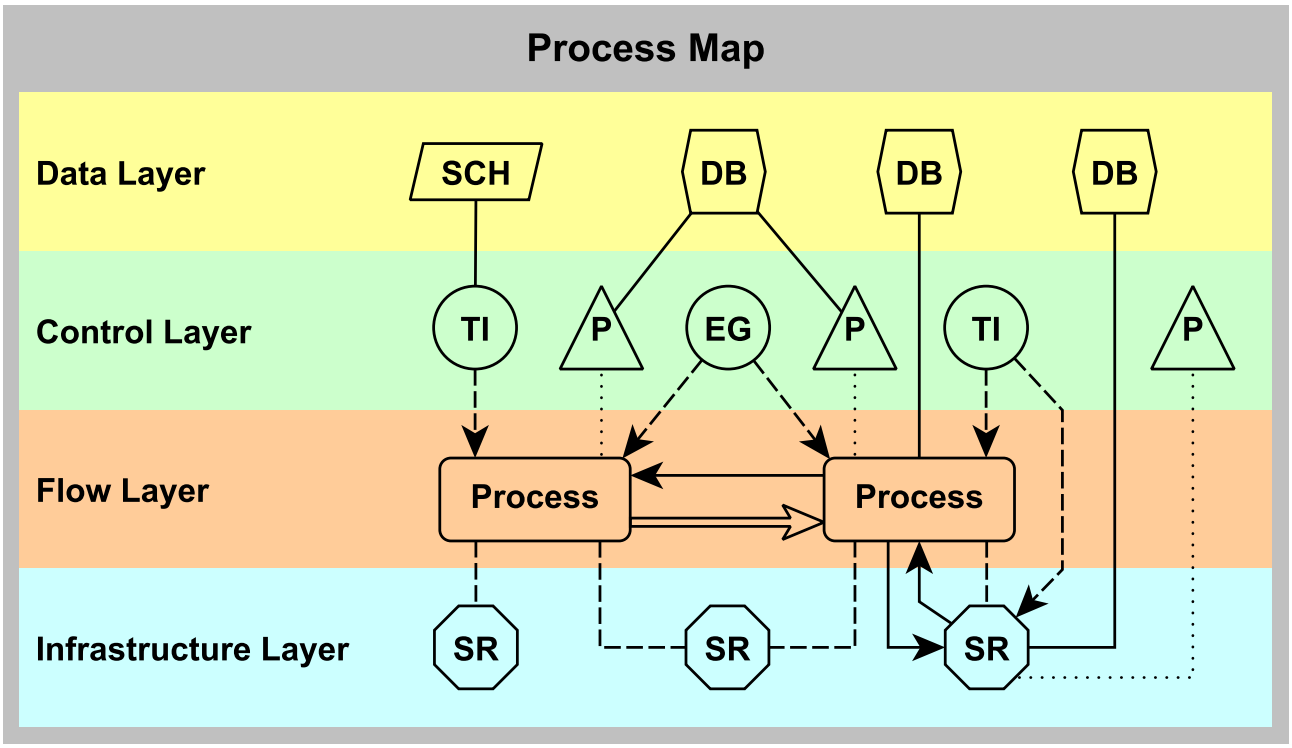

Fig. 1. UPM modeling layers.

fr – *flowing resource amount*,
t – *time*.

Examples of values of resource flow rates:

- 25 litres of water per minute,
- 42 man-hours per week,
- USD 2,500 per month.

When determining the levels of specific resource flow rates in the RC, one should follow the principle saying that the actual RFR is limited by the availability of the FR. It means that regardless of the value and nature of the force, it cannot exceed the value of the flow rate of the resource supplied to the input. In a special case, one cannot, for instance, expect that as a result of the suction force, FR will be taken from the storage if the storage is empty. Another case concerns a direct connection of two modules one of which would force the FR with a certain capacity and the other would suck this resource with another capacity. Such suction and force differences are possible only if there is a buffer between them in the form of the storage.

III. Transformation

The *Transformation* module is responsible for the transformation of input resources into output resources. The set of types of input resources must differ from the set of types of output resources. Input materials are transformed into output resources according to the formula determining the proportions

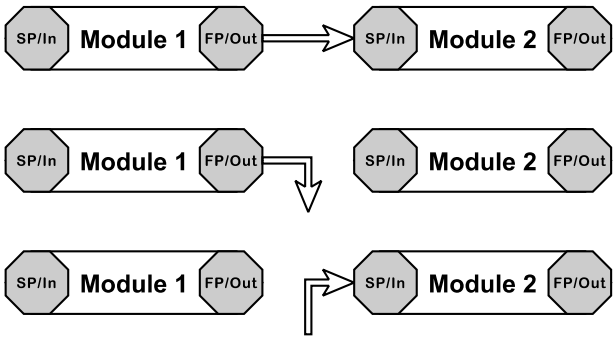

Fig. 2. Graphical representation of options of connecting and disconnecting the RC to modules .

in which they are used per unit of the resource being the result. Both material FR performing the role of raw materials or semi-finished products and immaterial resources, such as e.g. energy or financial resources, are consumable resources. Moreover, the productivity of a given module is determined as the number of generated output resources (results) per time unit (Fig. 3).

In case of assuming another configuration of this module, it generates resources on the basis of the SR. It means that FR

are generated within this module on the basis of exactly one SR and no other FR are not absorbed for this purpose.

Fig. 3. Graphical representation of the T module for the transforming version (on the left) and for the generating version on the basis of the SR (on the right)

Formally, T module, as a type is defined as:

$$\sigma(T) = \begin{pmatrix} name : NAME, \\ in : \langle PORT \rangle, \\ out : \langle PORT \rangle, \\ tr : \langle TR \rangle \end{pmatrix} \tag{3}$$

where

$$\sigma(TR) = \begin{pmatrix} rawmat : \langle FRP \rangle, \\ result : FRP, \\ mintime : time, \\ t : time \end{pmatrix} \tag{4}$$

and

$$\sigma(FRP) = \begin{pmatrix} quantity : double, \\ unit : NAME, \\ fr : FR \end{pmatrix} \tag{5}$$

where

T – the type defining *Transformation*,
$name$ – module name,
in – list of input *ports*,
out – list of output *ports*,
TR – type defining characteristics of the transformation,
tr – list of transformations,
$rawmat$ – list of materials used in the transformation, input FR,
$result$ – the result of transformation, output FR,
$mintime$ – minimal time needed to produce the result (maximum productivity),
t – current time needed to produce the result (current productivity),
FRP – type defining information in regard to particular pack of *Flowing Resource Pack* (FRP),
$quantity$ – quantity of FR,
$unit$ – unit of FR,
fr – *Flowing Resource*.

As the result of T module running, on the base of transformation table $tr : [TR]$, in defined time $t.time$ the $result : FRP$ resource is being produced, where

$$TR.t \geqslant TR.mintime \tag{6}$$

At the same time resources defined as materials $rawmat : [FRP]$ are consumed and are deleted from the system.

IV. Association-Oriented Modelling

Database layer of UPM (Fig. 4) was modeled in AODB due to the fact that the this metamodel has a high expressiveness that allows building complex solutions using relatively small number of elements. The most important features of AODB are [14]:

- performing exactly one function by each metamodel primitive as well as no overlying of primitive concepts in the functional aspect;
- direct implementation of the conceptual layer in a physical layer - without the need for mapping meta-model to another one;
- separation of data containers and the relationship between them - at the level of metamodel primitives, rather than in the semantics of a particular model;
- definition of data containers, both in terms of structure and mechanism of inheritance, take into account virtuality;
- definition of the relationships in aspects of:
  - arity - any arity of relationships is supported,
  - multiplicity - unrestricted multiplicity, both the data and the relationships site is allowed,
  - mechanisms implementing:
    - lifetime dependency – defined separately for each side of elements in relationship
    - navigability – the ability to physically move around relationships between elements of the database, in accordance with specific directions of references to these elements,
    - directionality – conceptual definition of the relationship direction, i.e. it does not reduce the possibility to move around the data structures,
    - inheritance – taking into account virtuality.

AODB also contains *Association-Oriented Data Language* (ADL), which allows to manipulate data for all CRUD operations (Create, Read, Update, Delete). Explicitly for AODB, there has also been defined the *Association-Oriented Query Language* (AQL), which is dedicated to association and graph-oriented language allowing to build complex queries using relatively short phrases and simple expressions.

An attempt to model the *Database Layer* of UPM in the relational database would be ineffective due to the lack of distinction in this database metamodel of data and relationships. As a result, created model would be ambiguous. Moreover, relationships between relations are implemented in a behavioral layer by performing operations based on searching data sets by the use of key attributes. This results in a significant load on the servers e.g. while join operations, involving relations with a large number of tuples. One of the most important features of AODB is the ability to model n-ary relationships directly, i.e. it does not require converting them into a set of

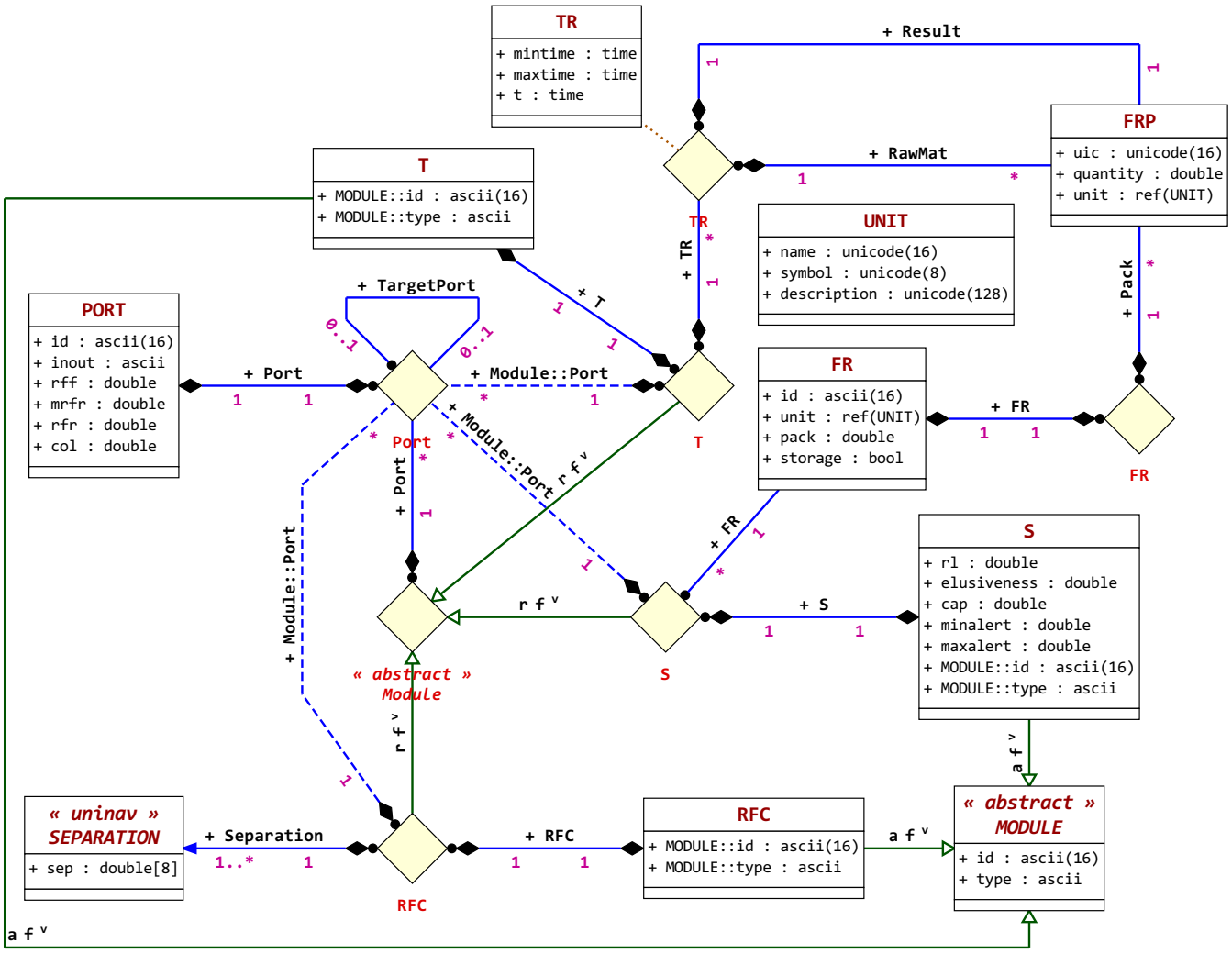

Fig. 4. Diagram of *Association-Oriented Database* implementing UPM meta-structure

.

binary compounds as is e.g. in relational metamodel or object-oriented database metamodel (ODMG 3.0) [16]. Modeling relationships as n-ary ones ($n \geqslant 1$) is natural for man who perceives reality just as such compounds. This issue was very extensively presented and thoroughly discussed in [17].

Another very important element is the modeling language, i.e. *Association-Oriented Modeling Language* (AML), which is an integral part of the AODB. It enables efficient and semantically unambiguous modeling of complex structures. It must be emphasized that AML is a language dedicated to AODB and built to its needs. Each element of AODB is reflected in AML, and created models do not require any modification consisting of e.g. mapping from the conceptual level to logical or physical one. Each model is automatically and without any simplifications converted into a ready-to-use database that can be physically stored on any data medium. Case study of medical database has been shown and described in [15], which indicates the most important elements of the AML.

The model of UPM *database Layer* (Fig. 4) shows how authors realized the structural aspects of the system in the chosen database metamodel. The diagram do not show the complete solution but rather the part crucial for this papers scope. One of the most important issues was to properly represent the relationships between key UPM primitives i.e. *Modules*, *Ports* and *Resources*. The taxonomy of *Modules* was transfered into appropriate structures of associations connected by generalization-specialization relationship (abstract collection MODULE is generalization of collections: RFC, S, T). *Module* has its *Ports*, what was represented by the compositional role Port in association Module to the association Port, that is inherited by all of child associations. Port as an association has a PORT collection assigend (role Port), which is used to store its properties. The structural aspect of *Transformation* is stored in the structure built from association TR and the collections: TR, FRP and UNIT. There is

also information regarding FR presented as a bicompositional tandem of association FR and collection FR.

V. Summary

In this paper the approach towards implementation of *Unified Process Metamodel* in *Association-Oriented Database Metamodel* has been presented. It was crucial to use the metamodel that would natively support the design of graph-based structures that AODB provides. Due to extensiveness and complexity of UPM definition authors decided to focus on one aspect of the system. The choice of *Transformation* and *Flowing Resources* seemed to be the best one, since processes are being mainly perceived as actions leading to the change of input into output. The transformation module is dedicated metamodel element that is responsible for transforming set of flowing resources into new resources. For the ease of perception the formal definition of FR has been provided as well as formal definition of T module. Furthermore, the motivation towards the choice of AODB as the database layer has been provided. There have been also complete structure of UPM presented.

Appendix

List of important acronyms:

ADL – *Association-Oriented Data Language*
AML – *Association-Oriented Modeling Language*
AODB – *Association-Oriented Database Metamodel*
AQL – *Association-Oriented Query Language*
FR – *Flowing Resource*
FRP – *Flowing Resource Pack*
RC – *Resource Channel*
RFR – *Resource Flow Rate*
RS – *Resource Source*
RTD – *Resource Taxonomy Diagram*
SR – *Static Resource*
T – *Transformation*
UPM – *Unified Process Metamodel*

References

[1] A. Langley, C. Smallman, H. Tsoukas, and A. H. Van De Ven, "Process studies of change in organization and management: Unveiling temporality, activity, and flow," *Academy of Management Journal*, vol. 56, no. 1, pp. 1–13, 2013.

[2] N. Slack, S. Chambers, R. Johnston, and A. Betts, "Operations and Process Management," *Operations Management*, p. 760, 2012.

[3] F. Rahimi, C. Møller, and L. Hvam, "Business process management and IT management: The missing integration," *International Journal of Information Management*, vol. 36, no. 1, pp. 142–154, 2016.

[4] G. M. Giaglis, "A Taxonomy of Business Process Modeling and Information Systems Modeling Techniques," *International Journal of Flexible Manufacturing Systems*, vol. 13, no. 2, pp. 209–228, 2001. [Online]. Available: https://link.springer.com/article/10.1023/A:1011139719773

[5] W. P. Hunek and P. Dzierwa, "New results in generalized minimum variance control of computer networks," *Information Technology and Control*, vol. 43, no. 3, pp. 315–320, 2014, dOI: 10.5755/j01.itc.43.3.6268.

[6] W. P. Hunek, "An application of polynomial matrix σ-inverse in minimum-energy state-space perfect control of nonsquare LTI MIMO systems," in *Proceedings of the 20th IEEE International Conference on Methods and Models in Automation and Robotics (MMAR'2015), Międzyzdroje, Poland*, 2015, pp. 252–255, dOI: 10.1109/MMAR.2015.7283882.

[7] OMG (Object Management Group), "BPMN Specification - Business Process Model and Notation," 2015. [Online]. Available: http://www.bpmn.org/

[8] H. Eriksson and M. Penker, "Business Modeling With UML," *Business Patterns at Work*, p. 12, 2000.

[9] C. V. Geambaşu, "Bpmn Vs . Uml Activity Diagram for Business Process Modeling," *Accounting and Management Information Systems*, vol. 11, no. 4, pp. 637–651, 2012. [Online]. Available: http://www.cig.ase.ro/articles/11{_}4{_}7.pdf

[10] M. R. Khabbazi, M. K. Hasan, R. Sulaiman, and A. Shapi ', "Business Process Modelling in Production Logistics: Complementary Use of BPMN and UML," *Middle-East Journal of Scientific Research*, vol. 15, no. 4, pp. 516–529, 2013.

[11] J. Recker, M. zur Muehlen, K. Siau, J. Erickson, and M. Indulska, "Measuring Method Complexity : UML versus BPMN," in *Americas Conference on Information Systems*, 2009, pp. 1–9.

[12] M. Krótkiewicz, M. Jodłowiec, K. Wojtkiewicz, and K. Szwedziak, "Unified process management for service and manufacture system - material resources," in *Proceedings of the 9th International Conference on Computer Recognition Systems CORES 2015, Wroclaw, Poland, 25-27 May 2015*, ser. Advances in Intelligent Systems and Computing, R. Burduk, K. Jackowski, M. Kurzyński, M. Wozniak, and A. Zolnierek, Eds., vol. 403. Springer, 2015, pp. 681–690. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-26227-7_64

[13] S. Liu and C. Wang, "Resource-constrained construction project scheduling model for profit maximization considering cash flow," *Automation in Construction*, vol. 17, no. 8, pp. 966–974, 2008. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S0926580508000654

[14] M. Krótkiewicz, *Asocjacyjny metamodel baz danych. Definicja formalna oraz analiza porównawcza metamodeli baz danych*. Opole: Oficyna Wydawnicza Politechniki Opolskiej, 2016.

[15] M. Jodłowiec and M. Krótkiewicz, "Semantics discovering in relational databases by pattern-based mapping to association-oriented metamodel - A biomedical case study," in *Information Technologies in Medicine - 5th International Conference, ITIB, 2016 Kamień Śląski, Poland, June 20-22, 2016 Proceedings, Volume 1*, 2016, pp. 475–487. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-39796-2_39

[16] R. G. Cattell, D. K. Barry, M. Berler, J. Eastman, D. Jordan, C. Russell, O. Schadow, T. Stanienda, and F. Velez, "The Object Data Standard: ODMG 3.0," p. 280, 2000. [Online]. Available: http://www.amazon.com/The-Object-Data-Standard-Management/dp/1558606475

[17] M. Krótkiewicz, "Association-Oriented Database Model – n-ary Associations," *International Journal of Software Engineering and Knowledge Engineering*, vol. 27, no. 02, pp. 281–320, mar 2017. [Online]. Available: http://www.worldscientific.com/doi/abs/10.1142/S0218194017500103