Expert Systems With Applications xxx (2015) xxx–xxx

Contents lists available at ScienceDirect

Expert Systems With Applications

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

Context-adaptive Petri nets: Supporting adaptation for the execution context

Q1 Estefanía Serral*, Johannes De Smedt, Monique Snoeck, Jan Vanthienen

Department of Decision Sciences and Information Management, Faculty of Economics and Business, KU Leuven, Naamsestraat 69, Leuven B-3000, Belgium

ARTICLE INFO

Keywords:
Petri nets
Context modeling
Context adaptivity

ABSTRACT

Petri nets (PNs) are a mathematical and graphical modeling language with powerful analysis techniques. They have been successfully used in several areas, such as business process management, human computer interaction, and pervasive computing. Within these areas, context adaptivity has recently emerged as a new challenge to explicitly address fitness between system behavior and its execution context. However, the existing PN formalisms do not provide reliable modeling, simulation, and verification techniques that can accurately consider the system's execution context and adapt to it in order to reflect the system execution reality. This paper addresses this problem by presenting context-adaptive Petri nets (CAPNs), a formalism that allows the modeling of context-adaptive behavior by integrating the powerful modeling and analysis techniques of PNs with very expressive context data management techniques. The formalism is supported by a tool that allows its modeling, simulation, and verification. The contributions have been validated using a case-based evaluation showing very promising results. CAPNs will allow organizations to accurately describe, enact, and analyze the behavior of their dynamic systems in a more reliable and realistic way, allowing them to leverage more informed decisions, to make better use of their resources, and to increase therefore their competitiveness.

1. Introduction

Petri nets (PNs) are a mathematical modeling language that has a graphical notation and a formal definition for its execution semantics (Murata, 1989). PNs have well-established formal mechanisms for modeling and analyzing concurrent systems and have been used for many purposes such as resource allocation, service coordination, process mining, model checking, behavior simulation, and state space analysis (SSA; Jensen, Kristensen, and Wells, 2007). For these reasons, PN are widely used in different domains such as business process management (BPM; Murata, 1989; van der Aalst, 2002), human computer interaction (HCI; Tran, Ezzedine, and Kolski, 2008) or pervasive computing (Tang, Guo, Dong, Li, & Guan, 2008).

In these fields, context adaptivity has emerged as a new perspective to offer more flexible and competitive systems that can adapt their behavior to changing circumstances or contexts. Users and organizations as well as their software systems currently operate in dynamic environments where context changes frequently. However, the existing PN formalisms do not provide reliable modeling, simulation and verification techniques that can accurately consider the system's execution context and adapt it in order to reflect the system execution reality.

According to Dey (2001) context is any information that can be used to characterize the situation of an entity; whether an entity is a person, place, or object that is considered relevant to the interaction between a user and a system, including the user and the system themselves. Therefore, different information is considered as context according to the specific domain of the system to develop. For instance, temporal, environmental, product, and organizational information has been typically considered as context information in BPM (De La Vara, Ali, Dalpiaz, Sánchez, & Giorgini, 2010).

A few extensions have been proposed to the basic PN formalism to deal with certain shortcomings in expressiveness that can help toward the introduction of context adaptivity in PNs (Genrich & Lautenbach, 1981; Jensen et al., 2007). For instance, data extensions allow the incorporation of data variables that can be used in guards or parameters inside a PN. Also, time extensions can enable modeling behavior in accordance to temporal restrictions. However, these extensions are not enough to properly support adaptation to the execution context. The following main limitations must be addressed:

- Limited data modeling expressiveness: Data types are restricted to the creation of simple data structures and offer limited expressiveness for data modeling, e.g., no support for inheritance.

* Corresponding author. Tel.:+3216376579.
E-mail addresses: estefania.serralasensio@kuleuven.be, fani.serral@gmail.com (E. Serral), Johannes.DeSmedt@kuleuven.be (J. De Smedt), Monique.Snoeck@kuleuven.be (M. Snoeck), Jan.Vanthienen@kuleuven.be (J. Vanthienen).

http://dx.doi.org/10.1016/j.eswa.2015.08.004

- No support for data reuse and persistency: Data types and variables are internally defined in the nets, therefore they cannot be reused for different systems, i.e., context should be redefined for each application even if it is in the same domain and shares the same context data. In addition, it is not possible to store either the data values and therefore to detect a context change (i.e., comparison between previous and new values).
- Limited data use: The data instantiation needs to be provided through tokens manually specified in the net as input for a transition. A data value is strictly local with respect to the transition where it is used (Bartkevičius, Kragnys, & Šarkauskas, 2006); to be able to use a certain context value, it needs to be passed across the net until it is used. Also, if any context data are instantiated in a net and is needed in another net, an explicit link between the two nets needs to be created to pass the data.

A few approaches have been also proposed to specifically support context adaptation using PNs. However, as will be explained in detail in the next section, the research in this area is rather limited, usually proposed for specific domain areas and without providing support for system simulation and verification.

Thus, the contribution of this paper is fourfold:

- We present context-adaptive Petri nets (CAPNs), a formalism that can define PNs capable to represent context-adaptive behavior. CAPNs integrate the PN formalism that can better support context adaptation with the most appropriate context modeling technique. The integration maintains a separation of concerns: while context data can be represented with a high degree of expressiveness, the system behavior is still described using the powerful modeling and analysis techniques of PNs.
- We provide tool support for the modeling, simulation, and verification of CAPNs. We have extended one of the most well-known tools for managing PNs to provide (1) a context manager to deal with context data, and (2) a design and execution environment for CAPNs. The tool also shows the feasibility of the defined formalism and facilitates its use by the research and business community.
- We tackle the problem of dealing with advanced data structures. By making use of the extension framework presented in Westergaard (2013), we allow data to be semantically represented, which enables a better way to describe data than the inscription languages typically used (Mortensen, 2001; Westergaard, 2013).
- A case based evaluation that shows the successful application of CAPNs in three different representative cases in various domains.

The remainder of this paper is organized as follows. Section 2 gives an overview of the existing context representation techniques and PN formalisms, and explains related approaches that use PN for context adaptation. Section 3 proposes the CAPN formalism and Section 4 presents the tool support. Section 5 evaluates the proposed solution. Finally, Section 6 concludes the paper, discusses the presented contributions and explains future research lines.

2. Background and related work

This section gives an overview of the existing context representation techniques and PN formalisms and selects the most suitable ones to support CAPN. Afterward, the related literature regarding the use of PN for context adaptation is analyzed showing how the contributions of this paper will go beyond the state of the art.

2.1. Context modeling techniques

In order to support context adaptation, appropriate context modeling and management techniques must be used. Different techniques have been proposed up until now to capture context. The most popular ones are as follows (Perera, Member, Zaslavsky, & Christen, 2014; Ye, Coyle, Dobson, & Nixon, 2007):

- Key-value pairs: These are the most simple data structures for modeling contextual information. Key-value pairs are easy to manage, but cannot model complex context data.
- Markup languages: These languages represent context as a hierarchical data structure consisting of markup tags with attributes and content. Markup-based models use a variety of markup languages including XML.
- Graphical models: These models display context using graphical notations, like UML.
- Object-oriented models: These models encapsulate context on an object level and provide an easy mapping from real world context concepts to modeling constructs.
- Logic-based models: It defines context as facts, expressions and rules. The model defines the conditions on which a concluding expression or fact may be derived from a set of other expressions or facts.
- Ontology-based models: These models represent context with its intrinsic semantics using formal axioms and constraints.

An exhaustive recent comparison of techniques can be found in Perera et al. (2014). Although all of the presented techniques have advantages and disadvantages, ontologies are one of the best choices to model context (Baldauf, Dustdar, & Rosenberg, 2007; Chen, Finin, & Joshi, 2004; Perera et al., 2014; Ye et al., 2007) and are drawing more and more attention as a generic and explicit way to capture and specify context (Perera et al., 2014).

An ontology is a formal representation of a set of concepts within a domain and the relationships between those concepts. An ontology mainly contains the following elements:

- **Classes:** The relevant kinds of entities or concepts. A class usually refers to a collection or a category of objects sharing some common properties; e.g., the classes product and location.
- **Data property:** The property that distinguishes a class from other classes and has a basic type, such as int, string, time, etc.; e.g., name and age.
- **Object property:** The property that identifies a relation between two ontology classes, i.e., identifies how an object is connected to other object in an ontology, e.g., a product_location, which relates the product class and the location class.
- **Constraints:** Rules that must be satisfied for the elements they apply to; e.g., the cardinality of a certain property must be 1, the class A is subclass of the class B, the object property is_in (which relates one location object to other location object) is transitive (i.e., if a location X is in the Y location and the Y location is in the Z location, then, X is inside Z too), etc.
- **Individuals:** Instances or objects of the defined classes; e.g., the individual Dell_latitud_e6410_16 of the class product, with name Dell_laptop, and which product_location is office236, which is an individual of the location class. An individual can belong to multiple classes.

Ontologies guarantee a high degree of expressiveness, formality, and semantic richness, and exhibit prominent advantages for automatically inferring new knowledge about the current context. For instance, it is possible to infer the set of activities taking place in a specific location or the specific sellers that are currently available. In addition, ontologies provide many benefits for reusing context as well as facilitating the interoperability of different systems. As such, it is possible to reuse and easily integrate standard or widely-accepted ontologies, such as ontologies that define locations or products.

Due to these numerous benefits and their wide adoption for context-adaptive systems, CAPNs use ontologies in order to allow context-awareness and adaptation in PNs.

Please cite this article as: E. Serral et al., Context-adaptive Petri nets: Supporting adaptation for the execution context, Expert Systems With Applications (2015), http://dx.doi.org/10.1016/j.eswa.2015.08.004