A Tool Based on Relations Graph Methodology to Model Coordination Mechanisms

Dennis G. Pelluzi
pelluzi@dca.fee.unicamp.br

Léo Pini Magalhães
leopini@dca.fee.unicamp.br

Technical Report 01-2008

April, 2008
Summary

Index of Figures and Tables ...........................................................................................................3
1. Introduction ...................................................................................................................................4
2. Relations Graph Methodology ......................................................................................................4
3. Modeling Tool ...............................................................................................................................7
   3.1 Class diagrams .........................................................................................................................10
4. Petri Net Markup Language .........................................................................................................12
5. Coordinator ....................................................................................................................................13
5. Example of using ..........................................................................................................................14
6. Concluding Remarks ....................................................................................................................17
7. References ......................................................................................................................................18
## Index of Figures and Tables

Fig. 1: An example of partition P(A) ................................................................. 6  
Fig. 2: The sequence of steps to model coordination mechanisms .................. 8  
Fig. 3: Graphical user interface of CMAM ...................................................... 8  
Fig. 4: Procedure to build the coordinator ..................................................... 10  
Fig. 5: Class Diagram of relation graphs ...................................................... 10  
Fig. 6: Class diagram of Petri nets ............................................................... 11  
Fig. 7: Class diagram of partition of activities ............................................ 12  
Fig. 8: Class diagram of PNML format ....................................................... 13  
Fig. 9: Coordination system ....................................................................... 14  
Fig. 10: Relation graph of the example ....................................................... 16  
Fig. 11: Colored Petri net of the animation process with the initial marking ....... 16  

Tab. 1: Primitive moving for a walking animation ......................................... 15
1. Introduction

The aim of this report is to present the prototype of a tool which supports the modeling of coordination mechanisms using the Relations Graph methodology (RG). The tool developed, called Coordination Mechanism Automatic Modeler (CMAM), builds the model of the coordination mechanism of a using Petri net. The model of the coordination mechanism is obtained from the specification in a graph of activities and their temporal relations. Additionally, the CMAM allows include resources and its relationships with activities. The inclusion of resources in the specification of the modeling process is an extension of the RG methodology.

The tool was developed using the Java platform J2SE 1.5 and has a graphical user interface for insertion of input data (relations graph). Once applied the modeling algorithm, the program provides a file describing the Petri net. This file is an instance of PNML (Petri Net Markup Language), which is a standard based on XML to describe Petri nets. The file can be used in Petri nets simulators to verify and validate the coordination mechanism and after that it can be used to build the coordinator. Coordinator is a software which interacts with the target application (activities) and it is responsible for managing the dependencies between activities.

The next section of this report presents the RG methodology shortly. In the following sections the computational structure of the modeling tool is presented and an example is illustrated. Finally, the conclusions are presented.

2. Relations Graph Methodology

When one designs a system which involves interdependent activities, a difficulty is to ensure that all constraints imposed for the execution of activities will be respected. This difficulty is due to constraints arising from the indirect relationships between the activities. To cope with this issue on temporal aspect, Cruz [Cruz, 04] proposed a methodology, called Relations Graph (RG). This methodology uses Colored Petri nets to model a coordination mechanism which ensures that activities will be executed without breaking the constraints. Furthermore, the RG methodology uses a linear complexity algorithm to automate the modeling of the coordination mechanism.

The RG methodology allows to express graphically and analytically temporal relations of
interdependent activities of a computational process. From the specification of activities and their relationships a model of coordination mechanism is obtained.

The RG methodology works on three levels of abstraction to model systems. At the specification level (N1), the system is defined through the temporal relations of activities. At the coordination level (N2), a coordination mechanism (CM) is built. Coordination mechanism is a structure responsible for ensuring the dynamic behavior of activities as defined at the specification level N1. Finally, at the level N3 (implementation level) a software called coordinator should implement the CM built at the level N2.

The system which will be modeled is described in terms of temporal relations among activities, more specifically, relations between the time spans of activities. The relations between two time intervals used by RG methodology are the same relations presented by Allen [Allen, 83] (equal, start, finish, before, during, overlap and meet). The specification is done through directed labeled graphs, where each node represents one activity and each edge represent one temporal relation between two activities.

The next step after the specification of the process is to build the complete model of coordination. To reach this goal, the GR methodology uses an algorithm which has complexity O(n), where n is the number of activities. The graph of a specification must be a tree. Such condition is imposed by the GR methodology and it aims to ensure the non existence of time inconsistencies [Zaidi, 99]. Based on this characteristic of the graph, it can classify the activities at different levels through the Algorithm 1 and as illustrated in Figure 1.

Algorithm 1: Partitioning graph.

Let \( I_k \) be the set of activities, \( k \in \mathbb{N}; \)
\[ k \leftarrow 0; \]
While there be activities which do not belong to a set \( I_k \), do:
  Remove from the graph all activities which have degree equal one\(^1\) and add them in set \( I_k; \)
  Update the degree of all remained activities;
  \[ k \leftarrow k + 1; \]
End while
End algorithm

---

\(^1\) Degree of an activity \( a \), denoted by \( \partial(a) \), is the number of direct relations of the activity, i.e., it is the number of adjacent nodes of the activity.
In the end, it have a set of different levels of activities, denoted by $P(A)$, whose elements are the sets $I_0, I_1, ..., I_k, k \geq 0$. A feature of the set $P(A)$, is that activities of the same level are not related to each other and activities of levels $I_k$, for $k > 0$, are related to at least one activity of the level $I_{k-1}$.

The next step after specifying activities and its relations is to build the complete model of coordination. To do that, the RG methodology uses the Algorithm 2.

**Algorithm 2:** Modeling the coordination mechanism

Obtain the partition set $P(A)$;

For each set $I_k$, $k \geq 1$, belongs to partition set $P(A)$ do

For each activity of set $I_k$ do

Identify and model the temporal constraints obtaining the local coordination mechanism (LCM).

Connect\(^2\) the LCM of activity $a_i, a_i \in I_{k-1}$, with the LCM of activity $a_j \in I_{k-1}$.

End for

If the last set $I$ of iteration has exactly two activities, then connect the LCM of the two activities.

End for

End algorithm

Details and a full explanation of the Algorithm 2 are outside of the scope of this report and it can be found on doctorate thesis [Cruz, 04].

Some types of non-temporal relationship, for example, a producer-consumer relation, can

\(^2\) Connect means to join two LCMs that have a common activity.
also be shaped using the GR methodology. In this case, one can use the temporal relationships 
meet or before to represent this type of interdependence. However, if the resource is not 
produced by an activity, then this form of representation is not appropriate.

Another quite common type of dependence among activities in computational processes is 
the dependence on resources. There are basically three types of dependence on resources: 
sharing, simultaneity and volatility [Raposo, 00]. With the exception of simultaneity, which can 
be represented by the temporal relationship equal, the other two types of management resources 
are not covered by the RG methodology.

An extension to the GR methodology was introduced in [Pelluzi, 07]. This extension 
allows the inclusion of dependences on resources. Resources used by activities are represented 
by vertices, which are different those vertices representing activities. Each resource vertex has a 
label \((t, n)\), where \(t\) is the type of resource according to its volatility and \(n\) is the number initial 
available instances. An resource is volatile \((v)\) if it is not longer available after one uses it and 
an resource is non-volatile \((nv)\) otherwise. The direction of the edges that connect one activity 
to one resource indicates either the activity uses that resource (resource to activity) or if the 
activity produces that resource (activity to resource). Those edges have a numerical value which 
indicates how many instances of resource are produced or consumed by the activity associated 
with it.

3. Modeling Tool

The purpose of the modeling tool is to automate the building of the coordination mechanism. 
This application, called CMAM (Coordination Mechanisms Automatic Modeler), generates 
from the specification in relations graphs, a Petri net which models the coordination mechanism 
of activities. This model is described in a XML file [XML, 06] using the Petri Net Markup 
Language (PNML) [Billington et al, 03]. The main idea is that from the Petri net model the 
designer can simulate and analyze the performance of the modeled process before implementing 
it.

Figure 2 shows the sequence of steps to generate the coordination mechanism through the 
CMAM. The entry of input data (relations graph) in CMAM is done through a graphical user 
interface. The user builds the relations graph entering activities and resources and indicating the 
relationships between them into a desktop.
The user interface was made using the library Jgraph [Jgraph, 06] version 5.8.2.2. This library allows the design and editing of graphs in an windows environment. The Jgraph library uses components of Java Swing [Loy et al, 02]. Figure 3 illustrates the user interface with an example.

![Diagram of the sequence of steps to model coordination mechanisms.](image1)

**Fig 2:** The sequence of steps to model coordination mechanisms.

![Graphical user interface of CMAM.](image2)

**Fig. 3:** Graphical user interface of CMAM.
The option for a graphical interface is to facilitate the use of the tool. One could provide the relation graph by other means, for example, through a text file.

Once it provide the relation graph, the CMAM verifies the existence of cycles which may indicate temporal inconsistency. If there is some temporal inconsistency, then the designer will be asked to review temporal relationships among activities.

After the verification of cycles and not having any one, the CMAM generates the partitioning of the graph (Algorithm 1) which will be used by the modeling algorithm (Algorithm 2). The latter algorithm builds the model of the coordination mechanism considering the temporal constraints and the dependences of resources.

The XML translator (Figure 2) is responsible for generating the file which describes the Petri net. The XML translator is extensible and can be used to generate different formats of XML to describe a Petri net. At this stage of development, the output files are generated in PNML (section 4) and CPN (which is used in the CPN Tools simulator [CPN, 07]) formats.

The use of CMAM is the first phase to build the coordinator. The complete procedure is illustrated in Figure 4. The CMAM application is responsible for the specification of the process and the modeling of the coordinator. The PNML output file is used by a Petri net simulator to verify and validate the model. Using the simulation, we can see if the model attend the problem of coordination. If there is some inconsistence, then we have to return to CMAM and review the specification of the process. After that all, the PNML file can be used to build the coordinator.
13.1 Class diagrams

The conceptual class diagram that implements a relation graph is shown in Figure 5.

Fig. 5: Class Diagram of relation graphs.
The class Graph is a composite of objects of the class Node. The class Node is an abstract class, therefore it can not be instantiated. Thus, to create a node object it has to be used either the class Task or the class Resource, both are subclass of Node.

The class Task has an association with the class TemporalRelation, which means each task object has one or more temporal relations with another activity. Each object of class TemporalRelation has exactly two activities, which are the extremes of an edge (sourceTask and targetTask). The Resource class has the attribute type which indicates either the object is a volatile resource or a non-volatile resource. The associations between the classes Task and Resource indicate which resources an object activity produces and/or consumes. This model does not allow classes have an edge linking a resource to other one, because that such relationships is not specified by the RG methodology.

The class diagram of relation graphs is used in CMAM to perform steps 1 and 2 in Figure 4.

The class diagram that implements a Petri net is shown in Figure 6.

![Class diagram of Petri nets.](image)

Fig. 6: Class diagram of Petri nets.
The classes Object and PNNode are abstract, which implies to instantiate only object of its subclasses Arc, Transition and Place. Herein the Petri net is seen as a directed graph, which nodes are either transitions or places. This representation is similar to that used by the PNML format that will be discussed forward.

Before to build the coordination mechanism it is necessary to obtain the partition of activities following the Algorithm 1. A partition of activities is composed by n levels where each one contains one or more activities (Figure 7). This partition is necessary for the modeling algorithm (Algorithm 2).

![Diagram](image)

**Fig. 7: Class diagram of partition of activities.**

### 4. Petri Net Markup Language

In order to facilitate the transfer of Petri net models among different tools, it was necessary to define a standard file format for exchange. An initial proposal to establish an international standard for high-level Petri nets occurred in 1995. In 2000, the first proposals were based on XML as a Petri Net Markup Language (PNML) [Jüngel et al, 00]. The PNML standard has been adopted by many tools that use Petri nets because it is flexible enough to integrate different types of Petri nets and it is open to future extensions [Billington et al, 03].

The standard PNML considers a Petri net as a directed and labeled graph where all information is represented in labels. A label can be associated with a node, a arc or a net. Figure 8 shows the basic structure of PNML. It should be noted that the UML diagram representation of Figure 8 does not define the XML syntax for PNML documents.
A document that meets the requirements of PNML is called a Petri Net Document (PetriNetDoc). This may contain several Petri nets, which has an identifier and a defined type. The definition of labels of a Petri net uses the syntax of Document Convention, which allows a tool to recognize the structure of the Petri net.

As shown in Figure 8 each object has a label that can be either an annotation or an attribute. A annotation label is used to display an information near the object, while a attribute label defines a feature of a object. Each object and annotation also have information about his position space, and optionally, its shape, color, size and font.

![Class diagram of PNML format.](image)

Fig. 8: Class diagram of PNML format.
5. Coordinator

Once the PNML that represents the model of the coordination mechanism has been generated, it begins the construction of the coordinator. The construction of the coordinator is done translating the model of the coordination mechanism obtained in a software component that is able to communicate with the applications activities. This communication is done through an interface that allows to encapsulate coordinator details of the implementation of activities.

The separation between the coordination mechanism and activities allows, for example, changing the implementation of activities without having to change the coordination mechanism. It can also change the dependences among activities without having to change the activity itself. This separation allows the coordinator to be independent of the type of the coordinated application. Activities are interdependent units of execution in an application and require authorization from the coordinator to be executed. Figure 9 illustrates in a high abstraction level the coordination system.

Fig. 9: Coordination system.

5. Example of using

A computational animation system contains, among other components, a control mechanism that is responsible for generating the desired action. There are different techniques for animation control found in literature. N. M. Thalmann and D. Thalmann [Thalmann, 91] proposed three categories of Movement Control Method (MCM) based on informations provided to control the movement of actors: geometric, physical and behavioral.

The behavioral MCM specifies the movement of an actor describing the desired behavior.

---

3 This approach also is presented in [Raposo, 01].
4 An actor is a scene object that will be animated.
This method allows the animator to specify the desired action and leave the system of animation work to create the animation. It can use this method of control with simple activities and use them together to create a more complex animation. In this example, it wants to create the walking movement of a human figure from simple animations such as the movements of its arms and legs.

From the idea of how a human figure moves, the animator describes this movement using temporal relations among activities to move the legs and arms.

Activities $a_0$ ... $a_7$, described in Table 1, represent simple animations and they will be used to compose the whole process of moving. The temporal relationship among activities is inserted graphically into CMAM.

<table>
<thead>
<tr>
<th>Activity</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a_0$</td>
<td>Lifts the right leg.</td>
</tr>
<tr>
<td>$a_1$</td>
<td>Moves the right leg forward and descend it.</td>
</tr>
<tr>
<td>$a_2$</td>
<td>Lifts the left leg.</td>
</tr>
<tr>
<td>$a_3$</td>
<td>Moves the left leg forward and descend it.</td>
</tr>
<tr>
<td>$a_4$</td>
<td>Moves the left arm forward.</td>
</tr>
<tr>
<td>$a_5$</td>
<td>Moves the right arm forward.</td>
</tr>
<tr>
<td>$a_6$</td>
<td>Moves the left arm backward.</td>
</tr>
<tr>
<td>$a_7$</td>
<td>Moves the right arm backward.</td>
</tr>
</tbody>
</table>

The human walking is conceptually well known. In this type of movement, the legs move forward and backward, providing support to the body, and the arms accompanying the movement of the legs. The temporal relations among activities reproduce the sequence of steps of this movement. Some constraints must be imposed as do not allow both legs are in the air at the same time as well as do not lift a leg more than once consecutively. Figure 10 illustrates the temporal relation graph.
Once the user has entered the graph through CMAM as illustrate in Figure 10, the software builds a model of the coordination mechanism and saved it in a file called mc_model.cpn. This file contains a description of Petri net using the CPN format, which is a specific markup language (XML) of CPN Tools.

Using this file, it can simulate the coordination mechanism. Figure 11 illustrates the initial state of the animation process using colored Petri net. In this figure, the inscription 1’a next to places indicates that there is a token of color "a". Each activity is represented by the places I_{ai}, E_{ai}, and F_{ai} (i = 0, 1, ..., 7). A token in E_{ai} place indicates that the activity a_i is running.
Initially, there is one token 1’a in each Iai place, which indicates that activities are requesting approval to start. However, only the activity a0 is allowed to start because the transition preceding the place Ea0 is the only enabled. Only after the end of a0, other activities will be allowed to start. The simulation indicates that activities are executed respecting the temporal constraints specified in the relations graph. For example, there is not a situation where the activities a0 and a2 are executed at the same time (situation in which the two legs are raised).

The whole process finished when all activities are completed correctly. In this case, timeout could be used to restart the process.

This example illustrates the steps used to develop a coordination mechanism for a computational animation. The activities can be reused for other types of animation such as the action of jumping or running, simply changing the temporal relations among activities. The advantage of this approach is that the animator can create from a simple animation set different scenes without having to recreate all activities.

6. Concluding Remarks

This report has presented the prototype of a coordination mechanism modeling tool. The procedure to build the coordination mechanism is divided into three phases: specification, simulation and construction of the coordinator. The application CMAM developed in this work converts a graph representation of dependences among activities (specification level) into Petri
net models of coordination. We decided to use an open standard representation of Petri net to facilitate the exchange of files among different applications that use Petri net, although there are some differences in PNML files used by applications of different developers, because the PNML standard is still in development.

To do the simulation, we use a free available simulator, which is the CPN Tools [CPN, 07]. There are several options for Petri net simulators, free or not, that meet different needs. The designer can use the tool that is more appropriate for his work.

We have also provided an example to illustrate the use of the tool. This example showed how the CMAM tool speeds up the construction of the Petri net model of coordination mechanisms and how its simulation allowed to analyze the performance of the modeled process.

7. References


