WEB-BASED BUSINESS PROCESS MODELING AND OPTIMISATION

Elisabeth Syrjakow, Michael Syrjakow
Institute for Computer Design and
Fault Tolerance (Prof. D. Schmid)
University of Karlsruhe
76128 Karlsruhe, Germany

ABSTRACT

In this paper a Web-based tool component for modeling and optimisation of business processes is presented. A core idea of our approach is that business process descriptions are stored in a standardised XML-based format, which can be automatically transformed into a Petri Net format. Petri Nets are used because they allow a formal analysis, evaluation, and optimisation of business processes. Our tool consists of a collection of collaborating software components, which can be flexibly distributed and accessed over the internet. The standardised XML-based description format for business processes allows to easily collaborate with business process related tools of this area.

KEY WORDS

WWW, XML, SVG, business process, component-oriented software design, Petri Net

1. INTRODUCTION

The Web-based business process modeling and optimisation tool, described in this paper, has been developed in the context of the research project “information logistics for the internet-based process interaction in the intersectoral cooperation”. The intention of this project consists in controlling badly structured, distributed business processes. An intersectoral cooperative planning is characterized by a high number of participants from different enterprises, high complexity, difficult coordination, scarce resources, fuzzy, and/or incomplete information as well as a dynamic change of boundary conditions. To give decision-makers comprehensive information about their possibilities for acting and the arising consequences, process models are designed, implemented, systematically analysed, and optimised. The modelled business processes have to be verified regarding their efficiency and correctness by formal methods. For that purpose Petri Nets are used because of the following advantages:

- Formal design and analysis
- Early detection of deadlocks and inconsistencies by structural analysis
- Behavioural analysis and prognosis by simulation
- Increase of effectiveness and efficiency as well as an improvement in utilisation of resources by optimisation

One big problem of business process modeling is that the involved processes are often described informally [1]. For a formal modeling and analysis this informal description has to be transformed into a formal Petri Net description. Today this task usually is done by hand, which has the following disadvantages:

- It cannot be flexibly and quickly reacted to changing boundary conditions or modifications of the descriptive model
- Initial skill adaptation training to the theory of Petri Nets is required
- Inconsistencies resulting from an incorrect model input by hand may occur

Another problem is the inability to exchange information between different working groups. One reason for this inability to effectively share business process information is the absence of a universal and standardised method for business process description. Usually each working group has independently decided how the analysed business processes will be represented. This prevents an effective information sharing between these groups.

To solve this problem we suggest an XML-based interchange format, which describes the structure of business processes in a standardised way. So, users of various tools can share or exchange business process descriptions. To close the gap between the informal and formal modeling levels, we describe how a descriptive process model can be automatically transformed into a Petri Net model.

One possibility for realisation of such a modeling and optimisation tool would be to upgrade standard software. Those systems however are not flexible enough for that purpose because they usually have a monolithic software structure. A monolithic software design is difficult to maintain and to extend and doesn’t correspond any more
with the modern distributed Web-centered technologies of
today. For this reason we suggest a modern multi-tier
client-server architecture, which enables to link independ-
ent software components over the internet in a more effi-
cient and flexible manner.

This paper is organised as follows: the next Section
gives a short introduction to the dynamic business process
optimisation procedure. Section 3 gives an overview of
our developed modeling and optimisation system by pre-
senting the design decisions, the overall architecture, and
discussing some of the applied technologies. In Section 4,
a model editor is presented in detail representing an
important component of our tool. Finally, in Section 5 we
summarise and draw some conclusions.

2. DYNAMIC PROCESS OPTIMISATION

2.1 The Overall Procedure

Figure 1 shows the procedure of dynamic process
optimisation. Altogether it comprises the following three
working steps: modeling, analysis, and optimisation. In
the first step a descriptive process model from a project
partner is transformed into a formal, executable model
based on Petri Nets. To consider the time aspect Stochastic
Petri Nets (SPN) [2] are used. In the following step the
generated Petri Net model is analysed regarding structure,
behaviour, and parameter sensitivity. In the third step
model optimisation, based on nature analogical optimisa-
tion strategies, takes place. Here, principles of biological
evolution [3] are imitated to solve the optimisation prob-
lem. The result from the dynamic process optimisation
procedure is provided to several project partners for re-
viewing and updating their own descriptive process
model. For exchange of business process information an
XML-based interface is used, which is described in more
detail in Section 2.2.

2.2 XML-based Interface

Project partners are often locally distributed and usually
work with different tools. For an intersectoral cooperation,
they have to share information and for that purpose they
need special interfaces to external systems. Today XML
(Extensible Markup Language) [4] seems to have the
power to become a major means for a homogeneous ex-
change of information. It enables a loose document-based
coupling of different systems. XML allows the specifi-
cation of specialised markup languages for exchange of
information in specific areas of research or business. In
our research work a specific DTD (Document Type
Definition) has been developed, which contains all rele-
vant information required in the involved working groups.
Outgoing from this DTD every project partner can auto-
matically extract information, which is relevant for him
and make a transformation to his own, internally used
format. The Extensible Stylesheet Language (XSL), which
is developed as a part of the W3C (World Wide Web Con-
sortium) Style Sheets Activity, enables to transform docu-
ments written in XML. The XSL Transformations (XSLT)
is designed for use as a part of XSL to describe how a
document is transformed into another XML document. An
overview of the current activities is available at
http://www.w3.org/Style/XSL/. For exchange of XML
files, we have a common server on which every project
partner has his own folder for receiving orders. A partner
is automatically informed about an order via Email. Figure
2 illustrates the XML-based interface between different
working groups.

2.3 Model Transformation

The model transformation, presented in this paper, is
based on a sequential approach [1]. This means that there
are two different modeling schemas: an informal and a
formal one. Starting from an informal description of busi-
ness processes, a transformation to SPNs takes place. The
mapping between the two different modeling schemas is based on the specification of Petri Net modules. Therefore Petri Net modules for processes and their interconnection possibilities are specified. According to DIN-69900, a process describes a transaction that is a time requiring event with a defined beginning and ending. It is in general identified by the fact that its execution requires an allocation of resources and generates costs. The Petri Net module of an elementary process is presented in Figure 3. It consists of the places “arrival”, “processing”, “exit”, and “resources” as well as of the transitions “start” and “end”. The places “arrival” and “exit” model an object, which is waiting for processing and a processed object, respectively. Using time transitions, timing delays such as the processing time for example can be modelled.

![Petri Net module of an elementary process](image)

Figure 3: Petri Net module of an elementary process

3. REALISATION OF THE MODELING AND OPTIMISATION SYSTEM

3.1 Design Decisions

Our system is designed as a collection of collaborating software components. These components are:

- model editor(s)
  A model editor can be realised as an independent stand-alone component. It allows the modeller to edit new and to modify existing models. There may exist textual and graphical editors. Their output is a model description in a specific description format which is characterised by the supported modeling technique.

- model analysis/evaluation modules
  These modules are used to analyse and to evaluate models generated by the model editor. We can distinguish between a mathematical analysis of structural properties (place-invariants, transition-invariants, etc.) and performance evaluations (stationary analysis, transient analysis). Performance evaluation can be computed either analytically or by simulation.

- experimentation modules
  They allow goal-driven experimentation with a model, for example to find optimal parameter settings, determine sensitive model parameters, perform a model validation, etc.

The collaboration of these components is based on two kinds of interactions: exchange of XML documents and invocation of model evaluation modules. For remote invocations component “wiring” standards like CORBA (Common Object Request Broker Architecture), EJB (Enterprise JavaBeans) or DCOM (Distributed Component Object Model) can be applied. Altogether the component-oriented software design of our tool has the following advantages: 1.) It enables a flexible distribution of the involved components within a computer network. 2.) It allows user access by traditional application clients or by Java-based Web clients. 3.) It enables an integration of existing monolithic tools by transformation of the proprietary model description format into a standardised XML-format and/or by appropriate component wrappers. 4.) It eases tool modifications and extensions. 5.) It represents a good basis for agent-based approaches. 6.) Besides all these technical advantages component-orientation opens several economic and organisational advantages like software reuse and clear separation of responsibilities.

3.2 Tool Architecture

For the distributed realisation of our component-oriented modeling and optimisation tool, we have chosen a modern multi-tier architecture shown in Figure 4.

![Distributed multi-tier architecture for business process modeling and optimisation](image)

Figure 4: Distributed multi-tier architecture for business process modeling and optimisation

It consists of three tiers. The first tier provides Web-clients running as Java-applets within a Web-browser such as
Netscape or Internet Explorer for example. The server, representing the second tier of the distributed architecture, provides a model editor and an optimiser component. The model editor offers the following functionalities: 1.) Import and export of XML files. 2.) Model transformation from a descriptive process model into an SPN. 3.) Specification of a goal function. The optimiser component offers different methods for model optimisation. Its task is to find optimal parameter settings of a model. For that purpose nature analogical optimisation strategies such as Genetic Algorithms (GA), a Hill-Climbing (HC) strategy, and a hybrid strategy (GA&HC) are applied. The third tier represents a backend-server on which the simulator component is located. For this purpose an existing SPN simulator has been used. Persistent modeling data are saved in a data pool.

3.3 Applied Technologies

The implementation of the client-server architecture, shown in Figure 4, is mainly based on the following three technologies: Java, XML, and CORBA. All Java programming was done using JBuilder 4.0 as a development environment. The graphical user-interface has been realised with the Java Swing classes [5]. Java’s object-orientation, platform independence, and the Applet concept were major factors for its choice as the development language. An additional advantage of Java is that it currently offers the best XML processing capabilities and support.

In addition to Java, XML was applied as another major technology. For XML file utilisation within Java, the AElfred XML parser from Microstar Software Ltd. was used. This parser provides the SAX (Simple API for XML) parsing capabilities. XML has been used because of the following advantages:

- it enables data exchange between incompatible systems
- it enables syntactical and semantical validation of a document
- easy to generate and to handle
- not limited to a special application
- Internet-based access/presentation
- support of an advanced search
- many XML tools are available for free

In order to connect server and backend-server components, we have decided to use the Common Object Request Broker Architecture (CORBA) [6]. CORBA represents a powerful middleware standard providing the following advantages:

- programming-language independent interface
  Interfaces between clients and servers are defined in a standardised Interface Definition Language (IDL).
- easy legacy integration
  Using IDL, programmers can encapsulate existing applications in wrappers and use them as objects on the ORB (Object Request Broker).
- rich distributed object infrastructure

Distributed applications require more functionalities than simple method invocations. CORBA offers them a rich set of distributed object services and facilities.

4. MODEL EDITOR

4.1 Graphical Editor

The model editor, shown in Figure 5, consists of the following basic parts:

- a tabbed pane with the panels “xmlImport”, “xmlExport”, and “xmlFiles”
- a text field to display XML files or the generated Petri Net
- a transformation button to automatically transform a process model into an SPN
- a goal function button to allow the specification of a goal function for optimisation

4.2 Import and Export of XML Files

For the data transfer between project partners XML files are used. For this reason import and export functionality for XML files is offered by the model editor. The import functionality allows the user to download a process model. After a correct input of his password an import window, as shown in Figure 6, appears. In this window all the XML files which are located on the server are shown. Additionally the date, when the file has been put on the server, is indicated.

Figure 5: GUI of the model editor

Figure 6: Import of an XML file
The export functionality enables the user to put an XML file onto the server. The export procedure is executed analogously to the import procedure with one difference: an email with a notice is sent to the appropriate project partner.

4.3 Generating Petri Net Files

In order to allow a user to transform an XML file into a Petri Net file, a method for reading the XML file is needed. Thus, XML processing capabilities are necessary. XML processing is handled by an XML parser that implements the SAX API, the DOM (Document Object Model) API, or both. The DOM is a tree-based API, which maps an XML document into a set of objects in a tree-like structure determined from the document DTD. The DOM API allows the programmer to read objects (XML nodes on the tree), to modify XML-nodes, and to add new XML nodes to the tree. Then the tree-structure can be navigated in order to search, extract or change information. In contrast, SAX parsers are faster, less memory-intensive, and provide a simple, efficient manner for processing an XML file in a sequential, one-pass fashion.

The task of reading an XML file and creating instances of the appropriate model transformation classes is well-suited for a SAX parser. The SAX parser reads through the XML files sequentially and fires events, which perform the desired information processing, when the structures of interest, such as the start of an element, are found.

4.4 Views

There exist two different views on a Petri Net model: a textual and a graphical. Textual representations are for example an XML format or a specific model description format. For a graphical representation the SVG format is well suited. SVG stands for Scalable Vector Graphics. SVG 1.0 is a Web standard (a W3C Recommendation). It is a language for describing two-dimensional graphics in XML. SVG supports three types of graphic objects: vector graphic shapes (e.g., paths consisting of straight lines and curves), images, and text. SVG drawings can be interactive and dynamic. Animations can be defined and triggered either declaratively (i.e., by embedding SVG animation elements in SVG content) or via scripting. An overview of the current activities is available at http://www.w3.org/TR/SVG/.

Basic forms for graphics are rectangles, circles, ellipses, lines, and polygons. Using them a graphical representation of a Petri Net can be easily built. For example, a circle for the representation of a place is described with SVG as follows:

```xml
<circle cx="20" cy="50" r="10" fill="white" stroke="black" stroke-width="1"/>
```

Thereby “cx” and “cy” are the coordinates for the central point of the circle and “r” represents the radius. The appearance of the circle is described with “fill” for the colour, “stroke” for the coat colour, and “stroke-width” for the coat thickness. For a transition a rectangle is defined with SVG as follows:

```xml
<rect x="80" y="40" width="4" height="20" fill="black" stroke="black" stroke-width="1"/>
```

This specification comprises the coordinates of the left upper corner as well as the width and height of the rectangle. The other specifications correspond to those of the circle. Figure 7 shows a graphical representation of a Petri Net model with SVG. The visualisation of the SVG graphic has been realised using Batik from Apache, which is available at http://www.apache.org.

![Visualisation of a Petri Net using SVG](image-url)
4.5 Specification of a Goal Function

A goal function describes a relationship between model input and model output parameters. The goal of model optimisation is to find a parameter combination where the goal function reaches an optimal value. The goal function can be optimised towards a minimal or a maximal value. Model input parameters for example may be processing times and resources. Model outputs may be throughput, waiting times, and resource utilisation. Figure 8 shows the window, which allows the specification of a goal function. It consists of the following parts:

- a radio button to enable the choice between the optimisation regarding time or resources
- a table, which contains time or resource parameters for each process
- a text field to specify a goal function
- a radio button to distinguish between a maximisation and a minimisation problem

Figure 8: Specification of a goal function

A detailed description of the process of model optimisation, the applied algorithms, and the developed graphical user interface can be found in [7].

5. CONCLUSIONS

The component-oriented design of our tool for modeling and optimisation of business processes, presented in this paper, offers the following main advantages: 1.) It eases the realisation of Web-based user access, which is a fundamental requirement of an intersectoral cooperation. 2.) It provides openness for all kind of extensions.

To allow a flexible information exchange between project partners, an XML-based interchange format is used. Despite all these technical advances, the task of creating a simulation model from an informal description of business processes still requires significant expertise. To perform this transformation, an approach was described, which allows to automatically map business processes to Petri Net modules.

ACKNOWLEDGEMENTS

We want to thank Prof. D. Schmid for his encouragement and support of our work. We also thank our students, especially Sabine Schillinger and Florian Schmidt for their engagement and contributions.

REFERENCES


AUTHOR BIOGRAPHIES

Elisabeth Syrjakow was born in 1970 in the Federal Republic of Germany. She received the Dipl.-Inform. degree from the University of Karlsruhe, Germany in 1999. Since then she has been with the professional group Modeling and Simulation at the Institute for Computer Design and Fault Tolerance at the University of Karlsruhe. Her Email and Web addresses are <lisa@ira.uka.de> and <goethe.ira.uka.de/people/lisa/>.

Michael Syrjakow was born in 1964 in the Federal Republic of Germany. He received the Dipl.-Inform. degree from the University of Karlsruhe, Germany in 1991. Since then he has been with the professional group Modeling and Simulation at the Institute for Computer Design and Fault Tolerance at the University of Karlsruhe. In February 1997 he received the Ph.D. in Computer Science from the University of Karlsruhe. His Email and Web addresses are <syrjakow@ira.uka.de> and <goethe.ira.uka.de/people/syrjakow/>.