# Programming in the Large based on the Business Process Modelling Notation

Christian Emig, Christof Momm, Jochen Weisser, Sebastian Abeck

Cooperation & Management
Universität Karlsruhe (TH)
Zirkel 2
76128 Karlsruhe
{emig | momm | weisser | abeck}@cm-tm.uka.de

**Abstract:** A software application is related to the processes it supports. Today, UML diagrams esp. use case diagrams and activity diagrams are often used to model the relevant aspects of the processes within the analysis phase. In the design phase the models are manually mapped to the business layer of the software application. In the context of Service-oriented Architectures (SOA) Programming in the Large takes a different approach: Business processes are described in a programming language, i.e. a process language which can be automatically mapped to an execution language and executed by a process engine. In this article we show how Programming in the Large can be practically applied in a software engineering process. We use the Business Process Model Notation (BPMN) as process programming language. A BPMN description can be mapped to the Business Process Execution Language (BPEL) which is a widely accepted standard to compose Web services.

## 1 Introduction

Service-oriented Architectures (SOA) will form the basis of future information systems, with all the required functionality being provided by loosely coupled Web services (i.e., core Web services) [ABH04] [IBM04]. These core Web services are then being assembled to composite Web services in order to directly support business processes. For software developers the challenge will no longer be the coding of the required functionality but the orchestration of already existing Web services to executable business processes [AHW03]. This approach has commonly been referred to in relevant literature as "Programming in the Large" whereas the development of core Web Services is known as "Programming in the Small" [Le03].

This paper presents a top-down approach for Programming in the Large in SOA. The impact of this paper lies on describing concepts of the higher level SOA layers like the choreography and service composition. Concepts that are not in the focus of the approach described in this paper are mainly blinded out. In other words the component layer and parts of the core Web service layer [BC04] are not examined here. Beginning with a concrete business process taken from the field of higher education it is demon-

strated how a high-level process definition has to be refined in order to be automatically mapped to an executable process definition based on BPEL4WS [BPEL1.1]. For the modelling of both, the high-level and the refined process definitions, the Business Process Modelling Notation (BPMN) [BPMN1.0] [Wh04] is used, which was specifically designed to support an automatic mapping to BPEL.

## 2 Refinement of High-Level BPMN Process Definitions

The BPMN pursues two major objectives: Firstly, the notation should be easy to understand for every role participating in the development process (business analysts, software developer etc.) and secondly it should reduce the gap between the business process design being focused on in the analysis phase and the process implementation being looked at in the design and implementation phase. This is ensured by setting up on a mathematically based, internal model that enables the mapping between BPMN's graphical elements to the underlying constructs of (XML-based) executable business process languages like BPEL.

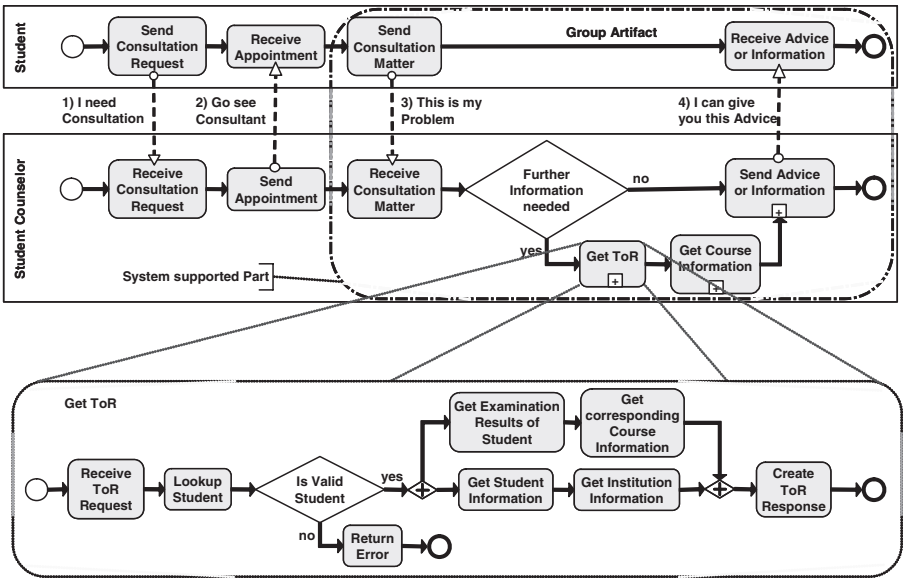


Figure 1: Refinement of "Get ToR" as Part of the Student Consultation Process

The business process, which is modelled in the upper section of Figure 1, is taken from the higher education area, for example a university. The business case concerns a student who needs consultation, for example after having failed an exam. In the Business Process Diagram (BPD) the two participants are modelled in two pools with their own internal sequence flow. At the right side of the upper diagram all those activities are grouped that the business analyst decided should be supported by computer systems. Such high-level process models are typically created during the analysis phase and can yet not be

mapped to an executable process definition. Therefore, most of the activities need to be further refined by means of more detailed BPDs, which corresponds to the design phase. In general, the following distinction between activities can be made: Activities that are "Service Tasks" provide some sort of service which can be automated and therefore mapped to BPEL. "User Tasks" on the other hand contain activities that have to be performed manually by a user. Then a mapping is not or only partially possible.

In the following we concentrate on the Service Task activity "Get ToR", which is carried out after the decision gateway where the process determined that further information is needed by the student counsellor for in-depth assistance. Thereby, a so-called Transcript of Records (ToR) is created by the supporting system and returned to the requestor. A Transcript of Records is a standardized aggregation of a student's achieved results at a university. The activity "Get ToR" is marked with a [+] that holds the information that there exists a decomposed view on this activity.

The bottom part of Figure 1 shows the refined process definition for the high-level activity "Get ToR". All activities in this diagram represent Service Tasks. Hence, the whole BPD can be mapped to a BPEL process definition. Thereby, the four activities executed within a parallel (AND) fork and join (Figure 1, top right) are directly mapped to Web service operation calls. These Web services may be implemented in Java, for example.

## 3 Mapping of the refined Process Definition to BPEL

This chapter demonstrates how to map the refined BPMN process "Get ToR" depicted in Figure 1 to an executable BPEL process under the precondition of maximizing the automatically executable parts. This can be either done using software tools like, for instance, the System Architect by Popkin Software [OR03] but can as well be done manually using the in-depth described mapping rules in [BPMN1.0]. For the latter approach a graphical BPEL design tool like the Oracle BPEL Designer can be employed in order to minimize effort.

Unfortunately, both approaches imply some further challenges. The generated code contains almost no information about the WSDL of the orchestrated Web services. If a Web service for a particular activity already exists this work can be done tool-supported by a BPEL design tool, because the WSDL documents hold all additionally needed information.

If there is no existing Web service to support the accordant activity, the procedure slightly differs. If so, the required core Web services have to be designed and implemented manually. Thereby, a well-refined BPMN process definition can be used to derive the necessary Web service descriptions. As a first step, these deduced WSDL documents can be orchestrated in a BPEL process without an underlying Web service implementation. As a second step, existing tools like WSDL2Java as part of the Apache Axis Framework [AXIS], which allow an automatic generation of skeleton code, can be employed. This skeleton code then needs to be enriched with the actual business logic. However, the implementation of the business logic as core Web services is part of the

Programming in the Small. As the emphasis in this paper is placed on the Programming in the Large, it is assumed, that the required core Web services along with their corresponding WSDL documents are already available.

When applying this approach to our exemplary "Get ToR" process modelled in Chapter 2 the resulting BPEL code looks as follows:

```
<!-- Orchestration logic -->
<sequence name="main">
    <receive name="ReceiveToRRequest" partnerLink="Client" ... />
    <invoke name="LookupStudent" partnerLink="StudentDBService" ... />
    <switch name="IsValidStudent">
        <case name="Yes">
            <flow ... >
                <sequence ... >
                    <invoke name="GetStudentInformation" ... />
                    <invoke name="GetInstitutionInformation" ... />
                </sequence>
                <sequence ... >
                    <invoke name="GetExaminationResultsOfStudent" ... />
                    <invoke name="GetCorrespondingCourseInformation" ... />
                </sequence>
            </flow>
        </case>
        <otherwise name="No">
            <invoke name="ReturnError" ... />
        </otherwise>
    </switch>
    <assign name="CreateToR">
        <!-- Copying some parts of the collected information into a new ToR variable -->
    </assign>
    <reply name="ReturnToR" ... />
</sequence>
```

Figure 2: BPEL excerpt of the generated ToR Process

Note that the displayed code is only an excerpt. Most of the attributes, variables and assignments as well as the complete error handling are omitted for better readability.

Within the generated BPEL code the "invoke" elements are used to call external Web services by using their WSDL specifications. In this case every external service corresponds with activities modelled in Figure 1. The "switch" tag maps the gateway element of BPMN to BPEL. A "sequence" is sequential action while a "flow" correlates to the fork object of BPMN. Tags can include many attributes which mostly can be automatically derived from the Web service description (WSDL) of the participating Web services.

BPEL process definitions generated in this way can by executed by so-called BPEL-Engines. Thereby the core Web services used by the BPEL process have to be deployed in advance. The BPEL process can only be executed if the WSDL documents for all involved Web service are available. Examples for existing BPEL engines are Active-BPEL [ABP] by ActiveEndpoints and the Oracle BPEL Process Manager [OBPM].

All engines have in common that the deployment of the BPEL process requires further engine-specific additions like for instance deployment descriptors.

## 4 Outlook

It is important to notice that Programming in the Large as illustrated in this paper, is the next step in the evolution of software engineering. However, it does not replace but perpetuates the existing approaches of programming. Existing approaches do not adequately support the first phase of software engineering where business processes of the future user of the software system have to be analyzed. Programming in the Large fills this gap by providing executable process descriptions which are based on service-oriented components, typically Web services.

To ensure a more efficient mapping between the BPMN diagrams and the executable BPEL code the tools have to be improved. Although this kind of programming is still in a very early stage, the high potential of this evolutionary step in software engineering is obvious. Standards, such as BPMN and BPEL, are available to apply this concept to practical software problems as we have demonstrated in the paper.

## 5 Literature


[ABH04] Ali Arsanjani, Bernhard Borges, Kerrie Holley: Service-oriented Architecture, Article published in DM Direct Newsletter, http://www.dmreview.com/article_sub.cfm?articleId=1013602, November 2004

[ABP] ActiveBPEL - The Open Source BPEL Engine, http://www.activebpel.org/

[AHW03] Wim M. P. van der Aalst, A. H. M. ter Hofstede, M. Weske: Business Process Management: A Survey, Lecture Notes in Computer Science, Band 2678, Seiten 1-12, Springer-Verlag, Berlin, 2003

[AXIS] Apache <Web services /> Project, http://ws.apache.org/axis/

[BC04] George Brown, Robert Carpenter: Successful Application of Service-Oriented Architecture across the Enterprise and Beyond, http://www.intel.com/technology/itj/2004/volume08issue04/art09_successful/vol8_art09.pdf, ISSN 1535-864X, November 2004

[BPEL1.1] Business Process Execution Language for Web Services (BPEL4WS), Version 1.1, http://www.ibm.com/developerworks/library/ws-bpel, May 2003.

[BPMN1.0] Business Process Management Initiative (BPMI): Business Process Modelling Notation (BPMN), Version 1.0, BPMI.org, May 2004.

[IBM04] IBM Redbook: Patterns – Service-Oriented Architecture and Web Services, April 2004

[Le03] Frank Leymann: Web Services—Distributed Applications without Limits, Business, Technology and Web, Böblingen, 2003

[OBPM] Oracle BPEL Process Manager http://www.oracle.com/technology/products/ias/bpel/index.html

[OR03] Martin Owen, Jog Ray: BPMN and Business Process Management – Introduction to the New Business Modeling Standard, Popkin Software 2003

[Wh04] Stephen A. White: Introduction to BPMN; IBM Cooperation 2004

# Workshop
# ”Sicherheit in komplexen, vernetzten Umgebungen”

Einen unaufhaltsamen Trend der heutigen Informationsgesellschaft stellt die zunehmende Vernetzung von Computersystemen in den unterschiedlichen Bereichen dar. Den vielfältigen sich daraus ergebenden Vorteilen steht eine zunehmende Menge von Bedrohungen entgegen, die auf vielen Ebenen einen Einfluss auf das Leben und die Gesellschaft haben können. Getreu dem Motto ”Informatik LIVE!” der diesjährigen Jahrestagung der Gesellschaft für Informatik setzt sich der Workshop mit diesen Einflüssen und gezielten Reaktionen darauf auseinander.

Die Ziele und Methoden der ”klassischen” Informationssicherheit werden auch in zukünftigen komplexen vernetzten Umgebungen Bestand haben. Die Schwerpunkte und Abhängigkeiten verlagern sich jedoch zusehends. So stehen beispielsweise Fragestellungen wie Verfügbarkeit und Modellierung von Vertrauen in solchen Umgebungen eher im Mittelpunkt als Verfahren zur Sicherstellung von Vertraulichkeit und Integrität. Vor diesem Hintergrund möchte dieser Workshop dazu beigetragen, den Dialog zwischen Experten aus dem Bereich der Informationssicherheit anzuregen und zu vertiefen.

Dieser Workshop trägt – ergänzend zu etablierten Veranstaltungen dieser Art – aktuelle Erkenntnisse und Ergebnisse von Forschungsarbeiten zusammen, die Lösungen für die sich aus der verstärkten Vernetzung ergebenden Probleme anbieten. Thematisch wird ein breites Spektrum geboten, so etwa Modellierung und Visualisierung von Aspekten komplexer Netze, Einsatz von Intrusion-Detection-Systemen in komplexen Umgebungen, PKI-Architekturen und Computerforensik.

Der Dank der Organisatoren geht an alle Beitragenden, an die GI-Fachgruppe SIDAR, an die Tagungsleitung sowie an das Programmkomitee, das sich wie folgt zusammensetzte: Ulrich Flegel (Universität Dortmund), Felix C. Freiling (RWTH Aachen), Olaf Gellert (Presecure GmbH), Marko Jahnke (FGAN/FKIE), Christoph Karg (FH Aalen), Klaus-Peter Kossakowski (DFN-CERT Services GmbH), Michael Meier (BTU Cottbus), Jens Tölle (FGAN/FKIE) und Stephen D. Wolthusen (Fraunhofer IGD).

Bonn, den 22. September 2005

Marko Jahnke und Jens Tölle
(FGAN/FKIE)