Computer Standards & Interfaces 31 (2009) 528–538

Contents lists available at ScienceDirect

Computer Standards & Interfaces

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

A two-stage modelling architecture for distributed control of real-time industrial systems: Application of UML and Petri Net

Francesco Basile *, Pasquale Chiacchio, Domenico Del Grosso

DIIIE, Università di Salerno, Italy

ARTICLE INFO

Available online 9 April 2008

Keywords:
UML
Petri Net
Object-Oriented paradigm
Industrial automation

ABSTRACT

The use of UML in the design process of distributed automation systems is here proposed. UML is used to formally express system's requirements, model the uncontrolled system and design the controlled one. It is here proposed a domain transformation: to go from the informal description to UML and from UML to PN models. In this way the capability to describe the system behavior is fully exploited, while the system analysis can be more properly performed in PN domain.
Furthermore, it is to show the possibility of conferring intelligence to real objects, even to immaterial objects, so that they can cooperate to fulfil the desired tasks in a distributed plant. To illustrate the methodology a real case study is used.

1. Introduction

One important feature of modern automation systems is their distributed nature which allows to use low-cost simple devices but requires additional efforts to manage their coordination. Moreover, to improve productivity, the optimization of resources must be carried out on-line.

For both aspects, the availability of models of the uncontrolled system and of the control devices is a primary need. At this aim, in this paper it is shown that the Unified Modeling Language (UML) can play a crucial role.

UML is seen to be useful in several stages of the distributed control system's design, making it faster, cheaper and more effective (see [1]). Moreover, UML is an effective tool to derive models that can be used for simulation and design purposes (see [2–4]).

UML has been derived in the software industry to describe user's requirements and speed up the code generation process (see [5,6,2,7–9]). UML is a formal graphic tool used to describe the structure and the behavior of a software system conceived within the object oriented paradigm. It has been developed in order to provide a design tool to visually express concepts which are common to object oriented programming ("a picture is worth a thousand words" — [10]). Commercial developing systems are available to translate UML diagrams directly into code for the most popular object oriented programming languages.

Other works have already demonstrated that UML can also be used to model production phases in industrial systems. As a matter of fact, object oriented models can be used to represent production processes when:

- real systems are complex;
- relations among parts cannot be expressed through mathematical laws;
- many process components (parts, machines, process) have
- common characteristics;
- the process is distributed;
- commercial software is unable to capture complex system dynamics.

In the real word it is usual to start a model building process from an informal description of the components of a system and of the interactions among them in terms of events, states and constraints. It could be useful to have a tool to build a representation of all its feasible behaviors and the set of rules that automate the translation of behaviors into a different operational domain, since behavioral traces are not very useful as analysis or control synthesis framework. It is here proposed a behavioral specification based on the event sequences of a process, whereas the operational domain is the PN formalism. Thus in building complex model we have two steps: from the informal description to UML and from UML to PN models (domain transformation).

In the following the reader is assumed to be acquainted with Petri Nets (PNs). For further details, it can be referred to [11].

PNs [11] are widely used to model and analyze industrial systems. The reasons are their formal semantics, graphical nature, expressiveness, the availability of analysis techniques to prove structural properties (invariance properties, deadlock, liveness, etc.) and the possibility to define and evaluate performance indices (throughput, occupation rates, etc.). Nevertheless building PN models of complex systems is a not easy and error prone task. Moreover, since the configuration of industrial systems may change during the system lifetime, it is also important to define a modeling approach which

* Corresponding author.
E-mail address: fbasile@unisa.it (F. Basile).

0920-5489/$ – see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.csi.2008.03.021

promotes component reuse. In this case it could be useful to combine an automatic model synthesis with a modular approach.

Furthermore, in PN domain [12] the controller synthesis paradigm can be adopted from control theory for continuous systems. Given a model of the plant dynamics and a specification for the desired closed loop behavior, the objective is to synthesize a controller to achieve the specifications. In this approach there is a clear distinction between the plant and the controller and the information flow between the plant and controller is modeled explicitly.

Finally, i.e. using UML to model industrial systems, it is possible conferring intelligence to real objects, even to immaterial objects. As a matter of fact, real objects are part of a distributed reality but usually have no skill. If intelligence is provided to them, they become actors of the control system thus cooperating to fulfill the requirements.

UML is a good mean to realize this idea, by modeling the real objects (material or even immaterial) as *Classes*[1], bringing them into an Object-Oriented framework. The methods of the classes model the way real objects interact to realize the control system.

To illustrate the idea reference to a real case study will be made throughout the paper.

In particular, the energy management of an industrial system is considered. It will be shown that the introduction of the "smart energy" object will help designing a management system to monitor and manage energy consumers thus optimizing power consumptions of the plant and saving energy. With reference to the case study, by modeling energy as classes, it is possible to declare the behavior that they have to assume in the Distributed Control System (DCS), defining a set of attributes, methods, relationships and associations, and thus make themselves intelligent components of their respective systems.

2. Background

In this section the basic concepts of UML useful in this paper are briefly recalled. The reader can find further details in [13,14,1].

UML is a visual language for specifying, modeling, designing and documenting a computer system, based on the Object-Oriented paradigm. UML is an Object Management Group (OMG) standard consisting of a collection of diagrams (see Fig. 1) that offer multiple model views of a system.

Several open source (Visual Paradigm suite [15], Star UML [16], Eclipse [17]) or commercial (Altova® [18], IBM®Rational®Software Modeler [19]) UML tools exist. The environment chosen in this work is the open source StarUML. A sreenshot is reported in Fig. 2.

UML allows analyzing the application's requirements and designing a solution that meets them, independently from the rendering to a specific technological platform. UML is not a programming language but thanks to specific tools it is often possible to automatically obtain data structures starting from a UML diagram.

In UML each model corresponds to a specific aspect of reality and through the combination of all diagrams a complete description of the entire system is provided. UML's Object-Oriented nature makes it particularly suited to the modeling of real world objects and aspects. The concept of *Object* is indeed borrowed just from the real world, thus UML can describe, in a formal and standard way, a concrete system or a productive process.

Fig. 1 shows the complete taxonomy of UML diagrams.

Diagrams are subdivided in:

- Structure diagrams;
- Behavior diagrams including Interaction diagrams.

The main difference among these two categories is how they describe reality. Structure diagrams show it from a static point of view and relationship among diagram's entities are constant in time.

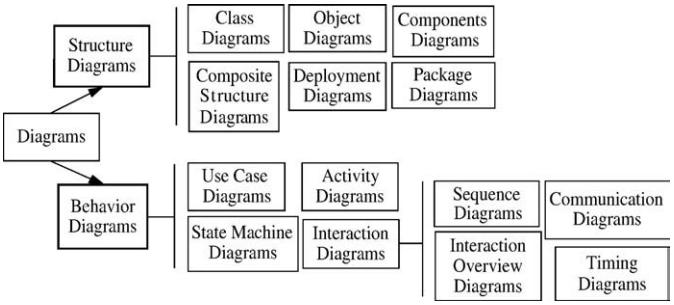

Fig. 1. UML diagrams taxonomy.

Behavior diagrams represent how the described process flows through components, users and in general through the system. Interaction diagrams show communication and temporal properties of the system, from an operational point of view. Both behavior and interaction diagrams describe the system's dynamics.

In the following a brief description of some structural and behavior UML diagrams is proposed.

Class diagrams. A class diagram is used to describe parts of a system, static relationship between them, their attributes, methods or functionalities and their belonging to subsystems.

Use case diagrams. A use case diagram contains actors of the system, use cases and actions that they can perform. It shows the functional relationship existing among them and with other functionalities of the system.

State diagrams. State diagrams are used to show the state changes of the system and the achievement of certain situations, usually at a class level.

Activity diagrams. The activity diagrams are usually used to analyze the behavior of complex use cases and to show interactions among them.

Sequence diagrams. Sequence diagrams show temporal interactions among actors and objects, which occurs through messages, to make clear the control flow into the system. They model the behavior of a system at a high level, at which only instance of classes (objects) and actors exist, to document how specific use cases are solved.

In Fig. 3 several connectors, which are kinds of relationship among elements of diagrams, are shown with a short description.

3. UML in automation

UML is a non-proprietary modeling and design tool that offers an Object-Oriented modeling framework. In automation methodological aspects of UML have been used within several applications for modeling as well as simulation and implementation purposes: modeling and implementation of object relational databases for the traceability in batch process [20]; specification and validation of scheduling policies in agile production systems [21]; modeling and implementation of production control systems [2]; design and implementation of simple network management protocol agents for remote control [8]; modeling and optimization of DCSs based on the industrial bus CAN [4]; modeling and validation of mechatronic systems [22]. The Object-Oriented approach is usually considered a basic principle for many modeling, analysis and design methodologies in various engineering areas, and UML is considered a "drastic way to revolutionize and improve the efficiency of software development process", allowing its easy modification, extension and maintenance [23]. Moreover it is used to support the *Metamodeling*[2] of reality and the Model-Driven development of the control system. UML's multiple

[1] *A Class* is a programming language construct that is used to group related instance variables and methods.

[2] While a model is a simple abstraction of the real world, a *Metamodel* is an abstraction too, which includes properties and methods for the model itself. Through a *Metamodel* it is possible to express additional semantics of existing information and specify methods and functions.