Knowledge-Based Generation of the UML Dynamic Models from the Enterprise Model Illustrated by the Ticket Buying Process Example

Ilona Veitaite[1](✉) and Audrius Lopata[2]

[1] Institute of Applied Informatics, Kaunas Faculty, Vilnius University, Kaunas, Lithuania
ilona.veitaite@knf.vu.lt
[2] Faculty of Informatics, Kaunas University of Technology, Kaunas, Lithuania
Audrius.Lopata@ktu.lt

Abstract. The main scope of this paper is to introduce knowledge-based Enterprise model as sufficient data storage for different Unified Modelling Language (UML) models generation, by using all collected data. UML models can be generated from the Enterprise Model by using certain transformation algorithms presented in previous researches. Generation process from the Enterprise model is illustrated by a particular Ticket Buying example. Generated UML dynamic Use Case, Sequence, State and Activity models of the Ticket buying process demonstrate fullness of stored information in the Enterprise model.

Keywords: Knowledge-based · UML · Enterprise model · IS engineering

1 Introduction

Nowadays information system (IS) engineering process is quite challenging as for analysts, designers and as for any IS design process professionals. Enterprise modelling has become one of the most important elements in IS design process. Enterprise models applications are adapted in various ways and diverse types of models are created based on chosen Enterprise model [1, 2].

UML is a highly recognized and understood platform for IS design. It is a standard notation among professionals. UML can be used to model not just object-oriented IS engineering, but application structure, behavior, or/and business processes. UML models can generate code from the design, apply design patterns, perform impact and complexity analysis [1, 6, 8].

To ensure all these UML model applications is possible only then, when data used for UML models design is verified, validated and of enough quality. Enterprise model completely provides all necessary data and UML models generated from it by using transformation algorithms fully match this requirement [3, 5, 7, 9, 10].

Particular Enterprise meta-model and Enterprise model structure used as the background for this research are presented almost two decades ago. All previous researches

A. Lopata et al. (Eds.): ICIST 2020, CCIS 1283, pp. 26–38, 2020.
https://doi.org/10.1007/978-3-030-59506-7_3

are dedicated to prove that composition of these EMM and EM is enough for generation of different types of models in IS modelling process [10–12].

2 Knowledge-Based Enterprise Meta-model and Enterprise Model

EMM is formally defined EM structure, which consists of a formalized EM in line with the general principles of control theory. EM is the main source of the necessary knowledge of the particular business domain for IS engineering and IS re-engineering processes (Fig. 1) [3, 4].

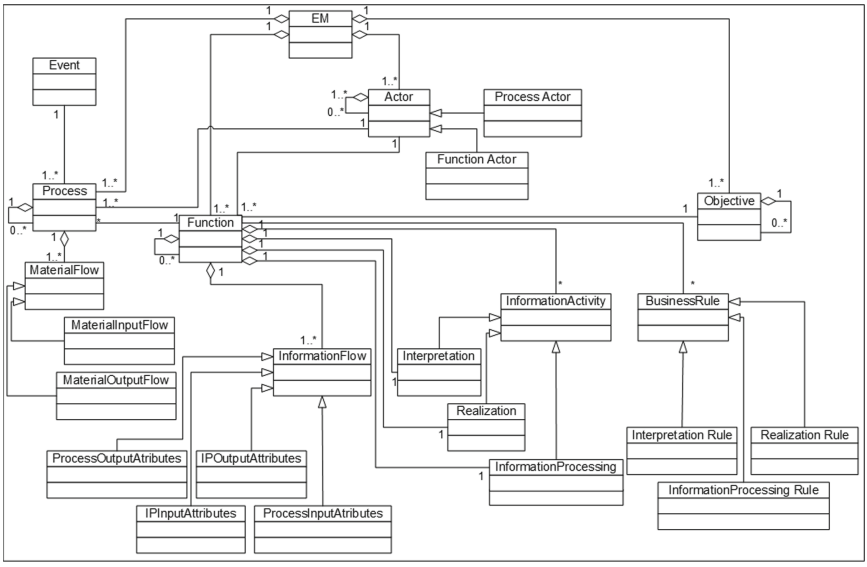

Fig. 1. Enterprise meta-model class diagram [3, 4, 10]

EM class model has twenty-three classes. Essential classes are Process, Function and Actor. Class Process, Function, Actor and Objective can have an internal hierarchical structure. These relationships are presented as aggregation relationships. Class Process is linked with the class MaterialFlow as aggregation relationship. Class MaterialFlow is linked with the classes MaterialInputFlow and MaterialOutputFlow as generalization relationship. Class Process is linked with Classes Function, Actor and Event as association relationship. Class Function is linked with classes InformationFlow, InformationActivity, Interpretation, InformationProcessing and Realization as aggregation relationship. These relationships define the internal composition of the Class Function. Class InformationFlow is linked with ProcessOutputAtributes, ProcessInputAtributes, IPInputAttributes and IPOutputAttributs as generalization relationship. Class InformationActivity is linked with Interpretation, InformationProcessing and Realization as generalization relationship. Class Function linked with classes Actor, Objective and Business Rule as

association relationship. Class Business Rule is linked with Interpretation Rule, Realization Rule, InformationProcessing Rule as generalization relationship. Class Actor is linked with Function Actor and Process Actor as generalization relationship [3–5, 11].

Figure 2 presents the transformation algorithm of UML model generation from EM process and is described by the following steps [10, 11].

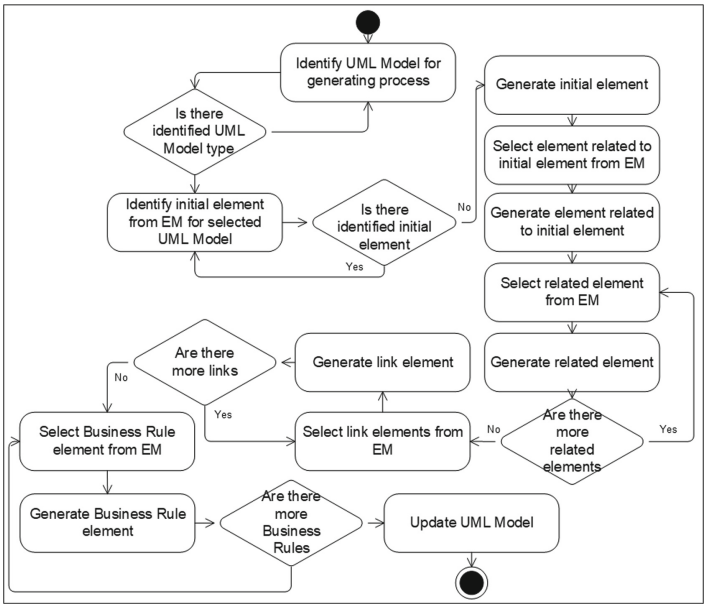

Fig. 2. The top level transformation algorithm of UML models generation from EM process [10–12]

- Step 1: Particular UML model for generation from the EM process is identified and selected.
- Step 2: If the particular UML model for generation from EM process is selected then algorithm process is continued, else the particular UML model for generation from EM process must be selected.
- Step 3: First element from EM is selected for UML model, identified previously, generation process.
- Step 4: If the selected EM element is an initial UML model element, then initial element is generated, else the other EM element must be selected (the selected element must be initial element).
- Step 5: The element related to the initial element is selected from the Enterprise model.
- Step 6: The element related to the initial element is generated as UML model element.
- Step 7: The element related to the previous element is selected from the Enterprise model.
- Step 8: The element related to the previous element is generated as UML model element.

- Step 9: If there are more related elements, then they are selected from EM and generated as UML model elements one by one, else the link element is selected from the Enterprise model.
- Step 10: The link element is generated as UML model element.
- Step 11: If there are more links, then they are selected from EM and generated as UML model elements one by one, else the Business Rule element is selected from the Enterprise model.
- Step 12: The Business Rule element is generated as a UML model element.
- Step 13: If there are more Business Rules, then they are selected from EM and generated as UML model elements one by one, else the generated UML model is updated with all elements, links and constraints.
- Step 14: Generation process is finished.

Table 1 presents part of Enterprise model elements and their descriptions in order to describe elements, which are necessary in this particular research.

Table 1. Description of knowledge stored in Enterprise model

Enterprise Model element	Description
Actor	In actor element can be stored information related with process or function executor. Actor element is responsible of information related with the process or function participant, it can be person, group of persons, subject such as an IS, subsystem, module and etc.
Process, Function	In process or function elements can be stored all information related with any user, entity, object, subject and its behavior. Process or function element is responsible of information related with any operation, activity, status change, movement which is implemented by any actor, entity, participant and etc.
Information Flow	In Information Flow element can be stored diverse information flow types, such as Information input and output attributes or/and process input and output attributes. Information Flow element is responsible of information related with each element input and output attributes, details which make impact on other elements, their state or status
Business Rule	In Business Rule element can be stored different rules such as interpretation, realization or/and information processing. Business rule element is responsible of information about how different elements in IS design phase are related; what restrictions and restraints are applied to these elements

3 Development of UML Models for Ticket Buying Process

This section deals with the detailed explanation of the Ticket buying process and how this process can be designed by using knowledge-based Enterprise model, where all knowledge related with the previously described example is stored. There is also explained, what knowledge is used for the generation particular UML models through certain transformation algorithms created for each UML model generation process [10, 12]. There are described UML Use Case, Sequence, State and Activity models generated form the Enterprise Model.

3.1 Ticket Buying Process Example and Its UML Models

The process of Ticket buying may seem very simple, but if this process would be analyzed from different perspectives in information systems design phase; if this process would be projected and designed for the fulfillment of its all possible functions it would take a lot of time and efforts of an analyst, designer and etc.

In IS lifecycle design phase all the details must be estimated. These details, this knowledge is stored in previously described Enterprise model and they are already verified and validated.

3.2 UML Use Case Model of Ticket Buying Process Example

A UML Use Case model is the primary form of system requirements for a new IS underdeveloped. Use cases specify the expected behavior – what?, and not the precise method of making it take place – how?. A key concept of use case modelling is that it assists to design a system from the end user's perspective. It is an powerful technique for communicating system behavior in the user's conditions by specifying all externally visible system behavior.

Table 2 presents UML Use Case model elements generated from the Enterprise model of Ticket buying example. In Enterprise Model all information related with actors, their functions and relationships between these functions is stored. There are three actors: Client, Manager and Ticket System. Ticket System as an actor is associated with all seven functions – use cases: Enquire ticket availability, Fill form, which includes use case of ticket booking or ticket cancelling, ticket booking includes ticket price payment and form printing, this includes ticket cancelling and this includes payment refunding. Client as an actor is associated with all functions except payment refunding, because it is Ticket system's function. Manager as an actor is associated only with two functions – uses cases: form printing and ticket canceling.

Figure 3 presents UML Use Case model of Ticket buying example generated step by step from the Enterprise Model through UML Use Case transformation algorithm.

Table 2. UML Use Case model elements generated from the Enterprise model of Ticket buying example [5, 7, 9]

Enterprise Model element	UML Use Case Model element	Ticket Buying example	Description
Actor	Actor	Client	There are three actors, each of them is behavioural classifier which defines a role played in particular example
		Manager	
		Ticket system	
Process, Function	Use Case	Enquire ticket availability	There are three use cases, each use case is a type of behavioural classifier that describes a unit of functionality performed by three actors
		Fill form	
		Book ticket	
		Pay ticket price	
		Print form	
		Refund payment	
		Cancel ticket	
Business Rule	Include	Six include elements	There are six include elements, each include is a directed relationship between two use cases which is used to demonstrate that behaviour of the included use case is inserted into the behaviour of the including use case

3.3 UML Sequence Model of Ticket Buying Process Example

UML Sequence model is an interaction model that detail how operations are implemented. This model captures the interaction between objects in the context of a collaboration. UML Sequence model is time focus and it shows the order of the interaction visually by using the vertical axis of the diagram to deliver time what messages are sent and when.

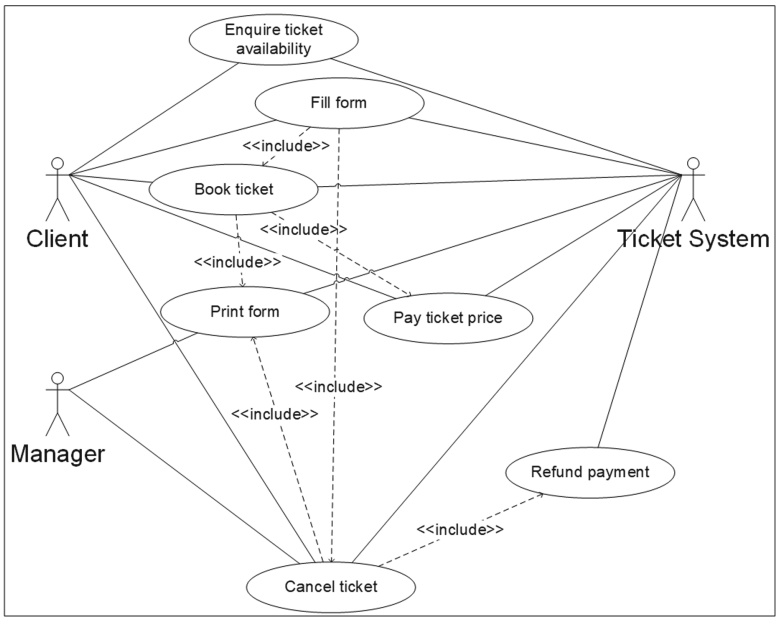

Fig. 3. UML use case model of ticket buying example

Table 3 presents UML Sequence model elements generated from the Enterprise model of Ticket buying example. In Enterprise Model all information related with actors and their collaboration is stored. There are three actors – process participants, which are called Lifelines in UML Sequence model: person – Client, subject – Ticket system, object – Ticket. Ticket has one execution specification, receives one message with details and sends one message of created ticket; Ticket system has three execution specifications, one is assigned for validation, after Client logs in, it returns result; second is assigned for form creation and third for ticket creation; all these are related with messages from Client. Client logs in, requests form, submits details, prints ticket – client sends four messages and receives two: validate login and acknowledgement of all requests of this particular process.

Figure 4 presents UML Sequence model of Ticket buying example generated step by step from the Enterprise Model through UML Sequence transformation algorithm [10].

3.4 UML State Model of Ticket Buying Process Example

UML State Model shows the different states of an entity. State model can also demonstrate how an entity responds to various events by changing from one state to another.

Table 4 presents UML State model elements generated from the Enterprise model of Ticket buying example. In Enterprise Model all information related with processes, functions and their state is stored. This model is from Client's perspective. There are

Table 3. UML Sequence model elements generated from the Enterprise model of Ticket buying example [5, 7, 9]

Enterprise Model element	UML Sequence Model element	Ticket Buying Example	Description
Actor	Lifeline	Client	There are three actors, in UML Sequence model three Lifelines, which are shown using a symbol that consists of a rectangle forming its "head" followed by a vertical line and these lines represent the lifetime of the actor – participant of the process
		Ticket system	
		Ticket	
Process, Function	Message	Login ()	There are eleven messages, related with actors and they define a communication between these actors
		Validate ()	
		Return ()	
		Request form ()	
		Create form ()	
		Submit details ()	
		Create ticket ()	
		Send details ()	
		Ticket created	
		Acknowledge	
		Take print ()	
Business Rules	Execution specification	Ten execution specifications	Each of ten executions specification element represents a period in the actor's lifetime

four information flows – composite states of a Client entity: validation, availability check, ticket booking and printing. All these composite states are conducted by particular behavioral state machine: Enter login details, Enter bus details, Enter self details, Booking successful, Logout.

Figure 5 presents UML State model of Ticket buying example generated step by step from the Enterprise Model through UML State transformation algorithm [10].

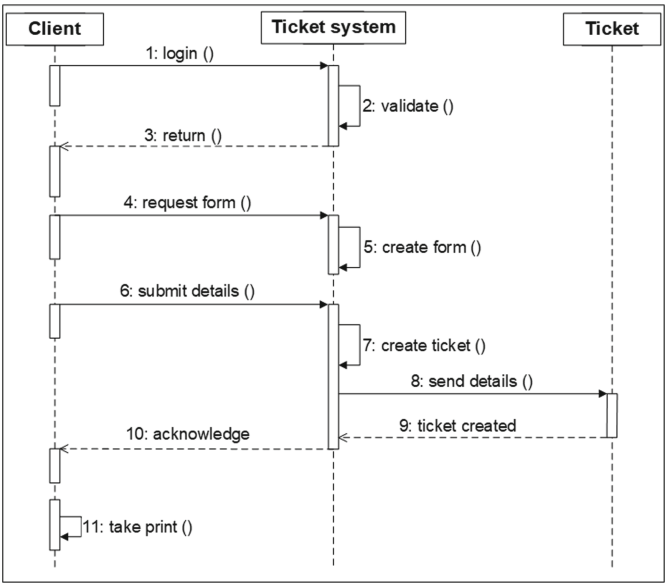

Fig. 4. UML sequence model of Ticket buying example

Table 4. UML State model elements generated from the Enterprise model of Ticket buying example [5, 7, 9]

Enterprise Model element	UML State Model element	Ticket Buying Example	Description
Process, Function	Behavioural state machine	Enter login details	Five states are used to specify discrete behaviour of a part of designed system through finite state transitions
		Enter bus details	
		Enter self details	
		Booking successful	
		Logout	
Information Flow	Composite state	Validation	Four states of an entity are defined as state that has substates
		Availability check	
		Booking Ticket	
		Printing	

3.5 UML Activity Model of Ticket Buying Process Example

UML Activity model describes how activities are coordinated to provide a service which can be at different levels of abstraction. Typically, an event needs to be gained by some operations, particularly where the operation is intended to gain a number of different

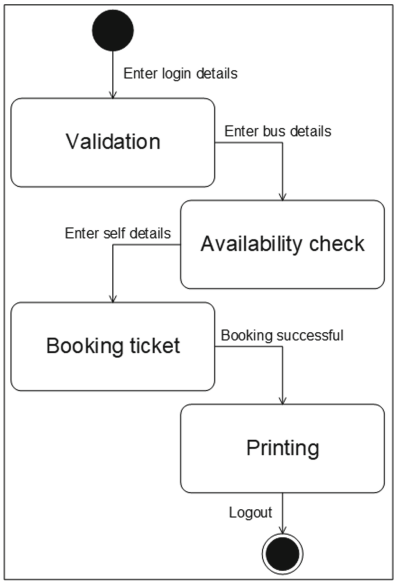

Fig. 5. UML state model of Ticket buying example

things that require coordination, or how the events in a single use case relate to one another, especially, use cases where activities may overlap and require coordination.

Table 5 presents UML Activity model elements generated from the Enterprise model of Ticket buying example. In Enterprise Model all information related with actors, their activities and relationships between these functions is stored. There is one business rule – control node, related with the process beginning, initial node. In this case, there is only one actor – one partition – Client. There are two Client activities before decision node: bus searching and checking tickets availability. In case, there is no available tickets, process finishes unsuccessful with one of final activity nodes. In other case, if there are available tickets, Client books tickets, fills details, submits details, makes payment and prints ticket. This process finishes as successful with second final activity node.

Figure 6 presents UML Activity model of Ticket buying example generated step by step from the Enterprise Model through UML Activity transformation algorithm [10].

Table 5. UML Activity model elements generated from the Enterprise model of Ticket buying example [5, 7, 9]

Enterprise Model element	UML Activity Model element	Ticket buying example	Description
Actor	Partition	Client	There is one partition and all activities are directly related with that actor
Function, Process	Activity	Search bus	There are eight activities directly related with one partition – Client. They represent a parameterized behaviour as coordinated flow of actions
		Check tickets availability	
		Book tickets	
		Fill details	
		Submit details	
		Make payment	
		Print ticket	
		Logout	
Business Rules	Control nodes	Initial node, two activity final nodes, decision node – are there available tickets	There are four control nodes: one node – Initial node in the beginning; one decision node, regarding which process finishes in success or otherwise; two activity final nodes one, in case of successful process, another, in case of unsuccessful process. Basically, control nodes are used to coordinate the flows between other nodes

All four UML dynamic models: Use case, Sequence, State and Activity of one Tickets buying process example are generated form Enterprise model, where sufficient, verified and validated data was stored. These four UML models define same example, but in diverse perspectives by showing different actors activities, states and use cases. Knowledge-based Enterprise model is sufficient storage of data, which is necessary for UML models generation by certain transformation algorithms of each UML model.

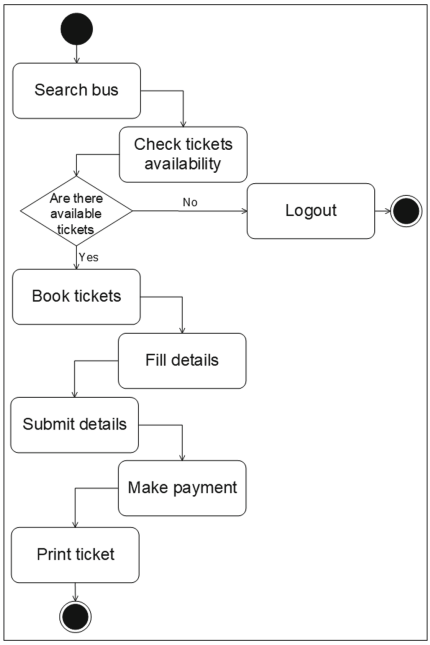

Fig. 6. UML activity model of Ticket buying example

4 Conclusions

The first part of the paper deals with the presentation of the knowledge-based Enterprise model, UML models generation form Enterprise model top level transformation algorithm, which is described step by step and also represents the idea of each UML model transformation algorithm. This part either presents some part of Enterprise model elements and describes their possible content necessary for further research.

The second part presents particular example, which data is stored in knowledge-based Enterprise model and it is used in generation process. There are presented four types of UML dynamic models for this particular example. Each mentioned UML model is generated through certain transformation algorithms.

Each subsection presents different UML dynamic model. All these UML models are generated from the Enterprise model. All information necessary for this generation process is stored in knowledge-based Enterprise model and all UML models elements of the analyzed example also described by their dependency to a certain elements stored in Enterprise model.

The presented example demonstrates that all knowledge stored in Enterprise model is enough for generation process; that Enterprise model elements are sufficient to convey all UML models element in different UML models perspectives. Every element of UML dynamic models can be generated from the Enterprise model and this can implement entire knowledge-based IS development cycle design phase.

References

1. Dunkel, J., Bruns, R.: Model-driven architecture for mobile applications. In: Abramowicz, W. (ed.) BIS 2007. LNCS, vol. 4439, pp. 464–477. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-72035-5_36
2. Eichelberger, H., Eldogan, Y., Schmid, K.A.: Comprehensive Analysis of UML Tools, their Capabilities and Compliance. Software Systems Engineering. Universität Hildesheim, version 2.0 (2011)
3. Gudas, S.: Architecture of knowledge-based enterprise management systems: a control view. In: Proceedings of the 13th World Multiconference on Systemics, Cybernetics and Informatics (WMSCI2009), 10–13 July, Orlando, Florida, USA, vol. III, pp. 161–266 (2009). ISBN - 10: 1-9934272-61-2 (Volume III). ISBN - 13: 978-1-9934272-61-9
4. Gudas, S.: Informacijos sistemų inžinerijos teorijos pagrindai/Fundamentals of information systems engineering theory (Lithuanian). Vilnius University (2012). ISBN 978-609-459-075-7
5. Jacobson, I., Rumbaugh, J., Booch, G.: Unified Modeling Language User Guide, 2nd edn. Addison-Wesley Professional, Boston (2005). ISBN: 0321267974
6. Jenney, J.: Modern methods of systems engineering: with an introduction to pattern and model based methods (2010). ISBN-13:978-1463777357
7. OMG UML: Unified Modeling Language version 2.5.1. Unified Modelling (2019). https://www.omg.org/spec/UML/About-UML/
8. Sajja, P.S., Akerkar, R.: Knowledge-Based systems for development. Adv. Knowl. Syst. Model Appl. Res. **1**, 1–11 (2010)
9. UML diagrams characteristic (2012). www.uml-diagrams.org
10. Veitaite, I., Lopata, A.: Transformation algorithms of knowledge based UML dynamic models generation. In: Abramowicz, W. (ed.) BIS 2017. LNBIP, vol. 303, pp. 59–68. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-69023-0_6
11. Veitaite, I., Lopata, A.: Problem domain knowledge driven generation of uml models. In: Damaševičius, R., Vasiljevienė, G. (eds.) ICIST 2018. CCIS, vol. 920, pp. 178–186. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-99972-2_14
12. Veitaite, I., Lopata, A.: Knowledge-based transformation algorithms of UML dynamic models generation from enterprise model. In: Dzemyda, G., Bernatavičienė, J., Kacprzyk, J. (eds.) Data Science: New Issues, Challenges and Applications. SCI, vol. 869, pp. 43–59. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-39250-5_3