process models in different ways. In commercial systems, the internal representation of models remains hidden for their users and they have no direct access to the data. These systems are focused on analyzing, modeling, implementing and monitoring business processes, but they do not provide the ability to generate new models from existing ones.

III. Data Model for Business Process Generator

In practice is very common to have set of processes, which are arranged in time in some way, i.e. it is not possible to start a process before another is finished. This means that when storing all business processes it is necessary to have a mechanism for presentation and storage of these relationships. Furthermore, together with all the characteristics of the process, it is necessary to store information about the relationships between processes.

The authors choose EPC (Event-driven Process Chain) standard for business process modeling. The EPC standard represents the control-flows of business processes through graphical notations. Over the years, the standard has been developed and established as a powerful tool for modeling, analyzing and transforming business processes. The basic elements of the standard are functions, events and connectors, and connections between them. But not every diagram composed of these elements is a correct EPC diagram. Functions describe activities of processes and events – preconditions and results of functions.

In the EPC standard the events are passive elements and serve as a link between the different functions of the business process (Fig. 1). Each function (process) is preceded by exactly one element, and exactly one element is followed by it, i.e. there is one input and one output point [7]. The authors describe in [8], how to unify presentation of these elements by two types of events: simple and composite. Simple events are single items and composite events combine together several events (possibly also composite) through connector. Each process (function) is preceded by a triggering event and ends with a terminating event, where the triggering and terminating events can also be composite.

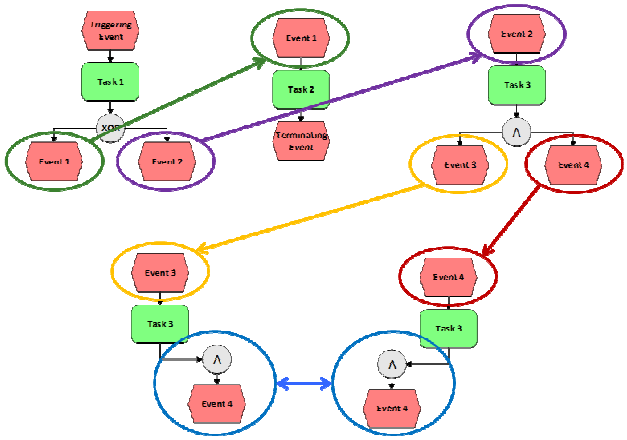

Fig. 1 Connecting processes by EPC events

The composite events are divided into two groups according to the connector: splitting and joining. The composite events are divided into: triggering and terminating depending on whether they precede or follow process. Therefore composite events are four types: terminating splitting (end composite splitting ECS), terminating joining (end composite joining ECJ), triggering splitting (start composite splitting SCS) and triggering joining (start composite joining SCJ) events.

For each process (function), there is exactly one entry and one exit point [7]. These entry and exit points can be events and connectors, where they are defined respectively as just a simple or a composite event (including connector and events connected to it). In the business processes repository for each process data about its entry and exit points is stored in addition to its characteristics.

A connector groups several related to it events, these events can be both simple and composite. Through connectors, for particular process, it is possible to find all triggering and terminating events. A process has only one entry and one exit point, but there may be more than one triggering events and more than one terminating events.

Modeling the database structure for this type of system in which the modeled processes will be stored is of great importance [9, 10].

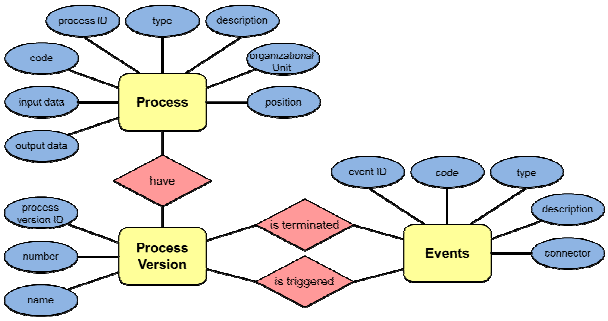

Fig. 2 Entity-relationship model

In Fig. 2 shows entity-relationship model representing the classes to be stored in the database. Each process has several versions, each version being associated with just two events - one initial and one final. It is possible for an event to be started for more than one version of different processes or final for more than one version of different processes. This means that the relations “is terminated”, “is triggered” and “have” are of type “one-to-many”. This model doesn’t include a relation between process and sub-processes. A process may be either a sub-process of a process, or a process that has sub-processes. Only simple processes without children can only play the role of sub-processes. There is also one or more processes that don’t have a parenting process. The relation between composite events and related events is also not represented in the model.

A process may be a sub-process of more than one parent process, when a functionality is used in two different processes. This “many-to-many” relation between a process and its sub-processes is represented in a table InterPP (Fig. 3), where the id number of each process is related to the id numbers of all its sub-processes. This is enough to define a tree structure of a process that includes all of its sub-processes at all levels. The different versions of a process that are stored in the table ProcessVersion may have different start and end events. For this reason, it is also necessary to store information between a process version and each versions of the sub-processes that are included in the

selected version. This “many-to-many” relation is represent by table InterVV.

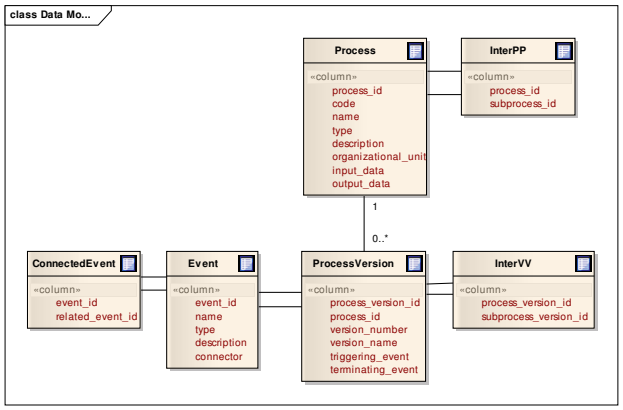

Fig. 3 . Relational database model

An event can be in the composition of more than one composite event. The table Connected_Event represents relation "many-to-many" between a composite event and all its related events, where the id number of each composite event is related to the id numbers of all events associated with it.

The Process, ProcessVersion, and Event classes are presented in separate tables that include all of their attributes. ProcessVersion class also contains key attributes: two for the Event class and one for the Process class respectively for the start and end event of the version and the process itself for the version.

The six tables Process, ProcessVersion, Event, InterPP, InterVV, and Connected_Event cover the main features of process modeling and versioning and form the database schema. Through them, it is possible to present each process as a tree, including all its sub-processes at all levels, as well as a set of sub-processes that form an EPC process diagram. The relations between these six main tables are presented in Fig. 3.

The Organizational Unit and Position attributes match the process executives. The Position describes contractors of a different type according to standard process modeling (EPC). They can be individuals, specialized teams, software applications, and more. If the contractor is not clearly specified, an entire organizational unit may be mentioned as such. It can also be defined when a particular contractor is assigned.

The process data is stored in two tables: Process and ProcessVersion. The common process characteristics (name, description, code, type, position, organizational unit and task) that are the same for all versions of the process are stored in a table Process. The relations between a process and its sub-processes are carried out with a “many-to-many” link, through an InterPP table to itself. The table ProcessVersion stores the version number as well as the start and end events. Thus, for different versions of a process, there can have different start and/or end events.

The Organizational Unit and Position attributes match the process executives. The Position describes contractors of a different type according to standard process modeling (EPC). They can be individuals, specialized teams, software applications, and more. If the contractor is not clearly specified, an entire organizational unit may be mentioned as such. It can also be defined when a particular contractor is assigned.

The process data is stored in two tables: Process and ProcessVersion. The common process characteristics (name, description, code, type, position, organizational unit and task) that are the same for all versions of the process are stored in a table Process. The relations between a process and its sub-processes are carried out with a “many-to-many” link, through an InterPP table to itself. The table ProcessVersion stores the version number as well as the start and end events. Thus, for different versions of a process, there can have different start and/or end events.

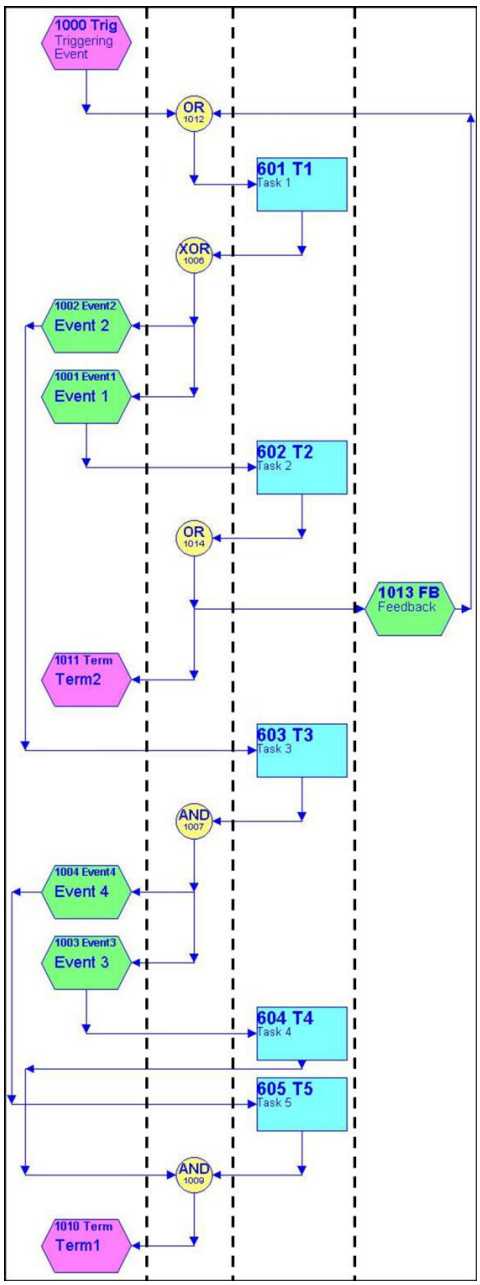

Fig. 4 Generate business process model

It is possible to include additional features that expand the model and make it more complete, but the basic and essential are present in the proposed model.

In the business process repository for each process a set of its sub-processes are stored as described above, which generate a graphical structure of the business process model. Each sub-process is characterized by one triggering and one terminating event. A proposed algorithm [11] receives this data and automatically generate a workflow diagram for the process (Fig. 4). As it has been mentioned, the business process repository contains description of business processes from each level. Thus, the components of the process, its sub-processes stay somewhat independent of each other. This achieves independence between the elements of the process and appears as source for further developments and improvements of these elements without affecting the already established business process models. Changes automatically, if necessary, reflect in the business process model. The algorithm's main task is to sort in such a manner the elements of graph structure so that the image is understandable. The elements of the structure include processes, their triggering and terminating events and connectors. These elements are input data for the algorithm.

The proposed business process model visualization algorithm improves the business process repository's functionality. It can be used to verify that it is possible for a set of business processes to form a new process from a higher level. The user has the ability to easily and quickly compare several business process models generated by the processes stored in the repository. Viewing stored processes is a convenient and easy way to manipulate with the data available in the repository.

The proposed system data model for each business process keeps a history of the changes the process has undergone over time. These changes are expressed in what new versions are created.

IV. Sowtware Architecture for Business process Generator

For business process generator where large amounts of business process data are stored, it is necessary to have access to these data from a centralized location [12]. Therefore, a web-based system has been selected. Web applications are client-server software, where the client is running on a user's browser. This eliminates the software dependency limitation between the web application and the user's operating system. Changes and upgrades to the application will save the inconvenience of replacing an already installed desktop application. It also provides access to shared data from multiple users.

The Model-View-Controller (MVC) architecture design pattern is leading in designing a business process generation system. It focuses on the division of responsibilities into several separate layers. This leads to a clear distinction between business logic, graphical interface and application data. The main advantage of this kind of architecture is that the model is independent of the controller and the view. Which allows the conversion of several different views without the need to change the model.

Although MVC was developed for desktop applications in the 1990s, it is widely accepted as a web application development architecture. There are multiple technology frameworks for web applications that impose MVC pattern. These frameworks differ in their interpretations, mainly in the way they assign MVC responsibilities between the client and the server.

Over the last 5 years, the technology has grown significantly on the client. Technological frames such as Angular, React, EmberJS, JavaScriptMVC, based on Ajax, use MVC components to perform partial logic on the client. Thus, the logic of a web application is distributed over the client and server, thus facilitating server performance and reducing traffic between individual clients and the application server.

Service-Oriented Architecture (SOA) is another architectural template that is followed. In it, application components provide services to other components through a communications protocol, typically over the Internet such as web services. The basic principles of SOA are independence between service provider, product, and implementation technology.

A Resource-Oriented Architecture (ROA), which represents SOA but is closely related to REST (Representational State Transfer) technology, is used in the implementation of the system. The main advantages of ROA are ease of realization, design flexibility and a lightweight approach to things. Thus, there is a disconnection between the technologies used for the client and server side. Once designed and implemented, resources for accessing system data can be accessed from any application, including web-based, mobile, desktop and other business systems. The only limitation to be respected when implementing resource users is the REST service protocol.

When developing the server part, the Data Access Object (DAO) template was used to distinguish low-level access to data on the one hand, and high-level business services on the other. Three are the basic elements of the template: DAO Interface - specifies the operations that are performed on the modeled object; DAO class - implements the DAO interface, responsible for retrieving the data from the data source; Model Object - contains get and set methods for accessing DAO class data.

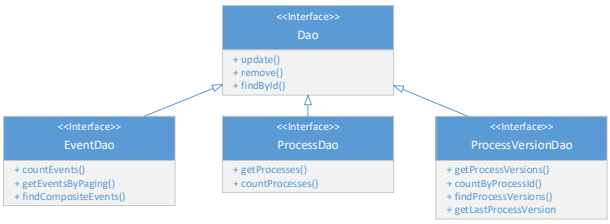

Fig. 5 Inheriting interfaces in the DAO template

A parent Dao interface has been created, which is inherited from all DAO interfaces. It contains the three basic methods for object manipulation: "update" - create and modify, "remove" - delete, "findById" - locate the object by key (Fig. 5). Every class that implements the DAO interface should also implement these three methods. For this reason, an abstract BaseDao base class has been created that implements the Dao interface and is inherited from all DAO classes.

Business logic from the server side is planned to implement as a web service by using REST technology and resource-oriented architecture. ROA is based on the concept of resource use. Each resource is a directly accessible component that is handled by a standardized common interface that makes the use of resources possible.

The client part of the system is running on a web browser. This limits the development capabilities, but with the development of JavaScript in recent years, it is possible to export part of business logic to the user's browser. The Angular technology framework is proposed for the implementation of the system.

The web-based user interface is consistent with browser capabilities. For a system that is used by different users, it's important not to depend on the type of device they use, the operating system, and the browser.

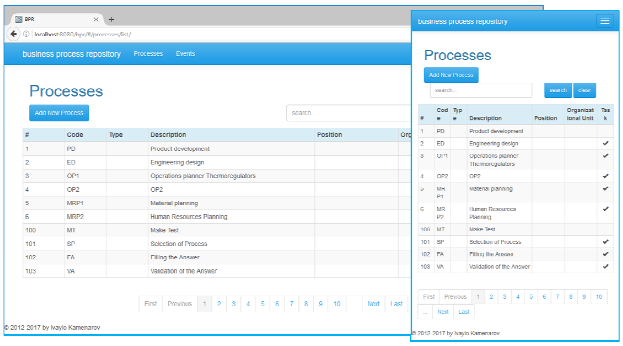

Fig. 6 System view via desktop and mobile device

V. Conclusions

Modern business process modeling environments are constantly evolving and improving. Typically, they are part of a wider range of programming systems.

The present work provides an opportunity to work directly with the internal representation of the process model. It builds on the EPC notation by enabling alignment of the input and output points of each process. Direct access to data makes it easy to define, modify, and generate business processes.

Added functionality to business process repository capabilities that helps business analysts work by facilitating the creation and comparison of different versions of a particular process.

Acknowledgment

This work is supported by the National Scientific Research Fund under the contract ДФНИ - И02/13.

References

[1] G. Atanasova and K. Grigorova, *The Place and the Role of Business Processes Generation in Their Live Cycle*, Proceedings of the union of scientists – Ruse vol. 13, 2016.

[2] W.M.P. van der Aalst, *Process Mining*, Comm. ACM, vol. 55, no. 8, pp. 76–83, 2012.

[3] J. Mendling, *Metrics for Process Models*, Springer, 2008.

[4] W.M.P. van der Aalst, *Business process management: a comprehensive survey*, ISRN Software Engineering,Volume 2013.

[5] G. Atanasova and I. Kamenarov, *Business Process Generation Opportunities*, Proceedings of the union of scientists – Ruse vol. 13, 2016.

[6] Z. Yan, R. Dijkman and P. Grefen, *Business Process Model Repositories - Framework and Survey*, Information and Software Technology vol. 54 no. 4 pp. 380-395 2012.

[7] W.M.P. van der Aalst, *Formalization and Verification of Event-driven Process Chains*, Department of Mathematics and Computing Science, Eindhoven University of Technology, 1997.

[8] I. Kamenarov, *Classification of Events in the EPC Standard*, Information Systems & Grid Technologies, Sofia, Bulgaria, pp. 320-327, 2013.

[9] K. Grigorova and I. Kamenarov, *Object Relational Business Process Repository*, Proceedings of the 13th Int'l Conference on ComputerSystems and Technologies pp. 72-78, 2012.

[10] N. Moghadam H.-Y. Paik, *Bpim: A multi-view model for business process instances*, Proceedings of the 11th Asia-Pacific Conference on Conceptual Modelling (APCCM 2015) vol. 27 pp. 30, 2015.

[11] I. Kamenarov and K. Grigorova, *Visualizing a Business Process Control-Flow*, International Journal of Knowledge Engineering, vol. 2, pp. 64–68, March 2016.

[12] R. Dijkman, M. La Rosa, and H. A. Reijers, *Managing large collections of business process models: current techniques and challenges*, Computers in Industry, vol. 63, no. 2, pp. 91–97, 2012.