Decision Support Systems 51 (2011) 53–64

Contents lists available at ScienceDirect

Decision Support Systems

journal homepage: www.elsevier.com/locate/dss

Building insightful simulation models using Petri Nets — A structured approach

Durk-Jouke van der Zee *

Department of Operations, Faculty of Economics & Business, University of Groningen, P.O. Box 800, 9700 AV, Groningen, The Netherlands

ARTICLE INFO

Article history:
Received 12 February 2010
Received in revised form 3 November 2010
Accepted 21 November 2010
Available online 26 November 2010

Keywords:
Simulation
Petri Nets
Visualization
Problem structuring & solving

ABSTRACT

Petri Nets have essential strengths in capturing a system's static structure and dynamics, its mathematical underpinning, and providing a graphical representation. However, visual simulation models of realistic systems based on Petri Nets are often perceived as too large and too complex to be easily understood. This constrains stakeholders in participating in such modeling and solution finding, and limits acceptance. We address this issue by considering a structured approach for guiding the analyst in creating more insightful models. Key elements are a domain-related reference architecture that supports conceptual modeling coupled with uniform rules for mapping high-level concepts onto low-level Petri Net components. The proposed approach is implemented and illustrated in the manufacturing domain.

1. Introduction

Petri Net formalisms have proved to be successful tools for the modeling and simulation of manufacturing systems [6,8,23,41]. Their success is underpinned by their basic strengths in accurately describing a system's static structure and its dynamics, the availability of mathematical analysis methods, and their graphical nature.

Many authors, however, argue that the use of Petri Nets in decision support for the design of manufacturing systems could be further improved if the model's size and complexity could be reduced. These characteristics are seen as hindering more active stakeholder understanding, participation, and acceptance of solutions [2,10,14,26]. In this article, we address this issue by developing guidance that could assist the analyst in defining more insightful visual models.

Stakeholders' understanding of visual model elements, and their workings, starts from a recognition of high-level concepts [12,21]. Examples of such concepts within the manufacturing domain relate to machines, buffers, planners, and goods. The selection of concepts and their visual representation follows from the analyst's creativity. This creativity is bounded and guided by implicit or explicit guidelines, that is by good modeling practices and principles [20], domain-related insights [30] and, last but not the least, the logic and libraries that underlie simulation software [11].

Typically, Petri Nets start from a few basic low-level components used to define high-level constructs that resemble manufacturing entities. These low-level beginnings explain model size on the one hand, and the great efforts that the analyst has to put into insightful model structuring on the other. Basically, we assume model structuring to refer to both sound conceptualization, i.e., the choice of entities, their activities, and their relationships considered characteristic of a domain; and formalization, i.e., the mapping of the respective system elements onto model components. In this paper, we consider guidance for the analyst related to both activities.

The purpose of this article is to propose a structured approach to guide the analyst in building more insightful Petri Net simulation models. Key elements of the approach are a reference architecture that captures essential object classes for a domain, and a set of mapping rules for representing the respective objects as Petri Nets. Here, a reference architecture relies on a set of decomposition principles characterizing the field of interest. In this paper, the approach is implemented within the manufacturing domain by linking a manufacturing reference architecture [36] to a specific Petri Net formalism, i.e., ExSpect™, by defining mapping rules. The use of this approach is illustrated and evaluated through a case study.

The remainder of the article is organized as follows. In Section 2, we provide an outline of our approach to guiding analysts in building more insightful Petri Net simulation models. Next, in Sections 3 and 4, we discuss key elements of its implementation in a manufacturing simulation, i.e., a reference architecture plus a set of rules for mapping object classes within the architecture onto basic Petri Net components. This includes a short introduction to ExSpect™. In Sections 5 and 6, we illustrate and evaluate the use of our approach with a case example. Finally, in Section 7, we summarize our main conclusions.

2. A structured approach to more insightful Petri Net modeling

In this section, we outline our approach to achieving more insightful Petri Net modeling, starting from the situation illustrated in Fig. 1. The figure distinguishes between the model cycle for a typical simulation

* Tel.: +31 50 363 4687; fax: +31 50 363 2032.
E-mail address: d.j.van.der.zee@rug.nl.

0167-9236/$ – see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.dss.2010.11.028

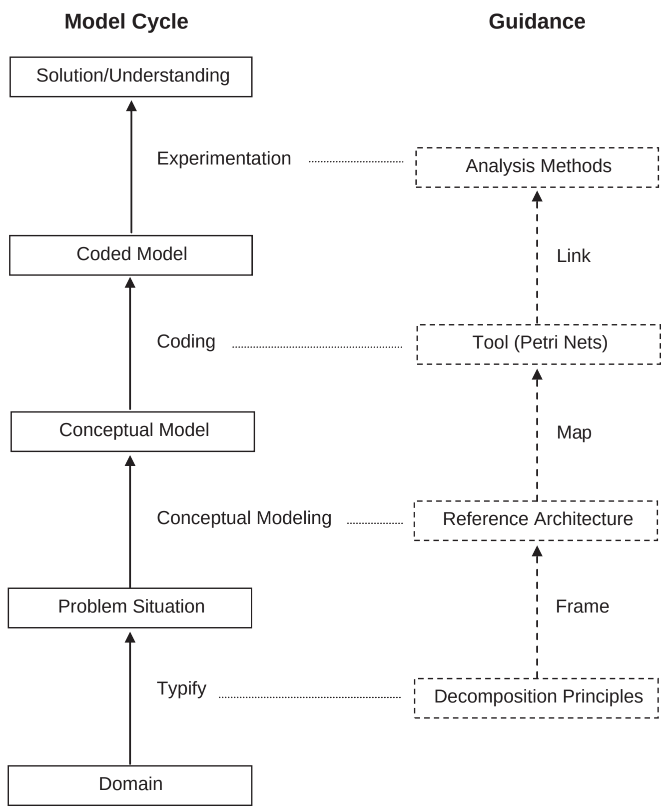

Fig. 1. A structured approach for more insightful modeling.

project (see, for example Robinson [24]) and guidance for the analyst in executing modeling activities — as it follows from our approach.

Milestones in a simulation project are a description of a problem situation, a conceptual model, a coded model, and decision support in terms of solutions, and stakeholder understanding. Below, we will discuss the modeling activities that underlie these milestones and relate these to guidance for the analyst.

A *description of a problem* will usually give some idea of its context, and the dilemma faced by one or multiple stakeholders. Although the quality and detail of a problem description may differ greatly, it will refer, either explicitly or implicitly, to phenomena which are characteristic of a certain domain. This is the net effect of applying those cognitive approaches that are referred to as decomposition and classification [4,29]. *Decomposition and classification principles*, either implicit or explicit, may be used to capture domain characteristics. For example, in the manufacturing domain, it is common to distinguish between infrastructural elements such as machines and buffers, and movable objects such as goods, staff, and tools. In earlier work [34,37], we identified and summarized several decomposition principles that are of relevance to the manufacturing field. As such, they may be helpful to an analyst in *typifying* a problem situation.

A *conceptual model* captures the model contents in terms of scope, i.e., the system entities to be modeled, and detail, i.e., the entity attributes to be considered. It serves both as a blueprint for coding and as a decision document, to be agreed upon by both analyst and stakeholders. We propose addressing the need for model detail, and its insightful representation, following from these uses through a reference architecture. A reference architecture refers to a well-defined conceptual view of a domain, identifying and characterizing generic object classes and their workings. It builds on a comprehensive (*framed*) set of domain-related and more general decomposition principles. Note that the use of such a reference architecture should typically be complemented by the application of good modeling practices [20], including model simplification principles and an evolutionary model set-up [25].

Essentially, we assume a conceptual model that captures the problem situation, in terms of high-level manufacturing concepts, as the basis for defining the executable *coded model*. In principle, high-level concepts cannot be mapped onto basic Petri Net components in a straightforward way due to the latter's low-levelness. Typically, high-level concepts can only be modeled by defining aggregates of basic Petri Net components. Our approach suggests the definition and use of explicit *mapping rules*. These rules prescribe a format according to which object classes in the reference architecture should be defined in terms of basic Petri Net components or aggregates thereof. The idea of uniformity, which underlies the rules, is meant to support the acquisition of model insight by creating “familiar” and “appealing” net structures.

User interaction with a visual simulation model is facilitated by *linking* the model to an experimental frame [40]. This defines experimental factors, i.e., model entities open to modification in some respect, and model outputs, i.e., observations on model behavior which indicate and/or explain system performance. Alongside performance evaluation through simulation, many Petri Net formalisms also support the use of mathematical methods for system analysis. Given our focus on the graphical qualities of Petri Nets, we do not consider such methods here. However, their possible integration into our approach to develop more insightful Petri Net simulation modeling is considered to be both relevant and an interesting issue for future research.

3. Conceptual model — a reference architecture for manufacturing simulation

In this section, we discuss a reference architecture that is suitable for guiding an analyst in defining a conceptual model for manufacturing simulation. First, we categorize the decomposition principles that underpin the architecture. Next, we consider essential characteristics of the architecture. Finally, we introduce a method for its use. For more details on these aspects, see Van der Zee and Van der Vorst [36], Van der Zee et al. [38]. Further, applications of the architecture may also be found in Van der Zee et al. [38], and Van der Vorst et al. [33].

3.1. Basis of the reference architecture — decomposition principles

The decomposition principles that underlie the reference architecture are shown in Table 1. Principles I – III are rather well-known and follow from the notion of a system boundary (I), basic system logic (II), and queuing systems (III). According to Lefrancois and Montreuil [13], making a distinction between intelligent and non-intelligent entities (IV) permits a more natural and richer presentation and implementation of modeled systems. In such a context, agents represent intelligent beings. Agents are used to implement the decision rules inherent to manufacturing system planning and control. Examples include routines for scheduling, dispatching, and releasing jobs to a machine or department. Further, principles I – IV tend to be valid for a much wider category than only manufacturing systems.

Table 1
Decomposition principles underpinning the reference architecture for simulating manufacturing [34].

Decomposition principles	
I	External and internal entities
II	Movable and non-movable entities
III	Queues and servers
IV	Intelligent and non-intelligent entities
V	Infrastructure, flows and jobs
VI	Modality: physical, information, and control elements