Chapter XVI

Business Processes in UML

Peter Rittgen
Technical University Darmstadt, Germany

ABSTRACT

Today, modeling business processes and modeling software is done using different notations that are designed to fit the special needs of the respective tasks. However, this fact results in a painful methodological gap between business models and software models, which is hard to bridge. This problem becomes even more painful if we try to build software to support certain business models because a smooth transition between the employed notations is usually not supported, or due to methodological problems, impossible. In order to allow for a smoother transition, we propose using Business Process Diagrams (BPDs), which are based on the UML activity diagrams (in both business and software worlds). We show how to derive BPDs from the well-known business process language of Event-driven Process Chains (EPCs) using Petri nets as a common process meta-model.

INTRODUCTION

When analyzing a company for possibilities of information systems support, a major task consists in identifying the relevant business processes and describing them in a suitable modeling language. Many such languages have been developed over the years: IDEF (Integrated DEFinition, (Bruce, 1992)); Role Activity Diagrams (Ould, 1995); and ARIS/EPC (ARchitecture of integrated Information Systems / Event-driven Process Chain, (Scheer, 1999)) to name but a few.

These languages share a common characteristic in that they are not equipped to support the design of software. The Unified Modeling Language (UML), cp., e.g., (Rational Software et al., 1997), on the other hand, does provide the features pertinent to software engineering although it is less qualified for use in domain-oriented models. In practice, this leads not only to a separation of concerns, but also to a heterogeneous
usage of modeling languages: domain experts using business languages and software engineers using UML. This creates an undesirable gap between domain and software models and represents a source for mistakes that are hard to correct.

Hence, we suggest Business Process Diagrams (BPDs), i.e., a language closely related to UML activity diagrams, for both business and software development when taking a closer look at a typical business process language called Event-driven Process Chains (EPCs). Our main objective is to ensure that all features of EPCs are present in BPDs, too. At the same time, we want to make certain that the semantics of the corresponding EPC and BPD diagrams coincide. We achieve this by using the formal, i.e., mathematical, process language of Petri nets as a common meta-model to define the meaning of both diagram types. That will allow us to conclude that the resultant language of BPDs is suitable for designing processes for not only software, but also in the business domain. It also enables an automatic transformation from EPC to BPD and back, i.e., switching between business and software views.

In the following sections, we first introduce Event-driven Process Chains as typical models for business processes and define their semantics in the light of the common meta-model of Petri nets. We then enhance the suitability of UML activity diagrams for business modeling, which leads us to Business Process Diagrams (BPDs). Their semantics are also based on the common meta-model, which ensures the compatibility of both EPC and BPD. We conclude by showing examples of typical business processes with representations as EPCs and BPDs.

BACKGROUND: EVENT-DRIVEN PROCESS CHAINS

Event-driven Process Chains were introduced to draw a graphical representation of a business process. They consist of the following elements as shown in Figure 1.

Ever since the introduction of EPCs by Scheer, there have been many opinions on how a correct EPC should look. Proposals ranged from syntactical issues (which nodes can be linked to one another?) to semantics (what is the exact meaning of a connector?). On the syntactical level, some rules have been established that are now generally accepted, for example, (Keller & Teufel, 1997): An EPC consists of strictly alternating sequences of events and functions (i.e., processes) that are linked by logical connectors (AND, OR, XOR). There are opening connectors (splits) and closing connectors (joins). The AND stands for parallel threads, the XOR for mutually exclusive alternatives, and the OR for arbitrarily selecting many alternatives. Events are instantaneous happenings that trigger a business function or process. Events may also be the results of finishing a function or a process. A function is an elementary business activity; a process is a business activity, which is refined through another EPC. A function/process can have the organizational unit that is responsible for it and data containers from where it gets input or stores output attached to it. Among the syntactical restrictions for EPCs that are named in Keller and Teufel (1997) are:

K1: There are no isolated nodes.

K3/4: Functions and events have exactly one incoming and one outgoing edge (except start and end events).

Copyright © 2003, Idea Group Inc. Copying or distributing in print or electronic forms without written permission of Idea Group Inc. is prohibited.
A Unified Software Reengineering Approach towards Model Driven Architecture Environment
www.igi-global.com/chapter/unified-software-reengineering-approach-towards/29610?camid=4v1a