Adaptive workflow nets for grid computing

Adaptive Workflow Nets for Grid Computing*

Carmen Bratosin, Kees van Hee, and Natalia Sidorova

Department of Mathematics and Computer Science
Eindhoven University of Technology
P.O. Box 513, 5600 MB Eindhoven, The Netherlands
c.c.bratosin@tue.nl, k.m.v.hee@tue.nl, n.sidorova@tue.nl

Abstract. Existing grid applications commonly use workflows for the orchestration of grid services. Existing workflow models however suffer from the lack of adaptivity. In this paper we define Adaptive Grid Workflow nets (AGWF nets) appropriate for modeling grid workflows and allowing changes in the process structure as a response to triggering events/exceptions. Moreover, a recursion is allowed, which makes the model especially appropriate for a number of grid applications. We show that soundness can be verified for AGWF nets.

Keywords: workflows, Petri nets, grid computing, coordination, modeling, verification.

1 Introduction

The notion of workflow appeared first in the world of enterprize information systems, where the execution of business processes is divided over several components, each with its own task. One of these components is a workflow engine that takes care of the control flow only. This separation of concerns is very fruitful and allows designers to prove (partial) correctness of the designed system.

Almost all the existing grid applications currently also use the idea of workflow to model processes. From the grid point of view, a workflow is a mean for the automation of processes, which involves the orchestration of a set of grid services, agents and actors that must be combined together to solve a problem or to define a new service [5]. The most common model used for grid workflows is the Directed Acyclic Graph (DAG). Although DAGs are intuitive for process descriptions, their modeling power has limitations (e.g. they does not support loop patterns and does not allow dynamic process changes driven by events happened in the system).

In [6], we introduced Adaptive Workflow Nets (AWF nets), an extension of workflow Petri nets [2,3] with the nesting concept [10]. AWF nets allow to include dynamic process changes and a fault handling mechanism into a model without forcing the user to get into implementation details. In this paper we

* This research is supported by the GLANCE NWO project "Workflow Management for Large Parallel and Distributed Applications".

V. Malyshkin (Ed.): PaCT 2007, LNCS 4671, pp. 15–21, 2007.

define Adaptive Grid Workflow nets (AGWF nets), a subclass of AWF nets appropriate for modeling grid workflows. AGWF nets allow changes in the process structure as a response to triggering events/exceptions (adaptivity). They make use of a pattern library, which easies reusability. Exception transitions are used as a solution to the robustness problem. Moreover, a (restricted form of) recursion is allowed, which makes it especially appropriate for a number of grid applications.

An important correctness property of workflow nets is soundness [2,3], which means that each computation *can* always terminate without leaving garbage[1]. In this paper we show that soundness can be checked for AGWF nets.

Related work. The advantages of the use of colored Petri nets for modeling grid workflows are considered in [9]. Tokens represent there real data and the net is used to model the interactions between different software resources. Similar graph representations can be found in [4,11]. Neither one however considers flexibility and adaptivity aspects.

The rest of the paper is organized as follows. In Section 2 we give basic definitions. In Section 3 we introduce the notion of adaptive grid workflow nets and formulate the soundness criterium for them. In Section 4 we discuss the obtained results and indicate directions for future work.

2 Preliminaries

$\mathbb{N}$ denotes the set of natural numbers. A *bag (multiset)* M over a set P is a mapping $M\colon P \to \mathbb{N}$. The set of all bags over P is also denoted by $\mathbb{N}^P$. We use $+$ and $-$ for the sum and the difference of two bags and $=, <, >, \leq, \geq$ for comparisons of bags, which are defined in the standard way. We overload the set notation, writing $\emptyset$ for the empty bag and $\in$ for the element inclusion. We write e.g. $M = 2[p] + [q]$ for a bag M with $M(p) = 2$, $M(q) = 1$ and $M(r) = 0$ for all $r \in P \setminus \{p, q\}$.

A *Petri net* is a tuple $N = \langle P, T, F, l\rangle$, where: (1) P and T are two disjoint non-empty finite sets of *places* and *transitions* respectively, elements of the set $P \cup T$ are called *nodes* of N; (2) $F \subseteq (P \times T) \cup (T \times P)$ is a *flow relation* between places and transitions and conversely; (3) l is a labeling function for transitions mapping each $t \in T$ to some label $l(t) \in \Sigma$, where Σ is a finite set of labels.

Let $N = \langle P, T, F, l\rangle$ be a Petri net and $T' \subseteq T$. The *projection* $N_{|T'}$ *of* N *on* T' is the net $\langle P, T', F', l'\rangle$, where $F' = \{(x, y) | (x, y) \in F \wedge x, y \notin T \setminus T'\}$ and $l'\colon T' \to \Sigma$ with $l'(t) = l(t)$ for all $t \in T'$.

Markings are states (configurations) of a net interpreted as bags over P. A *marked net* is a tuple (N, M), where N is a net and M is its marking.

Given a node $n \in (P \cup T)$, the *preset* ${}^\bullet n$ and the *postset* $n^\bullet$ of t are the sets $\{n' | (n', n) \in F\}$ and $\{n'' | (n, n'') \in F\}$ respectively. We will say that a node n

[1] Note that soundness differs from the halting problem, which is the property that a computation *will* always terminate.

is a *source* node iff ${}^\bullet n = \emptyset$ and n is a *sink* node iff $n^\bullet = \emptyset$. A *path* of a net is a sequence $\langle x_0, \ldots, x_n \rangle$ of nodes such that $\forall i : 1 \leq i \leq n : x_{i-1} \in {}^\bullet x_i$.

We define the firing relation $\longrightarrow$ as $M + {}^\bullet t \xrightarrow{t} M + t^\bullet$ for any marking M and transition t. $M \xrightarrow{t}$ is an abbreviation of $\exists M' :: M \xrightarrow{t} M'$. For $\sigma = t_1 \ldots t_n$, we write $M \xrightarrow{\sigma} M'$ iff $M \xrightarrow{t_1} \cdots \xrightarrow{t_n} M'$. Next, $M \xrightarrow{*} M'$ iff $\exists \sigma :: M \xrightarrow{\sigma} M'$ and $\mathcal{R}(N, M)$ denotes $\{M' \mid M \xrightarrow{*} M'\}$, the markings of N reachable from M.

A *workflow net* is a Petri net with one initial (source) place i and one final (sink) place f and every place and transition of the net being on a directed path from the initial to the final place. The initial marking of a workflow net is $[i]$ and the (desired) final marking is $[f]$.

3 Adaptive Grid Workflow Nets

In this section we define *Adaptive Grid Workflow nets* (AGWF-nets) and formulate the soundness criterium for them. We start with introducing a notion of Extended Workflow nets (EWF-nets), which form the basis for AGWF-nets.

Extended Workflow nets [6,7] are an extension of Workflow nets [2,3] that simplifies the modeling of exceptions by making a clear distinction between normal termination and termination caused by an exception. When an exception occurs, it is observed by some upper layer, which handles it. The execution of the EWF net is then terminated.

We consider a partition of the set of transitions $T = T_e \cup T_n$, where T_e is the set of *exception transitions* and T_n is the set of non-exception transitions. The set Σ of labels is partitioned into $\Sigma_e \cup \Sigma_n$ accordingly.

Definition 1 (Extended workflow net). *A net $N = \langle P, T_e \cup T_n, F, l \rangle$ is an* extended workflow net (EWF net) *iff (1) the net $N_{|T_n}$ is a workflow net; (2) for all $t \in T_e$, $t^\bullet = \emptyset$, ${}^\bullet t \neq \emptyset$, and ${}^\bullet t \subseteq P \setminus \{f\}$; (3) for all $t \in T_e$, $l(t) \in \Sigma_e$, and for all $t \in T_n$, $l(t) \in \Sigma_n$.*

As usual, the state of the net is given by its marking. The initial marking consists of a single token on the initial place. The only change in the semantics w.r.t. the standard semantics of Petri nets is that exception transitions terminate the execution of the net.

We allow standard algebraic operations on EWF nets: Two (unmarked) nets can be combined to produce a new net by means of sequential ($\cdot$) and parallel ($\|$) composition and choice ($+$). Parallel composition can also be applied to marked nets, and sequential composition to a marked net and an unmarked net.

Adaptive workflow nets. In [6], we introduced a class of nets, called *adaptive workflow nets* (AWF nets), allowing more flexibility and adaptivity than existing workflow systems. By adaptivity we understand an ability to modify processes in a structured way as response to some triggering events, for instance by extending a process with a subprocess. In [7] we considered a non-recursive subclass of AWF nets from [6] that is well-suited for modeling business workflows and showed how

to verify their soundness using abstractions. Recursion is however essential for a number of grid applications. Here we describe a recursive subclass of adaptive workflow systems appropriate for grid applications for which soundness is still decidable.

Let $Var = \{v, \dots\}$ be a finite set of *variable* names and Con a finite set of *constant* names. We assume a given library of process descriptions to be used as basic building blocks for constructing more complex processes by using net expressions. A net expression e and a token expression te are inductively defined as: $e := c \mid e + e \mid e||e \mid e.e$, $te := \mathfrak{b} \mid ce$ and $ce := v \mid ce||ce \mid ce.e \mid init(e)$, where $v \in Var$, $c \in Con$. The sets of all net expressions and token expressions are denoted by $Expr$ and $CExpr$, respectively. The expressions in $Expr$ will be interpreted as adaptive workflow nets while the expressions in $CExpr$ denote either black tokens ($\mathfrak{b}$) or marked adaptive workflow nets. Given an expression $e \in CExpr$, the set of variables appearing in it is denoted $Var(e)$ and the set of constants in it is denoted by $Con(e)$.

Firings of the adaptive net can depend on firings in the net tokens, which is modelled by the guards of transitions expressed in the guard language $\mathcal{G}$. A guard g is defined as $g := \top \mid final(v) \mid e(v)$, where $v \in Var$ and $e \in \Sigma_e$. A guard $final(v)$ is called *termination guard* and $e(v) \in \mathcal{G}$ is called an *exception guard*. The set of all guards is denoted by $\mathcal{G}$. Intuitively, the guard $\top$ of a transition t means that the firing of t does not depend on the internal states of the net tokens, $e(v)$ means that the firing of t is conditioned by the firing of an exception transition with label e in the token net v, whereas $final(v)$ means that it is conditioned by the token net v having reached the final marking $[(f, \mathfrak{b})]$.

We define now nested workflow nets as extended EWF nets.

Definition 2 (Adaptive workflow net). *A* Adaptive Workflow net $\mathcal{N}$ *is a tuple* $\langle P, T, F, \mathcal{E}, g, l\rangle$*, where* $\langle P, T, F, l\rangle$ *is an EWF net called* system net *and the extensions* $\mathcal{E}, g$ *are defined by:*

- $\mathcal{E}\colon F \to CExpr$ *are* arc expressions *such that*
  1. *All input arcs for transitions are mapped either to the black token or to variables, i.e. for every* $(p,t) \in F$, $\mathcal{E}(p,t) \in Var \cup \{\mathfrak{b}\}$*;*
  2. *Every two variables on two different input arcs of a transition are distinct, i.e. for all* (p,t), $(p',t) \in F$ *with* $p \neq p'$, $Var(\mathcal{E}(p,t)) \cap Var(\mathcal{E}(p',t)) = \emptyset$*;*
  3. *Every variable on the outgoing arc of a transition also occurs in the expression of some incoming arc of this transition, i.e. for all* $(t,p) \in F$, $v \in Var(\mathcal{E}(t,p))$ *implies* $v \in Var(\mathcal{E}(p',t))$ *for some* $(p',t) \in F$*;*
  4. *All outgoing arcs of the initial place and incoming arcs of the final place are mapped to the black token, i.e. for all* $t \in i^\bullet$, $\mathcal{E}(i,t) = \mathfrak{b}$ *and for all* $t \in {}^\bullet f$, $\mathcal{E}(t,f) = \mathfrak{b}$*.*
- g *is a function that maps transitions from* T *to expressions from* $\mathcal{G}$ *such that the variable of a guard* $g(t)$ $(t \in T)$ *appears in the expression of some incoming arc of* t *and does not appear in any outgoing arc of* t*, i.e.* $Var(g(t)) \subseteq \bigcup_{p \in {}^\bullet t} Var(\mathcal{E}(p,t))$ *and* $Var(g(t)) \cap \bigcup_{p \in t^\bullet} Var(\mathcal{E}(t,p)) = \emptyset$*.*

For the sake of brevity, we define the semantics of AWF nets at an informal level. An *adaptive workflow net* can be seeing as a special colored EWF net (the system net), whose tokens can be either (marked) adaptive workflow nets themselves, called *token nets*, or black tokens. Transitions with *true* as a guard may fire if there are enough tokens on their input places, like in classical Petri nets. A transition t guarded by $\mathit{final}(x)$ may fire if there are enough tokens on its input places and the place connected to t by the arc with variable x contains a token net that has reached its final state $[(f, \mathfrak{b})]$. This token will then be consumed from p during the firing. A transition t guarded by $e(x)$ may fire if there are enough tokens on its input places and some transition with label e is enabled in a token net contained in the place connected to t by the arc with variable x. Again, it is this token that will be used in the transition firing. Note that since we require that the output arc expressions do not contain variables from the transition guard, the net token x gets destroyed. The output token nets are computed according to the corresponding arc expressions where variables are substituted by the token nets from the input places, participating in the firing.

Soundness. Soundness is an important property of adaptive workflow nets stating that at any moment of system run there is a chance to terminate properly by reaching the final marking, also when no exception occurs in token nets. We define soundness for adaptive nets as proper termination of every reachable marking by firing only non-exceptional transitions without synchronizing on exceptions:

Definition 3 (Soundness for AWF nets). *An AWF net $\mathcal{N}$ is called* sound *iff $\mathcal{N}$ is quasi-live, and for all M such that $[(i, \mathfrak{b})] \xrightarrow{\sigma} M$, for some transition sequence $\sigma \in T_n^*$, there exists σ' such that $M \xrightarrow{\sigma'} [(f, \mathfrak{b})]$, and for all t from σ', $t \in T_n$ and $g(t) \in \{\mathit{final}(v), \top\}$.*

In [7] we defined a non-recursive subclass of AWF-nets for which soundness can be algorithmically checked:

1. $\mathfrak{N}_1$ and $\mathfrak{M}_1$ are the sets of all EWF nets and marked EWF nets, respectively;
2. $\langle P, T, F, \mathcal{E}, g, l\rangle \in \mathfrak{N}_{k+1}$, for $k \geq 1$, iff for all $a \in F$ and $c \in \mathit{Con}(\mathcal{E}(a))$, $\ell(c) \in \mathfrak{N}_k$. A marking M of $N \in \mathfrak{N}_{k+1}$ is a multiset over $P \times (\mathfrak{M}_k \cup \{\mathfrak{b}\})$. $\mathfrak{M}_{k+1} \stackrel{\text{def}}{=} \{(N, M) | N = \langle P, T, F, \mathcal{E}, g, l\rangle \in \mathfrak{N}_{k+1} \wedge M \in \mathbb{N}^{P \times (\mathfrak{M}_k \cup \{\mathfrak{b}\})}\}$ is called the set of marked nets of level at most k.

Note that $\mathfrak{N}_j \subseteq \mathfrak{N}_{j+1}$ and $\mathfrak{M}_j \subseteq \mathfrak{M}_{j+1}$, for all $j \geq 1$.

Since we want to have at least a restricted form of recursion for grid applications and still have an analyzable class of models, we introduce a form of well-foundedness for the recursion in Adaptive Grid Workflow nets.

Let $\mathcal{N}$ be a given AGWF net. We define the net collection $\mathit{Coll}(\mathcal{N})$ of $\mathcal{N}$ as the union of the set of constants (nets) used on the arc expressions of $\mathcal{N}$ and the net collections of these constant nets. The net collection of an AGWF net

can be computed by using standard fixed point algorithms. By inspecting the net collection, one can easily check whether a net belongs to $\cup_{j\in\mathbb{N}}\mathfrak{M}_j$.

Definition 4 (Adaptive Grid Workflow net). *An* Adaptive Grid Workflow net *(AGFW net) is an AWF net such that every net from $Coll(\mathcal{N})$ allows a firing sequence $[(i,\mathfrak{b})] \xrightarrow{\sigma} [(f,\mathfrak{b})]$ such that for any transition t from σ, we have $g(t) \in \{final(v), \top\}$ and for any $(t,p) \in F$, $Con(Expr(t,p)) \subseteq \cup_{j\in\mathbb{N}}\mathfrak{M}_j$.*

Note that the property required is checked at the level of EWF-nets, i.e. classical Petri nets, and not at the nested level. Intuitively, we require that there is at least one execution with *bounded nesting* allowed in every net involved in the process.

Now we show that soundness can be checked for AGWF nets. To reduce the verification of soundness to a finite problem, we introduce the abstraction α that replaces every token net in the AGWF net by a colored token with the set of exceptions of the net token as its color. An adaptive workflow net is thus abstracted by a colored EWF net whose color set is finite since the number of exceptions is finite. The guards of the type $final(v)$ are replaced by $\top$ in the abstract net, and the guards $e(v)$ are replaced by the guards $e \in \alpha(v)$. Parallel and sequential composition, as well as choice, are abstracted to the union of the sets of exceptions, and constants in the arc expressions are substituted by their sets of exceptions. Now we can formulate our main result:

Theorem 5 (Soundness check). *An AGWF net $\mathcal{N}$ is sound iff for every net $\mathcal{N}' \in Coll(\mathcal{N})$ the following properties hold: (1) $\alpha(\mathcal{N}')$ is quasi-live, and (2) for all abstract markings M_α reachable by firings of non-exception transitions in $\alpha(\mathcal{N}')$, i.e. $[(i,\mathfrak{b})] \xrightarrow{\sigma} M_\alpha$ with $\sigma \in T_n^*$, we have $M_\alpha \xrightarrow{\sigma'} [(f,\mathfrak{b})]$, where $g^\alpha(t) = \top$ for all $t \in \sigma'$.*

4 Conclusion

In this paper, we introduced adaptive grid workflow nets. Exceptions transition are used to model faults (e.g. failure of a job). The idea of nested nets is used to make models adaptable. A library of workflow nets is used to increase the reusability and achieve separation of concerns in process modeling. We showed that an important correctness property called soundness can be verified on this class of nets by using abstraction techniques. We conjecture that another important property of adaptive workflow systems called circumspectness[2] is also decidable for AGWF nets.

Our next step is to extend the workflow engine YASPER [8] for handling AGWF nets, and extend the existing translation of classical workflow nets to WS BPEL [1] for our model by incorporating the nesting mechanism and patterns for standard exception handling mechanisms.

[2] Circumspectness ensures that whenever an exception happens, the upper layer net is able to handle it.

References

1. Web Services Business Process Execution Language Version 2.0. WS-BPEL TC OASIS (2005) http://www.oasis-open.org/committees/download.php/11601/
2. van der Aalst, W.M.P.: The Application of Petri Nets to Workflow Management. The Journal of Circuits, Systems and Computers 8(1), 21–66 (1998)
3. van der Aalst, W.M.P., van Hee, K.M.: Workflow Management: Models, Methods, and Systems. MIT Press, Cambridge (2002)
4. Deelman, E., Blythe, J., Gil, Y., Kesselman, C., Mehta, G., Patil, S., Su, M.-H., Vahi, K., Livny, M.: Pegasus: Mapping scientific workflows onto the Grid. In: Dikaiakos, M.D. (ed.) AxGrids 2004. LNCS, vol. 3165, pp. 11–20. Springer, Heidelberg (2004)
5. Fox, G.C., Gannon, D.: Workflow in Grid Systems. Concurrency and Computation: Practice and Experience 18(10), 1009–1019 (2006)
6. van Hee, K., Lomazova, I.A., Oanea, O., Serebrenik, A., Sidorova, N., Voorhoeve, M.: Nested nets for adaptive systems. In: Donatelli, S., Thiagarajan, P.S. (eds.) ICATPN 2006. LNCS, vol. 4024, pp. 241–260. Springer, Heidelberg (2006)
7. van Hee, K., Lomazova, I.A., Oanea, O., Serebrenik, A., Sidorova, N., Voorhoeve, M.: Checking properties of adaptive workflow nets. In: CS&P 2006 - Concurrency 2006, Specification and Programming, 27-29 September 2006, Germany, pp. 92–103 (2006) (An extended version is to appear in Fundamenta Informaticae)
8. van Hee, K., Oanea, O., Post, R., Somers, L., van der Werf, J.M.E.M.: Yasper: a tool for workflow modeling and analysis. In: ACSD, pp. 279–282. IEEE Computer Society, Los Alamitos (2006)
9. Hoheisel, A.: User tools and languages for graph-based Grid workflows. Concurrency and Computation: Practice and Experience 18(10), 1101–1113 (2006)
10. Lomazova, I.A.: Modeling dynamic objects in distributed systems with Nested Petri nets. Fundamenta Informaticae 51(1-2), 121–133 (2002)
11. Oinn, T.M., Addis, M., Ferris, J., Marvin, D., Senger, M., Greenwood, R.M., Carver, T., Glover, K., Pocock, M.R., Wipat, A., Li, P.: Taverna: a tool for the composition and enactment of bioinformatics workflows. Bioinformatics 20(17), 3045–3054 (2004)