Chapter 10
Trying Out Reflective Petri Nets on a Dynamic Workflow Case

Lorenzo Capra
Università degli Studi di Milano, Italy

Walter Cazzola
Università degli Studi di Milano, Italy

ABSTRACT

Industrial/business processes are an evident example of discrete-event systems which are subject to evolution during life-cycle. The design and management of dynamic workflows need adequate formal models and support tools to handle in sound way possible changes occurring during workflow operation. The known, well-established workflow's models, among which Petri nets play a central role, are lacking in features for representing evolution. We propose a recent Petri net-based reflective layout, called Reflective Petri nets, as a formal model for dynamic workflows. A localized open problem is considered: how to determine what tasks should be redone and which ones do not when transferring a workflow instance from an old to a new template. The problem is efficiently but rather empirically addressed in a workflow management system. Our approach is formal, may be generalized, and is based on the preservation of classical Petri nets structural properties, which permit an efficient characterization of workflow's soundness.

INTRODUCTION

Business processes are frequently subject to change due to two main reasons (Aalst & Jablonski, 2000): i) at design time the workflow specification is incomplete due to lack of knowledge, ii) errors or exceptional situations can occur during the workflow execution; these are usually tackled on by deviating from the static schema, and may cause breakdowns, reduced quality of services, and inconsistencies.

Workflow management facilitates creating and executing business processes. Most of existing Workflow Management Systems, WMS in the sequel (e.g., IBM Domino, iPlanet, Fujisu iFlow, Team-Center), are designed to cope with static processes. The commonly adopted policy is that, once process changes occur, new workflow templates are defined and workflow instances are initiated accordingly

DOI: 10.4018/978-1-60566-774-4.ch010

from scratch. This over-simplified approach forces tasks that were completed on the old instance to be executed again, also when not necessary. If the workflow is complex and/or involves a lot of external collaborators, a substantial business cost will be incurred.

Dynamic workflow management might be brought in as a solution. Formal techniques and analysis tools can support the development of WMS able to handle undesired results introduced by dynamic change. Evolutionary workflow design is a challenge on which lot of research efforts are currently devoted. A good evolution is carried out through the evolution of workflow's design information, and then by propagating these changes to the implementation. This approach should be the most natural and intuitive to use (because it adopts the same mechanisms adopted during the development phase) and it should produce the best results (because each evolutionary step is planned and documented before its application).

At the moment evolution is emulated by directly enriching original design information with properties and characteristics concerning possible evolutions. This approach has two main drawbacks: i) all possible evolutions are not always foreseeable; ii) design information is polluted by details related to the design of the evolved system.

In the research on dynamic workflows, the prevalent opinion is that models should be based on a formal theory and be as simple as possible. In Agostini & De Michelis, 2000 process templates are provided as 'resources for action' rather than strict blueprints of work practices. May be the most famous dynamic workflow formalization, the ADEPTflex system (Reichert & Dadam, 1998), is designed to support dynamic change at runtime, making at our disposal a complete and minimal set of change operations. The correctness properties defined by ADEPTflex are used to determine whether a specific change can be applied to a given workflow instance or not.

Petri nets play a central role in workflow modeling (Salimifard & Wright, 2001), due to their description efficacy, formal essence, and the availability of consolidated analysis techniques. Classical Petri nets (Reisig, 1985) have a fixed topology, so they are well suited to model workflows matching a static paradigm, i.e., processes that are finished or aborted once they are initiated. Conversely, any concerns related to dynamism/evolution must be hard-wired in classical Petri nets and bypassed when not in use. That requires some expertise in Petri nets modeling, and might result in incorrect or partial descriptions of workflow behavior. Even worst, analysis would be polluted by a great deal of details concerning evolution.

Separating evolution from (current) system functionality is worthwhile. This concept has been recently applied to a Petri net-based model (Capra & Cazzola, 2007b), called Reflective Petri nets, using reflection (Maes, 1987) as mechanisms that easily permits separation of concerns. A layout formed by two causally connected levels (base-, and meta-) is used. the base-level (an ordinary Petri net) is unaware of the *meta-level* (a high-level Petri net).

Base-level entities perform computations on the entities of the application domain whereas entities in the meta-level perform computations on the entities residing on the lower level. The computational flow passes from the base-level to the meta-level by intercepting some events and specific computations (*shift-up* action) and backs when the meta-computation has finished (*shift-down* action). Meta-level computations are carried out on a representative of the lower-level, called *reification*, which is kept causally connected to the original level.

With respect to other dynamic Petri net extensions (Cabac, Duvignau, Moldt, & Rölke, 2005; Hoffmann, Ehrig, & Mossakowski, 2005; Badouel & Oliver, 1998; Ellis & Keddara, 2000; Hicheur, Barkaoui, & Boudiaf, 2006), Reflective Petri nets (Capra & Cazzola, 2007b) are not a new Petri net