AN EXCHANGE LANGUAGE FOR PROCESS MODELLING AND MODEL MANAGEMENT

Huaizhong Li **C. Peng Lam**

School of Computer and Information Science
Edith Cowan University, Perth, WA 6050, Australia
email: {h.li,c.lam@ecu.edu.au}

Abstract: Many development tools and environments have been used in process design to develop models and to perform simulations for processes. These development tools and environments generally use proprietary format to represent the developed models. Frequently, a model developed using a particular tool can not be used in another without tedious modifications. It is also common in process design that the developed models are not systematically managed, hence it is often impossible to reuse an existing model due to difficulties to retrieve knowledge and design artifacts relevant to the model. These problems greatly limit the exchange of the available resources in process design. In this paper, we propose an exchange language for process modeling and model management to promote interoperability of the process models and to enable systematic management of the process models.

Keywords: Process modeling, model management, reuse, XML, data model

1. INTRODUCTION

Modeling and model-based simulation have been extensively used across all development phases of the industrial processes. Process modeling is a critical and enabling technology that conveys knowledge from research to the industry (Krieger, 1995).

Mathematical models are essential for process development, design and operation. Mathematical modeling is characterized by a number of experts using different modeling and simulation environments to build models for varying purposes in process design (Foss *et al.*, 1998). Many commercial and research modeling and simulation environments, for examples, gPROMS, Aspen+, Matlab, Model.LA (Bieszczad, 2000), ModKit (Bogusch *et al.*, 2001) and ModDev (Jensen and Gani, 1999), are widely used in both research society and industry to develop models for processes and alternatively to perform process simulations.

It is well-known that heterogeneous modeling and simulation tools for process modeling exclusively use proprietary formats to represent process models. These modeling and simulation tools are incompatible in general, therefore, the models developed using these tools are not inter-operable. Unfortunately, it is a common practice in process design to employ different tools for different purposes. For example, Aspen+ is commonly used for steady state simulation, gPROMS has advantage in dynamic simulation, while Matlab is perhaps the most used tool for control design and optimization. Thus, additional efforts may have to be devoted to convert a model developed in a tool into a format which is usable in another tool (Schopfer *et al.*, 2000). Therefore, it is advantageous that a tool-independent modeling representation can be developed to facilitate information exchange between the heterogeneous tools.

Model reuse refers to the practice that uses available design artifacts, such as model knowledge and

tool specific implementations, in the development and maintenance of a process. Obviously, model reuse across different modeling and simulation tools can save time and resource for the process design. However, model reuse in the process design life cycle is also difficult. Due to the proprietary formats used to represent the models in the development tools, current model reuse practice is largely confined to within the same development tool only. Furthermore, besides the incompatibility of the modeling and simulation tools which hiders the reusability of the models, currently there is no well-established standard to develop process models. Hence individual engineers and organizations may develop and maintain models in their own ways which may be cryptical to others. A neutral exchange language for process modeling may help standardizing the model development procedure.

In the development life cycle for process design, model knowledge is frequently lost due to improper documentation or lack of systematic management mechanism. It is common that directories in a file system are adopted to serve the purpose of distinguishing different versions in the evolution of the process models. Existing models are either hard to be found, or hard to be used because of insufficient documentation for the models. As a result, repeated modeling is common in process research and industry (Schopfer *et al.*, 2000). From the authors' experience, a systematic model management strategy is helpful in model reuse for process modeling.

Model integration consists of vertical and horizontal integrations. Vertical integration means the integration of the models of comparable granularity but different functionality to form a complete process, while horizontal integration implies the integration of sub-models into a model to expand the functionality of the model. Model integration heavily involves model reuse in different ways. Model can be reused either *a-priori* where the reusable models are designed and implemented with specific consideration for integration, or *a-posteriori* where the models are reused and integrated almost arbitrarily (Marquardt *et al.*, 2000).

It is clear that there are two key issues related to model reuse and model integration:

Model Management Process models have to be managed in a systematic ways to promote reuse and evolution in the process development life cycle

Model Interoperability Process models have to be inter-operable for model reuse and integration between different development environments. This is especially important for *a-posteriori* model integration

One of the best known notions for reuse is 'design once, use many times'. A centralized model repository certainly reduces the risk of repeated design of the same model. A centralized model repository also eases the task of model management. These considerations clearly call for a framework for process modeling which has a centralized data management system, the associated external converters and integrators, and the model exporters. Such framework can act as an integrated part for a heterogeneous modeling and simulation environment to provide interoperability to the other heterogeneous modeling and simulation environments, and to offer the benefits of reusing different versions of the proprietary artifacts from the other heterogeneous modeling and simulation environments.

In the following, we present an exchange language for process modeling and model management. The exchange is defined by a XML schema.

2. AN EXCHANGE LANGUAGE FOR PROCESS MODELING AND MODEL MANAGEMENT

As discussed in the previous section, A centralized modeling framework can reduce the risk of repeated design of the same model and ease the task of model management. Hence, we propose a centralized framework for process modeling and model management. The conceptual architecture of the framework is illustrated in Fig. 1. The framework is currently under development, it has the following functionalities:

- A data exchange server to interact with the various modeling and simulation environments. The data exchange server accepts proprietary models derived from a modeling and simulation source tool, and exports client-specific models to a particular target tool
- An array of converters and integrators to convert the received proprietary models into a neutral exchange format used in the framework, and an array of exporters to transform the models in the exchange format into the proprietary formats which can readily be used in the client tools
- A modeling and management unit which is the core of the framework. This unit performs the following functions:
  - Manage model repositories to store and to retrieve the transformed models from the converters and integrators
  - Provide an on-line modeling tool as a virtual development environment for authenticated engineers to model processes and to store the derived models in the repositories

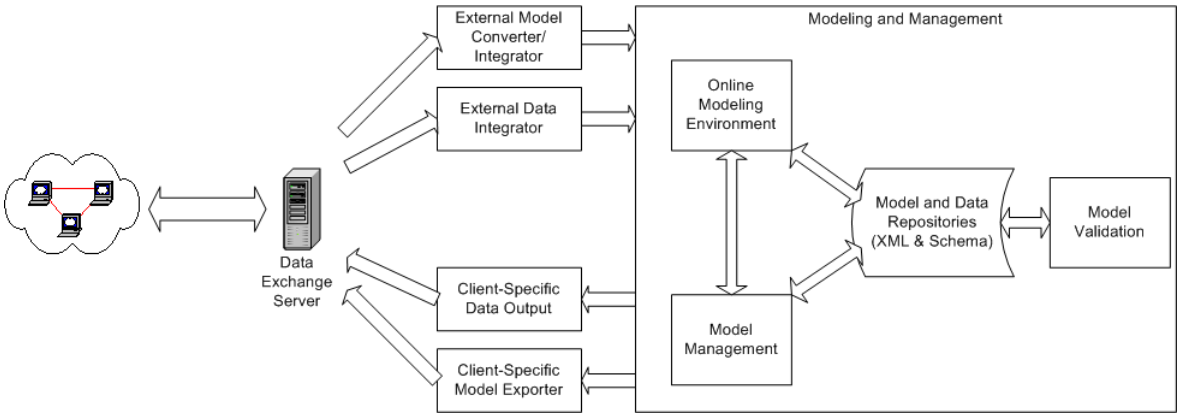

Fig. 1. Conceptual Process Modelling and Model Management Architecture

· Provide model management to the modeling artifacts.

To promote model interoperability and to enable model management, it is advantageous that the models used inside the proposed framework are represented using a neutral exchange language. One of the prominent possibilities for a neutral exchange language is XML. As stated by Wedel (Wedel, 2002), XML has great advantages to represent process models which can be easily represented using the XML flexible meta model. Many existing tools can be used and integrated in the process development environments to process XML data. XML representation of the models also enables systematic model management. Models represented in XML format can be managed and validated using a corresponding XML schema for process modeling.

In the following, we present an exchange language for process modeling and model management. The proposed exchange language is one of the key elements for the modeling and management unit.

We have developed a XML schema to define an exchange language for representation of the process models. As illustrated in Fig. 2, a process model (we call it 'process' in the schema) comprises three parts, namely, the model parts ('model') used to integrate the process, the connections between the models, and the elementary version control mechanism for the process. Due to space limitation, we only present the coarse skeleton of the exchange language in this paper. The actual schema, the relevant documentation, and some sample process XML instances can be obtained from the authors.

Connections between models refer to the coupling between the input ports and the output ports of the relevant models. As shown in Fig. 3, a connection has a source *connectFrom* from an output port of a model, and a target *connectTo* to an input port of a model. The nature of the connection is defined by *link* which contains a list

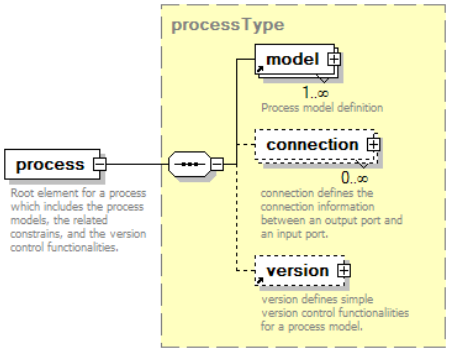

Fig. 2. Process Structure

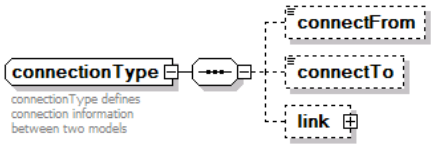

Fig. 3. Connection Structure

of the allowable connection types, for example, a *link* can be stream or liquid for chemical processes.

A process consists of one or more integrated model parts, referred to as horizontal model integration in the previous section. For example, a reaction/separation chemical process in (Lee *et al.*, 2000) is made up of two systems namely a reactor system and a separation system, the reactor system is a continuous stirred tank reactor (CSTR), and the separation system consists of an extractor unit, a flash unit and a distillation column. The CSTR, the extractor unit, the flash unit and the distillation column are all model parts in the developed schema. A model can be represented in a structure shown in Fig. 4. Specifically, a model has a name and a unique ID which can be used to assist in model management. The model can be a continuous, or a discrete, or a hybrid model indicated by *domain*. A model has one or many input ports and output ports, each port is described by a *portName*, a unique *PortID*, and a unit for the port. Such description eliminates any confusion in the connection of ports.

A model can also have sub-models, as illustrated in Fig. 4. The inclusion of the sub-models implements the vertical integration concept discussed in the previous section.

A process model has mathematical concepts like variables, parameters, and equations. The behavior of a model is determined by its variables which are constrained by mathematical equations. The variables, parameters and equations for a model are defined in our schema as *variables* and *equation*. Each variable or parameter has a name, a unique ID, the default value, its unit, and the constraint for the variable or parameter. the definition of *variables* enables the handling of general variables and parameters in various processes.

We have created an equation structure, shown in Fig. 5, to describe the equations for a model. A equation in the exchange language comprises the left-hand-side *equationLHS* and the right-hand-side *equationRHS*, *relation* is defined to describe the relationship between the two sides. Though the term 'equation' has been used in the exchange language, the equation type in the schema is capable to define interim variables, mathematic terms, equations or inequalities.

It should be mentioned that there are alternative ways to represent equations. In CapeML (Wedel, 2002), MathML is adopted for this purpose. However, as indicated in (Wedel, 2002), a MathML-based solution adds difficulties to the implementation of the tools, and the mathematical viewpoint of MathML is counterintuitive to many simulation tools. Unlike MathML, the equation definition in our schema originates from process viewpoint. Namely, the definition of an equation in the schema is designed to be as similar as possible to the one used in process research and industries. Through repeatable use of *operator*, *matrix* and *variableArray*, complex equations in the processes can be straightforwardly represented using the proposed schema.

To assist model management, the elementary version control mechanism has been implemented in the schema. A particular version of a process or a model part consists of the version ID, the version object and the version resource, as shown in Fig. 6. Creators of the model, checkin rules, checkout rules, version history and baselines for the version objects form the content of the resource. The version mechanism implemented in our schema is a simplified variant of many popular version control systems. However, we believe that this simplified version control mechanism is appropriate, and it should be also sufficient for the process model management.

Before we conclude the paper, we need to justify the exchange language proposed in this paper. To our knowledge, two exchange languages have been proposed for process modeling. One of the languages is Modelica (Modelica, 2002). Modelica can comprehensively represent models. However, as pointed out in (Wedel, 2002), Modelica is easy for human to read, but more difficult to be parsed by a software program. Unlike Modelica, our exchange language shares commonality with CapeML (Wedel, 2002) on the ground that the purpose of the language is to promote model reuse and integration for different modeling and simulation tools, hence it is more important if the model representation can be easily handled by computers.

Another exchange language is CapeML (Wedel, 2002) for CAPE-OPEN as previously discussed. CapeML is implemented using XML DTD, while our exchange language is defined using W3C schema. There is rich literature available about the differences between DTD and schema, or about the advantage of the W3C schema, hence we will not argue the benefits of using the W3C schema here. One thing that we have to mention is that it is possible to use reasoning in W3C schema. We plan to use reasoning in the next draft of the exchange language to further improve model exchange capability.

Both Modelica and CapeML do not concern about model management which is crucial for model reuse and model integration. As illustrated above, model management is part of the developed exchange language in this paper. Prototype research has also been carried out to manage the process models using the developed exchange language. The details will be reported elsewhere.

3. CONCLUSION

In this paper, we have proposed and implemented an exchange language for process modeling and model management. The exchange language is formulated in XML format and is defined by a XML schema to facilitate easy integration with existing tools and environments for process modeling. The developed exchanged language can be used to promote interoperability of process models and to enable systematic management of process models.

REFERENCES

Banares-Alcántara, R. (1995). Design support system for process engineering I. requirements and proposed solutions for a design process representation. *Computer & Chemical Engineering* **19**, 267–277.

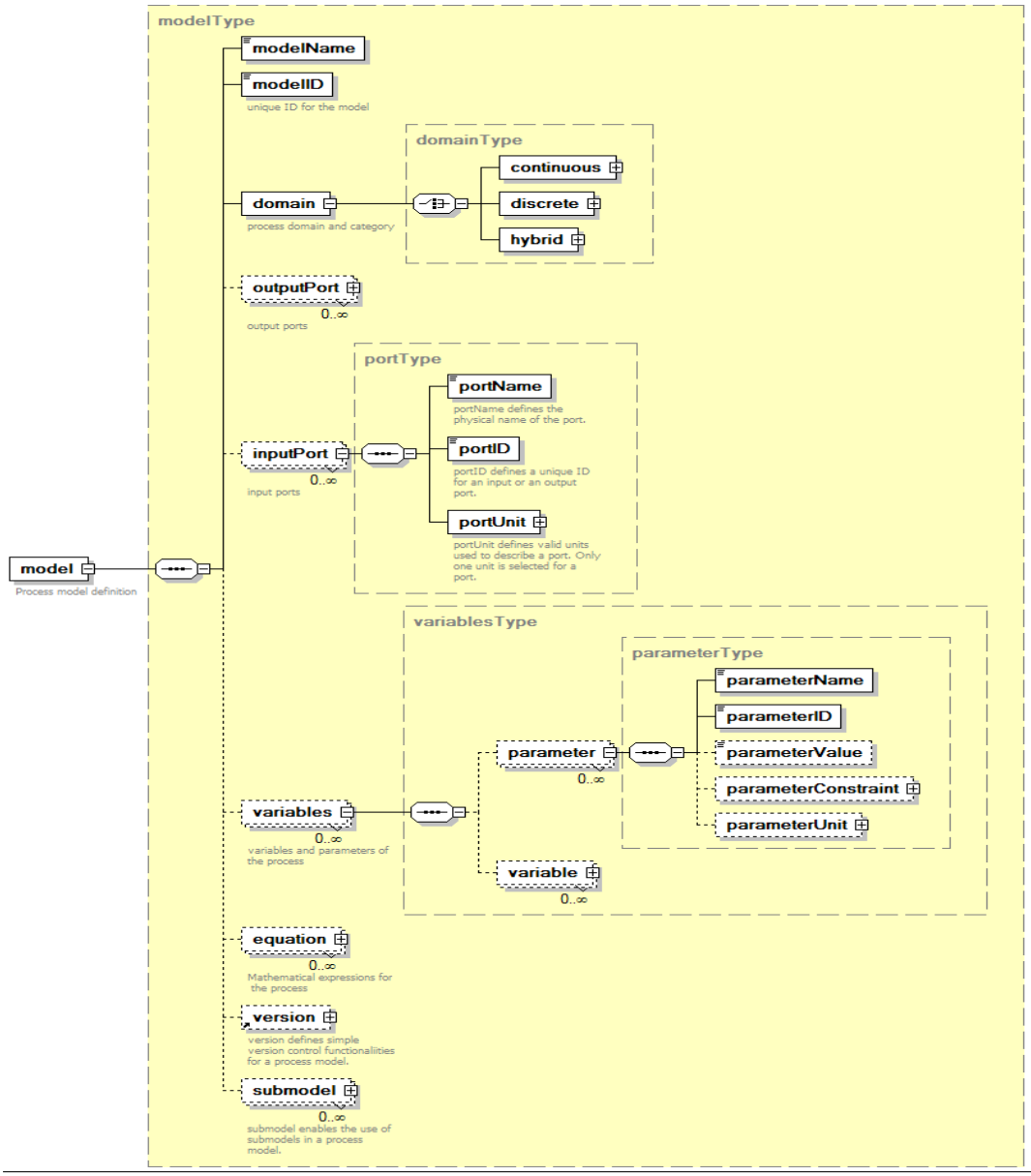

Fig. 4. Model Structure

Bieszczad, J. (2000). A framework for the language and logic of computer aided phenomena-based process modeling. PhD thesis. Massachusetts Institute of Technology.

Bogusch, R., B. Lohmann and W. Marquardt (2001). Computer-aided process modeling with modkit. *Computers and Chemical Engineering* **25**, 963–995.

Foss, B., B. Lohmann and W. Marquardt (1998). A field study of chemical process modeling. *J. Process Control* **8**, 325–337.

Jensen, A.K. and R. Gani (1999). A computer-aided modelling system. *Computers and Chemical Engineering* **23**, 673–678.

Krieger, J.H. (1995). Process simulation seen as pivotal in corporate information flow. *Chemical & Engineering News* **73**(13), 50–61.

Lee, P.L., H. Li and I. T. Cameron (2000). Decentralized control design for nonlinear multi-unit plants: a gap metric approach. *Chemical Engineering Science* **55**(18), 3743–3758.

Marquardt, W., L.v. Wedel and B. Bayer (2000). Perspectives on lifecycle process modeling. In: *Foundations of computer-aided process design* (M.F. Malone, J.A. Trainham and B. Carnahan, Eds.). Vol. 96. pp. 192–214. AIChE Symp. Series 323.

Modelica (2002). *Modelica: A unified object-oriented language for physical systems modeling*. http://www.Modelica.org/.

Schopfer, G., L.v. Wedel and W. Marquardt (2000). An environment architecture to support modeling and simulation in the process design lifecycle. In: *AIChe Annual Meeting*. Los Angeles.

Wedel, L.v. (2002). CapeML - A model exchange language for chemical process engineering. Technical report. RWTH Aachen.

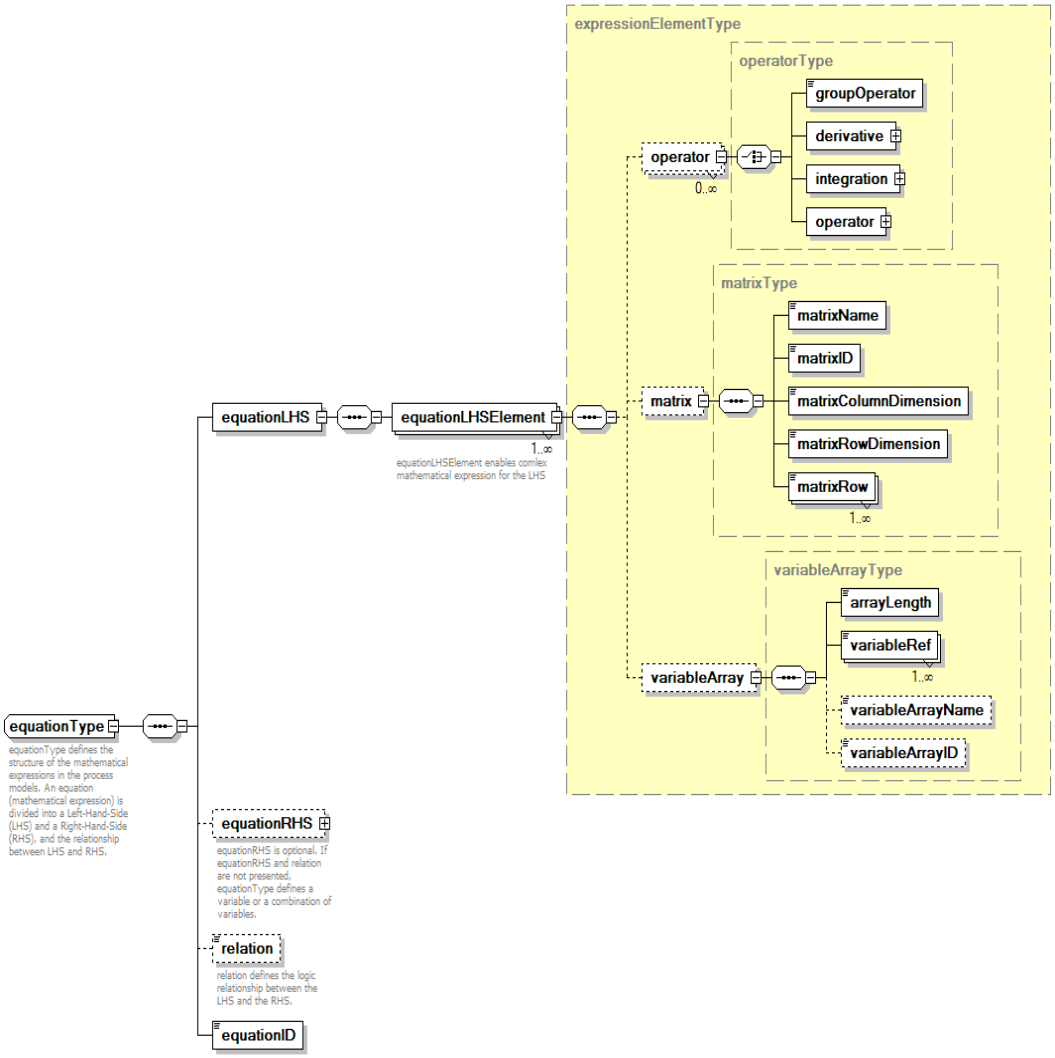

Fig. 5. Equation Structure

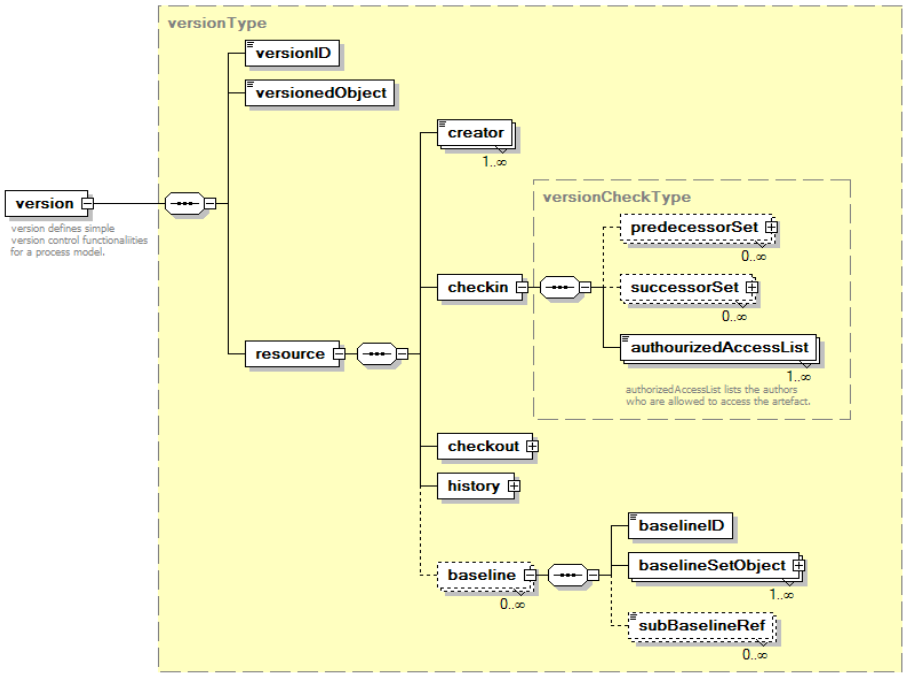

Fig. 6. Version Structure