Using Graph of Relations to Construct Coordination Mechanisms for Workflows

Dennis PELLUZI [a,1] , Léo P. MAGALHÃES [a]

[a] *Department of Computer Engineering and Industrial Automation, School of Electrical and Computer Engineering, State University of Campinas*

Abstract. This work approaches the problem of coordination of activities in workflow systems. Workflow systems are characterized by a composite of several interdependent activities with a common goal. In order to guarantee the correct execution of activities, a coordination mechanism is required. One of the difficulties in designing coordination mechanisms is to guarantee that such mechanisms are consistent with the specification of the workflow. Some works suggest the use of modeling tools such as Petri Nets or coordination languages to construct a coordination mechanism. The Graph of Relations methodology, namely GR methodology, is one of them. This paper uses the extended GR methodology to construct coordination mechanism for workflow systems.

Keywords. Coordination mechanism, workflow systems, Petri net, graph.

Introduction

A workflow system is composed of a set of activities that are related. If there are dependences among activities, for example, temporal relations, then there is the necessity of a coordination mechanism [1]. This coordination mechanism has to guarantee the restrictions imposed on the execution of activities.

Regarding Internet, we have Service-Oriented Computation. In this architecture, software is seen as a service and an application web is composed of several services [2]. In other words, each service is seen as a basic block of construction of an application (services composition). The services composition and the Internet allow Inter-organizational collaboration among activities through Inter-organizational workflow. In order to make it possible, it is necessary to coordinate activities. One of the difficulties to design workflow systems, with interdependent activities, is the construction of the coordination mechanism. The designer must know the behavior of activities and the relations among them. Some authors like [1], [3] and [4] approach the management of dependences among activities under different applications.

[1] Corresponding author: E-mail: pelluzi@dca.fee.unicamp.br

1. Coordination

Some works use Petri Net-based models of coordination, for example, [4], [5] and [6]. Models based on Petri Nets (classic or extended) [7] are chosen because they have mathematical support for analysis and simulation of activities behavior. Some of these works consider specific coordination mechanisms for a class of application, for example, multimedia. While others, as [4], consider independent coordination mechanisms which can be used for many classes of application.

Some works ([8] and [9]) explore the use of coordination languages. Such languages are specific of a determinate issue (coordination of concurrent processes or construction of collaborative applications) and aim to assist the programmer. Therefore, the coordination languages have similar syntax to the programming languages. However, coordination languages have low abstraction level and do not support verification techniques.

In a simple way, coordination is the effort to guarantee that the parts of an environment work together, without conflicts, in order to achieve a common goal. According to Malone [1], coordination is the management of dependences among activities. If there is not interdependence among activities then coordination is not necessary.

In the context of this work, activities are interdependent if they are related to themselves. In other words, the execution of an activity depends of the execution of others in some way. If the execution of activities leads to a common objective, then the activities are collaborative.

Service-Oriented Computation (software as a service) has been an increasing interest in software engineering. Services can be executed in different and distributed platforms. Each service is seen as a basic block of construction of applications (services composition). Web service is the most important realization of that concept. A web service (WS) provides specialized service for other WS. Thus, we can construct a complex application using web services composition.

Services composition is the main interest in applications development using web services, [10]. The services composition allows the Inter-organizational collaboration among activities (Inter-organizational workflow). As pointed by van der Aalst *et al* [11], there are several languages of WS composition such as WSFL, WSCI, WS-Coordination, BPML, XLANG and BPEL4WS. However, these languages provide different techniques for web service composition without a solid coordination theory [12].

2. Extended GR Methodology

Cruz [13] proposed a methodology, namely Graph of Relations (GR) methodology, which allows expressing graphically and analytically temporal interdependences among activities in a computational environment. The GR methodology has three abstraction levels for modeling systems. In the first level (specification level – L1), it defines the behavior of the system by specifying the relations among activities. In the second level (coordination level – L2), a coordination mechanism (CM) is constructed. Coordination Mechanisms are artifacts used to guarantee the dynamic behavior according to the specification of system established in the L1 level. Finally, in the level

L3 (execution level) a program called coordinator implements the CM constructed in the L2 level.

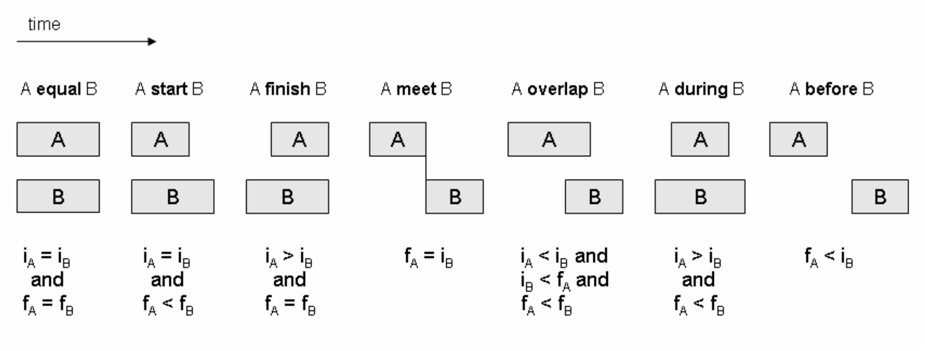
time
A **equal** B
A **start** B
A **finish** B
A **meet** B
A **overlap** B
A **during** B
A **before** B
A
B
$i_A = i_B$ and $f_A = f_B$
$i_A = i_B$ and $f_A < f_B$
$i_A > i_B$ and $f_A = f_B$
$f_A = i_B$
$i_A < i_B$ and $i_B < f_A$ and $f_A < f_B$
$i_A > i_B$ and $f_A < f_B$
$f_A < i_B$

Figure 1. The seven primitive mutual-exclusion relationships between two intervals.

The type of behaviors specified in initial version of GR methodology was the temporal one. The behavior of a system is described in terms of temporal relations among activities, or more specifically, temporal intervals. According to J. Allen [14], there are seven primitive mutual-exclusion relationships between two time intervals. These relationships form the set D = {e, s, d, f, o, m, b}.

The letters e, s, d, f, o, m and b indicate, respectively, the relations equal, start, during, finish, overlap, meet and before (Figure 1). Letters *i* and *f* respectively represent the instant points of beginning and end of an interval.

2.1. Specification Level

The temporal relationships are expressed through a direct labeled graph, called graph of relations (GR). One activity is represented by a vertex and the relation between two activities is represented by an edge. The edges of the graph have a label that defines the type of interdependence. In order to avoid timed inconsistencies, the graph must not have cycles with vertices which represent activities. A formal proof is presented in PhD thesis of Cruz [15].

The extended GR methodology allows specifying resources [16]. The resources also are represented by vertices. The resource vertices have a label (t, n) where *t* indicates the type of the resource (volatile or non-volatile) and *n* indicates the number of available instances.

A resource is volatile (v) if its number of instances decrease after an activity use it and it is not-volatile (nv) otherwise. The direction of edges indicates if the activity uses the resource (resource $\rightarrow$ activity) or if the activity produces the resource (activity $\rightarrow$ resource). The edges have a numerical value that indicates how many instances of the resource are produced or consumed by the activity.

Formally, the graph of relations is defined by expression E(A, R, F, G, S, P, w, u) where:

A is a set of vertices representing the activities;

$R \subset A \times A$, is a set of edges representing the relations;

$F{:}R \rightarrow D$ is a function, called edge labeling function that associates a non-empty subset of D with each edge.

$D = \{e, s, f, d, o, m, b\}$ is the set of primitive temporal relations;

$G{:}A \rightarrow A$ is a function that associates a subset of A with each element of A.;

S is a set of vertices representing the resources;

$P \subset (A \times S \cup S \times A)$ is a set of edges associating a resource with one or more activity and vice-versa;

$w{:}P \rightarrow \mathbb{N}$ is a function that sets a integer value for each edge of set P;

$u{:}S \rightarrow T \times \mathbb{N}$ is a function that associates a ordered pair (t, n) with each element of set S where $t \in T = \{nv, v\}$ and $n \in \mathbb{N}$.

The function G indicates alternative relationships for one given activity, i.e., the activity in question relates with only one activity of the label defined by G. Figure 2 shows the graphical representation of expression E1(A, R, F, G, S, P, w, u) defined below. To facilitate the understanding of the graph, the vertices of set S (resources) are represented by squares while the vertices of set A (activities) are represented by circles.

$A = \{a_1, a_2, a_3, a_4, a_5, a_6, a_7\}$

$R = \{(a_2, a_3), (a_2, a_4), (a_1, a_5), (a_5, a_6), (a_7, a_6)\}$

$F(a_2, a_3) = F(a_2, a_4) = \{e\}, F(a_1, a_5) = \{b\}, F(a_5, a_6) = \{e, s\}, F(a_7, a_6) = \{f\}$

$G(a_2) = \{a_3, a_4\}, G(a_1) = G(a_3) = G(a_4) = G(a_5) = G(a_6) = G(a_7) = \emptyset$

$S = \{s_1, s_2\}$

$P = \{(a_1, s_1), (s_1, a_2), (s_2, a_5), (s_2, a_7)\}$

$w(a_1, s_1) = w(s_2, a_5) = 2, w(s_2, a_7) = w(s_1, a_2) = 1$

$u(s_1) = (nv, 2), u(s_2) = (v, 5)$

2.2. Coordination Level

In the coordination level, Petri Nets (PN) are used to model the behavior of activities. Each activity is represented by a pair of transitions, t_{Ia} and t_{Fa}, and three places, I_a, E_a and F_a (Figure 3). The fire of transitions t_{Ia} and t_{Fa} indicates the beginning and the end of the execution of the activity, respectively. One token in the place I_a represents a request to start the activity. One token in the place E_a indicates that the activity is running and one token in the place F_a indicates that the activity is finished.

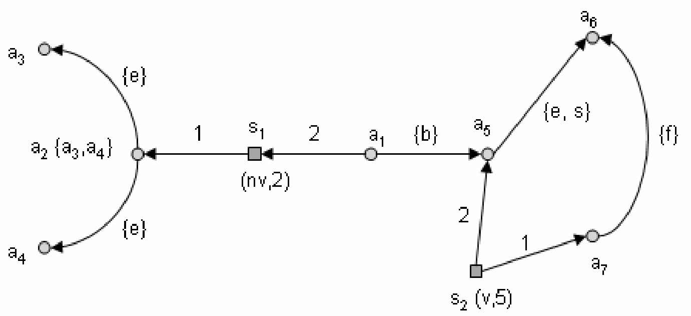

Figure 2. Graph of the expression E1.

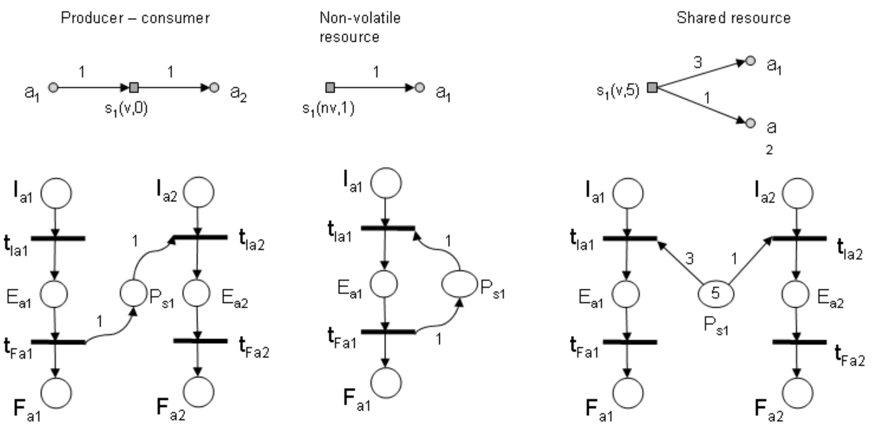

Figure 3. Petri Net-based model of activities and resources.

A resource is represented by one place and the number of tokens in this place indicates the amount of available instances. Figure 3 illustrates Petri Net-based models of activities and resources.

The representation of the temporal relationships in Petri Nets is constructed inserting constraints on the fire of transitions t_{Ia} and t_{Fa}. A timed constraint is defined by equalities and inequalities involving the initial and final instants of activity (Figure 1). These equalities or inequalities are translated into PN model according to rules 1 and 2.

Rule 1. Inequalities x < y or x > y, where x and y represent the initial or final moments in the execution of the activities involved, are translated into PN by adding an arc from the transition that corresponds to the least-value variable to place P_Z, $z \in \mathbb{N}$, and an arc from P_Z to the transition that represents the largest value. For example, in relation a_1 *before* a_2, we have $f_{a1} < i_{a2}$, which means including an arc from t_{Fa1} (which represents f_{a1}) to place P_2 and an arc from P_2 to t_{Ia2}, which represents i_{a2} (Figure 4-b).

Rule 2. Adding an equality to the Petri Net consists in performing a merge operation of the transitions associated to the equation variables (Figure 4-a).

The Petri net model is constructed using an algorithm described in [13].

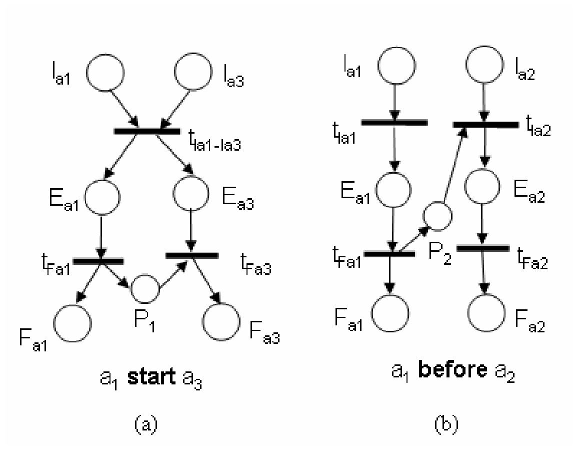

Figure 4. Petri Net model of temporal relations.

2.3. *Execution Level – the Coordinator*

The construction of the coordinator is done translating the model of coordination mechanism to a software component which is capable to communicate with activities of the application. This communication is done through an interface that defines the operations which an activity must satisfy to communicate with the Coordinator (Figure 5). The separation between the coordination mechanism and activities allow modifying the execution of activities without having to change the coordination mechanism. We can also modify the dependences among activities modifying the coordination mechanism without having to modify the activities. This separation makes the coordinator independent of the type of application. The activities are interdependent units of execution in an application that need authorization of the Coordinator to be executed.

The Coordinator controls the execution of the activities, authorizing or not their beginning or end, based on the Petri Net model. Each activity has one transition that authorizes its beginning or end, as seen previously. If this transition is enabled, then the activity is authorized to start or finish. Otherwise, the activity will have to wait for an authorization.

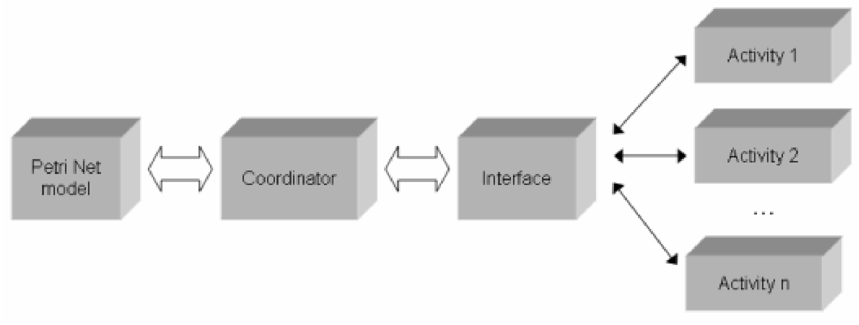

Figure 5. High Abstraction Level of coordination.

3. Workflow Systems

In this section, we will introduce the use of extended GR methodology to design workflow systems. The GR methodology is not a complete model of specification of workflow, because it does not describe all workflow patterns [17]. However, some of these patterns can be implemented through the GR methodology. Figure 6 shows some examples.

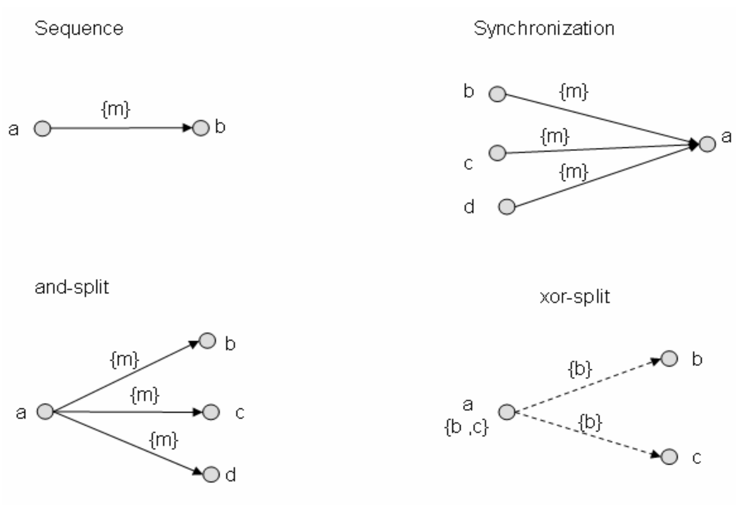

Figure 6. Some workflow patterns.

Considering the Reference Model of the Workflow Management Coalition [18], the use of level L1 of GR methodology corresponds to the definition process. That is, the graph of relations describes the logic of the business process in a high abstraction level of notation. The levels L2 and L3 correspond to the execution level of workflow, they are responsible for the administration of the process, distribution and invocation of the activities. Based on these concepts, we can use the extended GR methodology to coordinate a workflow system, for example, web services-based workflows.

The Petri net model (Figure 3) indicates that the workflow stops if the available resources are not enough for the execution of one or more activities. In this case, it is necessary to allocate more instances of the resource, if it is possible. The exception handling in workflow system is not aimed by this work, but we can refer the work of Kumar and Wainer [19] which deals with this issue.

4. Conclusion

In workflow processes, we have activities which compete for resources or that produce resources which will be used by other activities. We also have temporal relationships. In such situations, it is necessary to have a coordination mechanism which manages the use of resources and the order of executions. We have presented an extension for the GR methodology and its application in workflow systems. This extension allows the GR methodology to lead with dependence of resource. Although the extended GR methodology is not a complete model of specification of workflow, it can deals with the most common workflow patterns.

The specification of temporal relations and resources, which is used and/or produced by activities, is done through a direct labeled graph (graph of relations). Based on this graph, we construct a PN model of coordination which can be used by the coordinator (a software component). The coordinator verifies in the model if an activity can be executed. This communication is done through signals that are sent between the activities and the coordinator.

One of the justifications to use Petri Nets-based models is the possibility to simulate and to analyze the system. With the analyses, the system designer can evaluate if the activities will be executed according to imposed restrictions.

References

[1] T. W. Malone and K. Crowston. The Interdisciplinary Study of Coordination. *ACM Computing Surveys* **26**(1), (1994), 87–119.

[2] M. P. Papazoglou. Service-Oriented Computing: Concepts, Characteristics and Directions. *Proceedings of the Fourth International Conference on Web Information Systems Engineering* (WISE'03), (2003).

[3] K. Schimidt and C. Simone. Coordination Mechanisms: Towards a conceptual foundation of CSCW systems design. *Computer Supported Cooperative Work: The Journal of Collaborative Computing* **5**(2-3), (1996), 155–200.

[4] A. B. Raposo, A. J. A. da Cruz, C. M. Adriano and L. P. Magalhães. Coordination Components for Collaborative Virtual Environments. *Computers & Graphics* **25**(6), (2001) 1025-1039.

[5] P. Hsu, Y. Chang and Y. Cheg. STRPN: A Petri-Net Approach for Modeling Spatial-Temporal Relations between Moving Multimedia Objects. *IEEE Transc. on Software Engineering* **29**, (2003).

[6] T. Gonsalves, K. Itoh and R. Kawabata. Use of Petri Nets in the Performance Design and Improvement of Collaborative Engineering Systems. *Integrated Design and Process Technology*, IDPT, (2004).

[7] T. Murata. Petri Nets: properties, analysis and applications, *Proceedings. of the IEEE* **77**(4), (1989), 542–580.

[8] M. Cortes. A Coordination Language for Building Collaborative Applications. *Computer Supported Cooperative Work*, Kluwer Academic Publishers, (2000).

[9] G. Ciobanu and D. Lucanu. A Specification Language for Coordinated Objects. *Proc. of the 2005 conference on Specification and verification of component-based systems*, ACM Press, (2005).

[10] S. Dustdar and W. Schreiner. A Survey on Web Services Composition. *Int. J. Web and Grid Services*, **1**(1), (2005).

[11] W. M. P van der Aalst, M. Dumas and A. H. M ter Hofstede. Web Service Composition Languages: Old Wine in New Bottles? *Proc. of the 29th EUROMICRO Conf. on New Waves in System Architecture*, Los Alamitos, CA, (2003).

[12] S. K. Prasad and J. Balasooriya. Fundamental Capabilities of Web Coordination Bonds: Modeling Petri Nets and Expressing Workflow and Communication Patterns over Web Services. *Proc. of the 38th Hawaii International Conference on System Sciences*, (2005).

[13] A. J. Cruz, L. P. Magalhães, A. B. Raposo, R. S. Mendes and D. G. Pelluzi. Coordinating Multi-task Environments Through the Methodology of Relations Graph. *Proc. of the 13th International Workshop on Groupware* (CRIWG 2007), *Lecture Notes in Computer Science* **4715**, Springer, (2007), 127-142.

[14] J. F. Allen. Maintaining Knowledge About Temporal Intervals. *Communications of the ACM* **26**(11), (1983), 832-843.

[15] A. J. Cruz. *Grafo de Relações : Uma Metodologia Para Coordenar Dependências entre Atividades em Ambientes Computacionais*. Phd thesis, FEEC, University of Campinas, Brazil, (2004).

[16] D. G. Pelluzi. *Modelagem e Construção de Mecanismos de Coordenação em Ambientes Computacionais*. MS. Dissertation, FEEC, State University of Campinas, Brazil, (2007).

[17] W. M. P van der Aalst, A. H. M. ter Hofstede, B. Kiepuszewski and A. P. Barros. Workflow Patterns. *QUT Technical report*, FIT-TR-2002-02, Queensland University of Technology, Brisbane, (2002).

[18] Workflow Management Coalition (WfMC) *The Workflow Reference Model* – TC00 – 1003 v1.1. http://www.wfmc.org/standards/referencemodel.htm, (1995).

[19] A. Kumar and J. Wainer. Meta Workflows as a Control and Coordination Mechanism for Exception Handling in Workflow Systems. *Decision Support Systems* **40**, (2004), 89–105.