# Analysis of Data Formats of Visual Models of System Design in Conditions of Synchronous Design Technologies

Andrey Vlasov, Vadim Shakhnov and Ludmila Juravleva

*Bauman Moscow State Technical University, 5/1 Baumanskaya 2-ya str., Moscow, 105005, Russia*

**Abstract**

The paper considers the methodology of synchronizing visual models of the system project through knowledge XML templates. The object of study is the means of formal description, storage, and transfer of design information about complex systems in the context of digital industry transformation. A new concept of system analysis of complex systems and the generalized design environment which implements it are analyzed. The study suggests a concept of cognitive metalanguage VI-XML implementing metaphors of cognitive visualization, taking into account the synchronization of different abstraction levels. This allows the encapsulation of visual metaphors of different abstraction levels into a single closed multi-level system model. Methods of establishing correspondence between the elements of heterogeneous diagrams of visual models are proposed, and the mechanism of translating component representation of the diagram to VI-XML and vice versa based on knowledge paradigms is substantiated. The proposed method is based on the integrated adaptive visual design environment, which analyzes components of visual-conceptual, structural-functional, and object models of various notations in a single decision tree. The method provides an efficient and adequate representation of the subject area knowledge in multi-level visual models and implementation of management tools in terms of their evolutionary development and adaptation to the demands of the system project.

**Keywords**

Design, software system, quality, software engineering, testing

## 1. Introduction

The paper is devoted to analyzing formalized data representation of visual models of the system design in the implementation of synchronous design technologies [1]. Ensuring project data synchronization within a single system project in the context of digital industry transformation is an important task [2]. A modern digital production system is an integrated system covering all components of the product life cycle [3]. Each life cycle component is reflected in a single synchronous model of the production system, which provides information transparency, reproducibility, manageability, and reliability of the model as a whole [4].

Complex systems consist of a large number of parts and a large number of connections between them [5]. The greater the number of connections, the more difficult it is for research to achieve the final result, i.e. the derivation of patterns of the object functioning to ensure the effectiveness of activities. At the same time, thecomplex system is considered to be the one in which the model does not have enough information to effectively manage this system [6–8].

The use of visual methods and system design tools as the main instruments for generating, storing, and processing knowledge about the system is now a de-facto standard. Various visual modeling notations are used to implement visual models of complex systems [1–3, 9, 10]. The main problem of

SibDATA 2021: The 2nd Siberian Scientific Workshop on Data Analysis Technologies with Applications 2021, June 25, 2021, Krasnoyarsk, Russia
EMAIL: vlasov.a.i@ymservices.ru (A. Vlasov)
ORCID: 0000-0001-5581-4982 (A. Vlasov); 0000-0003-4013-1905 (V. Shakhnov); 0000-0003-1994-8830 (L. Juravleva)

their joint use within the framework of a single system project is the heterogeneity of the presentation of these models [11, 12] and their storage formats [13].

The objective of this study is a comprehensive solution to formalization and data processing of the visual models of complex systems through a single meta-description format based on a hierarchically linked evolutionary component structure (an ordered set of meta-objects and interrelations).

The paper suggests an approach to the knowledge description of the visual models using concepts and judgments as meta-documents VI-XML (Visual Intelligence XML, a universal knowledge description language based on XML, which describes knowledge using concepts and judgments).

The proposed methodology addresses the main problems of systems engineering: cognitiveness (difficulty in identifying concepts when moving from one level of abstraction to another), convergence (blurring of boundaries between individual levels of abstraction), and encapsulation (fragmentation of classical visual languages and isolation of available tools) [3].

The proposed concept of visual design based on knowledge structures of complex systems, presented in the form of a matrix of the model component interaction, combined by a single component methodology implemented considering socio-productive systems, provides the unity of the system design regardless of the notations used. Based on the requirements of consistency, truthfulness, as well as complexity and multivariant representation of knowledge about components, objects, and processes, a unified data format of visual models and integrated visual design environment, which combines the basic tools for formalization, storage, and processing of knowledge of complex systems represented as a hexagonal dense-packed knowledge model, is implemented [3].

## 2. Literature review

Visualization of the design of complex systems is understood as a methodology for the formal description of the system design process and a finished project in the form of a universal calligrapher with the possibility of hierarchical decomposition of project elements. “Visual Modeling” implies the use of visual expressions (graphs, drawings, pictograms, tables), which are elements of a graphic language in the design process. This term refers to the systems which allow representing knowledge about a design object in a two (or more) dimensional form [8, 13, 14].

Visual design methods are based on visual languages and methods for developing models based on them. A visual language is a language which systematically uses visual meanings to describe its main objects in textual and graphical notation.

The traditional classification of visual design languages divides them into [6, 8, 11, 13]:

- diagrammatic languages that serve for the use of diagrams and charts, usually previously used for “paper” design;
- iconic languages that use natural images (pictograms) to represent objects and actions;
- form languages, relying on the use of forms and document forms in design.

The main directions in the field of advanced visual languages are the following:

- creating visual languages to support cognitive design;
- creating visual languages to support functional design;
- creating visual languages to support object-oriented design;
- creating universal visual languages for group (parallel) design.

In general, the object of representation using visual design methods is knowledge and data on a specific subject area, describing the projected complex system in a proper form.

One of the basic visual design elements is the orientation to visual metaphors which are understood as an associative technique which creates a certain visual series following the concepts and objects of this applied field. The choice of a metaphor is the choice of a sign system to be used in a given visual notation, and for each notation, it is different. The purpose of the visual metaphor is to create visual images that correspond to operations on model objects. The role of the visual metaphor is to provide an understanding of the displayed entities of the application area and to create new entities based on the internal logic of the metaphor itself. This largely determines the global problem of visual methods; in different notations the same metaphors carry different meanings. The global standardization of visualization metaphors allows creating arrangements so that certain objects are

depicted in a certain way, thereby engaging the visual channel in the design and development as well as in the transfer of knowledge about the systems being built and already created and operating ones.

The methods of using visual modeling prescribe rules for using visual languages to solve certain problems. The methods are implemented in software tools which make it convenient to work with visual languages and use one or another application method. Such programs primarily include graphical editors and model validation tools, final code generators based on diagrams, etc.

There are many methods of visual design and modeling [13–22], but all of them have a significant disadvantage in terms of narrow orientation to a particular stage of design (abstract, conceptual, structural, object, information, etc.). There are almost no comprehensive methods which allow the migration of knowledge from one method to another through a generalized description.

Different visual methods differ in the composition and nature of the models developed during the project and in the approaches to their formalization [22–24]. At the same time, an important task is to choose an effective method (format) for presenting information whichwould ensure the safety of information about the systems under consideration and the portability (export/import) of this information. Storing information in a database does not provide explicit portability on physical media and requires knowledge of data manipulation languages (DML) to extract information. Binary files and specialized formats cannot be read without the appropriate software or require knowledge of special algorithms for writing/reading them. The text-based open presentation of information is devoid of such disadvantages [25–27].

The main open data storage formats are [6, 22]: UFF (Universal File Format) which a text or binary format used for the exchange of test and analytical data; DITA (Darwin Information Typing Architecture) which is an XML-based standard focusing on supporting the entire development, release, and delivery cycle of technical information; SAT (Standard ACIS Text) which is a ACIS text format, whose files in the SAT format are available for viewing in any text editor; CSV (Comma Separated Values); RTF (Rich Text Format); XML (eXtensible Markup Language).

CSV is a text format designed to represent tabular data. Each line of the file is a row of the table. The values of individual columns are separated by a delimiter, for example, a comma (,) or a semicolon (;). The character of the separator used depends on the system settings. In the US, it is a comma, and in Russia, it is a semicolon since the comma is used for fractional numbers (unlike in the US, where it is a dot). The values containing reserved characters, such as comma, semicolon, or newline, are framed by a double quote ("); if the value contains quotation marks, they are represented in the file as two consecutive quotation marks.

RTF is a proprietary cross-platform format for storing marked-up text documents, which was proposed by Microsoft and Adobe as a meta tag format in 1982. Since then, the format specification has changed several times. All modern word processors support RTF documents.

XML is recommended by the W3C Consortium (W3C, http://www.w3c.org), it is a markup language designed for storing structured data and exchanging information between programs, which is a simplified subset of SGML (Standard Generalized Markup Language). The results of the comparison of the listed formats are summarized in Table 1.

The XML format has several advantages over the RTF or CSV formats.

The only drawback of XML is its redundant syntax, but this is balanced by its advantages. XML is preferred because it represents tabular data (such as relational data from a database or large tables) and pseudo-structured data (such as Web pages or full-text documents). XML is a set of general syntactic rules for describing document structures using markup tags (the descriptors taken in the angle brackets '<' and '>'). However, the tags are not initially defined and must be created by the developer. Hence, one of the main advantages of XML is the unlimited extension. This has led to the creation of a huge number of subsets of XML, such as XHTML (a family of web page markup languages, [http://www.w3.org/TR/xhtml11/]), XMCD (this markup language is used to describe data in the Mathcad software package, [http://www.ptc.com/products/mathcad/]) [15]. The comparative characteristics of XML, RTF, and CSV are presented in Table 1.

**Table 1**
Comparison of the XML, RTF, and CSV data formats

| Criteria | XML | RTF | CSV |
|---|---|---|---|
| Platform independent | Yes | Yes | Yes |
| Presence of parser implementations for all modern programming languages | Yes | No | No |
| Suitable for describing all types of data | Yes<br>but network models are difficult to describe | No<br>text only | No<br>tables only |
| Open, i.e., it is not protected by patents and can be implemented without any financial deductions to third parties | Yes | No | Yes |
| Self-documenting format | Yes<br>describes the structure and names of the fields, as well as the values of the fields | No | No |
| Redundant syntax | Yes<br>one structure can be described in several equal ways | No | No |
| Extensibility | Yes<br>allows defining and using own dictionaries | No | No |

Therefore, the task of global synchronization of visual models is reduced to the choice of a single extensible and complementary format for their representation.

Modifications of the formats based on the XML core are increasingly used in various visual models [15–18]. However, developers try to generate XML templates for a specific model. It is necessary to move away from the syntax of a particular model and build schemes on more general knowledge categories to make them universal. It is necessary to create a universal knowledge template (scheme) which will describe all the variety of abstractions of the subject area. At the same time, the only drawback of such an XML scheme is the redundant syntax, but this is compensated by its advantages. XML is preferred because it represents tabular data (such as relational data from a database or large tables) and pseudo-structured data (such as Web pages or full-text documents).

## 3. Methods: XML schemes of visual models of the system project and their synchronization

Design technology is defined as a combination of three components:

- a step-by-step procedure which determines the sequence of process design operations;
- criteria and rules used to evaluate the results of process operations;
- notations (graphical and textual means) used to describe the designed system.

The process instructions, which form the main content of the technology, should include the description of the sequence of technological operations, conditions, which predetermine which of the operations will be performed, and descriptions of the operations themselves. For example, in the system project for developing a digital element base, the following design levels can be distinguished: conceptual, system, behavioral, register transfer, logic, and geometric levels (Figure 1) [20].

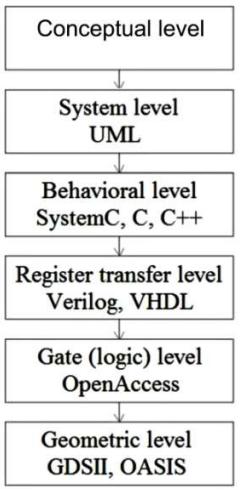


**Figure 1**: Design levels of the digital element base

The initiating stage is a conceptual-abstract visualization [22–25]. The least formalized methods are used here since their main task is to make modeling as simple as possible for specialists in subject areas to maximize their creative abilities. The use of abstract modeling in the early stages allows moving to business process modeling with the least cost (i.e., already at the initial stage in the process of studying the subject area, even before the start of formalization, a system of concepts and decomposition schemes is set which are adequate to the subject area understudy).

The next stage is related to the development of functional models from the viewpoint of structural-functional and process approaches [25–27]. For modeling, it is particularly important to increase the level of the model adequacy; for this purpose, the cycle introduces such stages as simulation modeling and optimization of business processes by integral and differential criteria, which evaluate the results of decisions based on the models built.

At the system stage, an object-oriented model is implemented, which is built with the help of the rational unified process, thus allowing one to create a framework of the information infrastructure itself, which is further developed [20].

Consider the problems of synchronizing visual models at the conceptual, functional, and system levels.

The universal language for describing visual models is based on fundamental knowledge paradigms. Theoretical knowledge is presented in basic semantic forms: concepts, classification of concepts, judgments (statements), conclusions, dependences, laws, principles; methods, methodologies, and technologies. Among these, concepts, judgments, and inferences are basic mental operations [12]. The objective is to create a universal format for the representation of visual models, which will ensure the safety and portability of the parameters of visual models of different levels of abstraction.

XML is preferred for pre-existing data formats since XML can easily represent tabular data (such as relational data from a database or large tables) and pseudo-structured data (such as Web pages or business documents). This has led to the widespread adoption of XML as a language for information exchange.

XML is a set of general syntactic rules describing document structures using markup tags (descriptors taken in the angle brackets '<' and '>'). However, the tags are not initially defined and must be created by the developer. Hence, one of the main properties of XML is in the fact that it is infinitely extensible. This has led to the creation of a vast number of subsets of XML, such as XHTML (a family of web page markup languages, [http://www.w3.org/TR/xhtml11/]), XMCD (this markup language is used to describe data in the Mathcad software package, [http://www.ptc.com/products/mathcad/]). As a subset of XML, the VI-XML language for describing knowledge about complex technical systems will also be developed.

The extensibility of XML manifests itself in many ways. First of all, unlike HTML, XML does not have a fixed dictionary. With XML, everyone can define special dictionaries for specific applications or different industries. Second, applications which process or use the XML formats are more resistant

to changes in the XML structure; they offer the applications that use other formats. For example, an application which depends on processing a <Customer> element with the customer-id attribute should not typically be interrupted if another attribute, such as last-purchase-date, has been added to the <Customer> element. This flexibility is unusual for other data formats and it is a significant advantage of using XML.

Since each XML subset can have its system of attributes, tags, and nesting rules, a means is needed to check (validate) whether a given document belongs to a particular subset. In other words, a description of the rules that the XML document must obey is required. There are several languages which provide the ability to create such a description (scheme): DTD (Document Type Definition) [http://www.w3.org/XML/1998/06/xmlspec-report-19980910.htm], XDR (External Data Representation) [http://www.tools.ietf.org/html/rfc4506], RELAX NG (regular language for XML Next Generation) [http://www.relaxng.org].

The classic and proven solution is DTD, and the most promising is RELAX NG. However, XSD is by far the most popular XML description format (see the link to the specification below). It is designed so that it can be used in creating software for processing XML documents. Java development tools and .NET have built-in tools for the analysis of XSD. Besides, only XSD has the status of the W3C recommendation.

Creating rules for describing concepts and judgments will allow creating a language for describing any knowledge. Let us develop such a language as a subset of XML and call it I-XML (Intelligence XML). XSD schemes for I-XML (ixml.xsd) describe an XML document with the root element “knowledge”, which has two children representing a list of ideas and a list of opinions. The ideas and opinions are complex types (“ideaType” and “opinionType”, respectively) inherited from the “rootKnowngeElement” type. The knowledge root element contains one idea and opinion element each, and they contain any number of the idea or opinion elements, respectively. After analyzing the structure of IXML, the following conclusion can be drawn: this language effectively describes the knowledge but does not fully fulfill their visual representation tasks. Visually, an I-XML document can only be represented as a tree structure. It is necessary to modify I-XML so that the contained information can be represented in graphical notations used in the design routes of technical systems. The proposed language is called VI-XML-Visual Intelligence XML.

Since VI-XML is extensible and can be used as a subset of IXML to describe any graph model, it can describe any number of visual languages. The only condition is to increase the number of tags used in the language.

## 4. Results and discussion

The present study considers the features of the implementation of various visual models in common visual notations employing VI-XML. For each notation, a correspondence matrix is compiled: notation element-idea/opinion-semantic tag VI-XML. In Figure 2A, an example of the abbreviated VI-XML description is provided to explain the use of semantic IDEF0 tags.

From the perspective of VI-XML, the IDEF1x notation is a collection of entities with attributes connected by three types of relationships: one to one, many to many, and one to many. All this already includes the description of the class diagram. Attributes – fields, entities – classes, relationships – associations. In the software development theory, this relationship is called ORM – Object-Relational Mapping. Thus, the IDEF1x diagrams as a subset of the class diagram are considered. This provides a high degree of integration and parameterization between them. To indicate to the VI-XML handler that a class is also an entity and should accordingly be displayed on the IDEF1x diagram, the “type” attribute in the “class” tag is introduced. To link between the IDEF0 and IDEF1X diagrams, a new proposition must be entered in the “value” field, where the ID of one of the fields is used as a predicate. VIXML models can be built without any connection between the descriptions in IDEF0 and idef1x. However, it is the possibility of their connection that can make VI-XML a very convenient tool.

```
<?xml version="1.0"?>
<knowledge>
  <ideas>
    <activity>
      <id> object_A</id>
      <head> name_object_A </head>
      <level>0</level>
    </activity>
    <arrowData>
      <id>equipment</id>
      <head> equipment </head>
    </arrowData>
            ...
  </ideas>
  <opinions>
    <control>
      <id> object_B </id>
      <subject> name_object_A </subject>
      <predicate>equipment</predicate>
    </control>
            ...
  </opinions>
</knowledge>
```

**Figure 2**: Listing a VI-XML representation of a part of the IDEF0 hierarchical diagram

Summarizing all the above mentioned, a VI-XML XSD scheme can be created (Figures 3, 4). In the above diagram, the description of only four selected visual notations considered and the relationships between them are given as examples. It can be easily extended in a similar way to support other necessary visual notations.

```
<?xml version="2.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">

  <xsd:element name="knowledge">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="ideas" type="ideasType"/>
        <xsd:element name="opinions" type="opinionsType"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>

  <xsd:complexType name="ideasType">
    <xsd:sequence>
      <xsd:element name="arrowData" type="complexIdea" maxOccurs="unbounded"
                   minOccurs="0"/>
      <xsd:element name="class" type="complexIdea" maxOccurs="unbounded"
                   minOccurs="0"/>
      <xsd:element name="activity" type="leveledIdea" maxOccurs="unbounded"
                   minOccurs="0"/>
      <xsd:element name="field" type="ideaType" maxOccurs="unbounded"
                   minOccurs="0"/>
      <xsd:element name="method" type="ideaType" maxOccurs="unbounded"
                   minOccurs="0"/>
      <xsd:element name="actor" type="ideaType" maxOccurs="unbounded"
                   minOccurs="0"/>
      <xsd:element name="useCase" type="ideaType" maxOccurs="unbounded"
                   minOccurs="0"/>
      <xsd:element name="entity" type="ideaType" maxOccurs="unbounded"
                   minOccurs="0"/>

    </xsd:sequence>
  </xsd:complexType>

  <xsd:complexType name="ideaType">
    <xsd:complexContent>
      <xsd:extension base="rootKnowledgeElement">
        <xsd:sequence>
          <xsd:element name="head" type="xsd:normalizedString"/>
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>

  <xsd:complexType name="opinionsType">
    <xsd:sequence>
      <xsd:element name="input" type="opinionType" maxOccurs="unbounded"
                   minOccurs="0"/>
      <xsd:element name="output" type="opinionType" maxOccurs="unbounded"
                   minOccurs="0"/>
      <xsd:element name="instruction" type="opinionType"
                   maxOccurs="unbounded" minOccurs="0"/>
      <xsd:element name="mechanism" type="opinionType" maxOccurs="unbounded"
                   minOccurs="0"/>
      <xsd:element name="activityStructure" type="opinionType"
                   maxOccurs="unbounded" minOccurs="0"/>
      <xsd:element name="association" type="opinionType"
                   maxOccurs="unbounded" minOccurs="0"/>
      <xsd:element name="composition" type="opinionType"
                   maxOccurs="unbounded" minOccurs="0"/>
      <xsd:element name="aggregation" type="opinionType"
                   maxOccurs="unbounded" minOccurs="0"/>
      <xsd:element name="dependency" type="opinionType" maxOccurs="unbounded"
                   minOccurs="0"/>
      <xsd:element name="generalization" type="opinionType"
                   maxOccurs="unbounded" minOccurs="0"/>
      <xsd:element name="signature" type="opinionType" maxOccurs="unbounded"
                   minOccurs="0"/>
      <xsd:element name="onetomany" type="opinionType" maxOccurs="unbounded"
                   minOccurs="0"/>
      <xsd:element name="manytomany" type="opinionType" maxOccurs="unbounded"
                   minOccurs="0"/>
    </xsd:sequence>
  </xsd:complexType>

```

**Figure 3**: Part of the XSD scheme for VI-XML (vixml.xsd)

```
75    <xsd:complexType name="opinionType">
76       <xsd:complexContent>
77         <xsd:extension base="rootKnowledgeElement">
78           <xsd:sequence>
79             <xsd:element name="subject" type="xsd:IDREF"/>
80             <xsd:element name="predicate" type="xsd:IDREF"
81    maxOccurs="unbounded"/>
82           </xsd:sequence>
83         </xsd:extension>
84       </xsd:complexContent>
85     </xsd:complexType>
86
87     <xsd:complexType name="rootKnowledgeElement">
88       <xsd:sequence>
89         <xsd:element name="id" type="xsd:ID"/>
90       </xsd:sequence>
91     </xsd:complexType>
92
93     <xsd:complexType name="complexIdea">
94       <xsd:complexContent>
95         <xsd:restriction base="ideaType">
96           <xsd:attribute name="type" type="elementWithType" use="reqiured">
97         </xsd:restriction>
98       </xsd:complexContent>
99     </xsd:complexType>
100
101    <xsd:complexType name="leveledIdea">
102      <xsd:complexContent>
103        <xsd:restriction base="ideaType">
104          <xsd:sequence>
105            <xsd:element name="level" type="xsd:positiveInteger"/ >
106          </xsd:sequence>
107        </xsd:restriction>
108      </xsd:complexContent>
109    </xsd:complexType>
110
111    <xsd:simpleType name="elementWithType">
112      <xsd:restriction base="xsd:string">
113        <xsd:enumeration value="entity"/>
114        <xsd:enumeration value="actor"/>
115        <xsd:enumeration value="fieldValue"/>
116        <xsd:enumeration value="none"/>
117      </xsd:restriction>
118    </xsd:simpleType>
119
120  </xsd:schema>
```

**Figure 4**: Final part of the XSD scheme for VI-XML (vixml.xsd)

Summing up, the Root element knowledge contains one element ideas and opinions, and they can, respectively, include any number of elements with the ideaType or opinionType. The idea and opinions tags are inherited from rootKnowledgeElement, thus, they both have the id attribute. The ideaType is the base type for concepts. Tags that have the ideaType are used to denote ideas related to visual languages. The opinionType is the base type for opinions. Tags that have the opinionType are used to indicate opinions related to visual notations. This construction allows creating a description for any visual graph model.

Based on the proposed VI-XML.xsd scheme, a universal visual modeling environment VI was created, whose interface is shown in Figure 5. This universal visual modeling system provides the formation of graphical and textual notations with the ability of data migration between models of different levels and types. The systems include a graphical editor, a library of visual primitives, a VI-XML parser, a PDF report generator, and system components to implement these properties. The system structure includes a module for object representation of models, a user interface module, a diagram editor module, and a layer for the interaction with frameworks, Eclipse Platform and GEF. It is also justified to create peripheral client modules built into the system: a report generator and an import/export module in VIXML. The system is based on the Model-View-Controller design pattern, where the model is an object view module, the view is the interface and diagram editor, and the controllers are the GEF and Eclipse Platform.

This universal visual modeling system provides graphical and textual notations with the ability of data migration between models of different levels and types. To implement these properties, the systems include a graphical editor, a library of visual primitives, a VI-XML parser, a PDF report generator, and system components. As a result, new tools and a new developed methodology of applying a comprehensive approach to the system visual modeling of processes in complex systems at all the stages of life cycle are proposed, from the synthesis of ideas and their formalization to the object modeling of the system, in which the route of model building is iterative.

Based on the proposed VI-XML schema.xsd [13], a universal visual modeling environment VI is created, whose interface is shown in Figure 1 [1, 3].

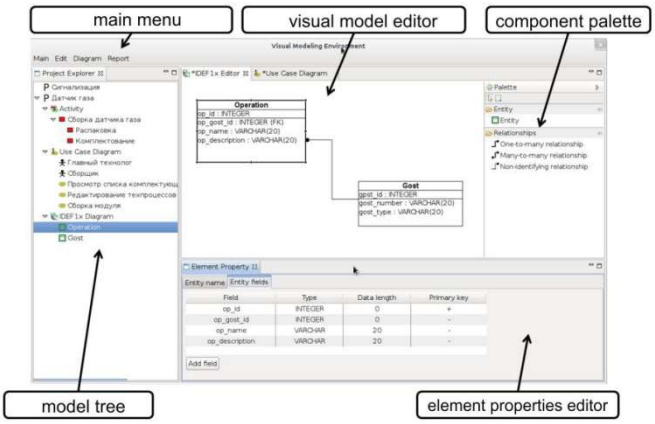


**Figure 5**: Visual modeling environment interface based on VI-XML

The requirements for the universal visual modeling system are formulated. The results of the development of the extensible language and data format of the visual models VI–XML (Visual Intelligence eXtensible Markup Language) are presented.

## 5. Conclusion

The study proposes a new method of applying an integrated approach to system visual modeling of processes in complex systems at all the stages of the life cycle, in which the model construction is iterative. A system of visual design principles built based on expediency, continuity, and covering all the stages of work, including the process modeling and design of information components of the life cycle support on the example of digital technology products, is formed. The requirements for the universal visual modeling system are formulated. The development of the extensible visual model representation language VI–XML (Visual Intelligence extensible Markup Language) is presented.

The practical value of the study lies in its application for the implementation of useful paradigms for managing complex systems. The implementation of the results aims at performing the system stages of the visual design of complex systems, increasing the efficiency of the design as a whole, without losing any essential project elements, and without exceeding the established design deadlines.

## 6. Acknowledgements

Some project results were obtained with the Ministry of Science and Higher Education financial support for project No. 0705-2020-0041, “Fundamental research of methods of the digital transformation of the component base of micro-and nanosystems”.

## 7. References

[1] A. A. Demin, A. I.Vlasov, Visual methods of formalization of knowledge in the conditions of the synchronous technologies of system engineering, in: ACM International Conference Proceeding Series, 2017, p. 3166098.

[2] A. A. Demin, A. E. Kurnosenko, V. A. Shakhnov, A. I. Vlasov, Industry 4.0 visual tools for digital twin system design, Advances in Intelligent Systems and Computing 1295 (2020) 864–875.

[3] A. I. Vlasov, The concept of visual analysis of complex systems in the context of synchronous design technologies, Sensors and Systems 8–9 (2016) 19–25.

[4] A. M. Antipin, D. V. Zhegalin, Methodology of end-to-end design of complex functional systems, Intersectoral Information Service 1 (2014) 51–55.

[5] F. I. Peregudov, F. P. Tarasenko, Introduction to system analysis, Vysshayashkola, Moscow, 1989.

[6] D. A. Pospelov, Large systems: situational management, Znanie, Moscow, 1975.
[7] D. A. Pospelov, Situational management: theory and practice, Nauka, Moscow, 1986.
[8] Lomako E. I. Mathematical and conceptual means of systematic, Sistemnaya Entsiklopediya, Moscow, 2008.
[9] D. V. Koznov, Visual modeling. Theory and practice, NOU Intuit, Moscow, 2016.
[10] V. V. Ilyin, Modeling of business processes. Practical experience of the developer, Intermediator, Moscow, 2008.
[11] L. N.Lyadova, R. A. Nesterov, On the approach to generating analytical models based on visual models of business processes, Vestnik Permskogo Universiteta, Series: Mathematics. Mechanics. Computer science 4(31) (2015) 95–104.
[12] D. Varro, A. Pataricza, Metamodeling mathematics: a precise and visual framework for describing semantics domains of UML models, volume 2460 of Lecture Notes in Computer Science, 2002, pp. 18–33.
[13] A. I. Vlasov, L. V. Juravleva, V. A. Shakhnov, Knowledge-based model for formal representation of complex system visual models, Advances in Intelligent Systems and Computing 1251 (2021) 618–632.
[14] G. N. Kalyanov CASE technologies. Consulting in the automation of business processes. Hotline-Telecom, Moscow, 2000.
[15] T. A. Gavrilova, D. V. Kudryavtsev, I. A. Leshcheva, Ya.Yu. Pavlov, On one method of classification of visual models, Business Informatics 4(26) (2013) 21–34.
[16] R. K. L. Ko, S. S. G. Lee, E. W. Lee, Business Process Management (BPM) standards: a survey, Business Process Management Journal 15(5) (2009).
[17] S. A. White, C. Bock, BPMN 2.0 handbook second edition: methods, concepts, case studies and standards in business process management notation, Future Strategies Inc, 2011.
[18] D. S. Gonoshilov, A. I. Vlasov, Simulation of manufacturing systems using BPMN visual tools, in: Journal of Physics: Conference Series, volume 1353, 2019, 012043.
[19] S. J. Mellor, M. J. Balcer, Executable UML: a foundation for model-driven architecture. Addison-Wesley Professional, 2020.
[20] S. E. Khalzev, V. A. Shakhnov, A. I. Vlasov, Visual methods of high-level system design for digital hardware components, in: Journal of Physics: Conference Series, volume 1515(4), 2020, N042024.
[21] D. V. Koznov, E. V. Larchik, A. N. Terekhov, View to view transformations in domain specific modelling, Programming and Computer Software 41(4) (2015) 208–214.
[22] I. G. Ozerova, E.A.Dmitrieva, G. P. Tsapko, V. N. Vichugov, Methodology of automated construction of schemes in business process management systems, Bulletin of Tomsk Polytechnic University 311(5) (2007) 51–55.
[23] W. Sadiq, M. E. Orlowska, Analyzing process models using graph reduction techniques, Information Systems 25(2) (2000) 117–134.
[24] F. Akhter, Unlocking digital entrepreneurship through the technical business process, Entrepreneurship and Sustainability 5(1) (2017) 36–42.
[25] A. I. Vlasov, L. V. Zhuravleva, V. V. Kazakov, Methods for formalizing cognitive graphics and visual models using XML schemas, Bulletin of the Bauman Moscow State Technical University, Instrument-Making Series 1(134) (2021) 51–77.
[26] A. I. Vlasov, Spatial model to assess evolution of visual design techniques for complicated systems, Sensors and Systems 9(172) (2013) 10–28.
[27] A. I. Vlasov, L. V. Juravleva, V. A. Shakhnov, Visual environment of cognitive graphics for end-to-end engineering project-based education, Journal of Applied Engineering Science 17(1) (2019) 99–106.