Transformation of Decisional Models into UML: Application to GRAI Grids

Reyes Grangel Seguer, Michel Bigand, Jean-Pierre Bourey

Transformation of Decisional Models into UML: Application to GRAI Grids

Journal:	*International Journal of Computer Integrated Manufacturing*
Manuscript ID:	TCIM-2009-IJCIM-0109.R1
Manuscript Type:	Special Issue Paper
Date Submitted by the Author:	26-Feb-2010
Complete List of Authors:	Grangel Seguer, Reyes; Universitat Jaume I, Dept. de Llenguatges i Sistemes Informàtics Bigand, Michel; Univ Lille Nord de France Bourey, Jean-Pierre; Univ Lille Nord de France
Keywords:	ENTERPRISE MODELLING, MODELLING
Keywords (user):	MODEL TRANSFORMATION, UML Profile

Transformation of Decisional Models into UML: Application to GRAI Grids

María de los Reyes Grangel Seguer [a], Michel Bigand [b;c] and Jean-Pierre Bourey [b;c*]

[a] *Grupo de Investigación en Integración y Re-Ingeniería de Sistemas (IRIS), Dept. de Llenguatges i Sistemes Informàtics, Universitat Jaume I, 12071 Castelló, Spain;*
[b] *Univ Lille Nord de France, F-59000 Lille, France;*
[c] *ECLille, LM2O, F-59650 Villeneuve d'Ascq, France*

* Corresponding author. Email: jean-pierre.bourey@ec-lille.fr

(Received 26 October 2095; final version received 25 February 2010)

Model Driven Engineering (MDE) significantly improves the software development process as well as enterprise integration. Based on both modelling and transformation activities it is possible to link models at different levels of abstraction in a classical forward engineering process in order to produce code or, in a reverse engineering process, to produce models from code by a bottom-up approach. But it can also be used to bridge the gap between models belonging to different domains that use different formalisms.
This paper therefore focuses mainly on the transformation of decisional models belonging to the Enterprise Modelling domain into UML models used in the Information Technology domain. This kind of transformation is one component of a more general model-driven approach to solve business process interoperability problems or, more widely still, integration problems. More precisely, the paper describes two UML Profile definitions used to transform GRAI Grids into UML Use Case Diagrams or Activity Diagrams. The implementation of these Profiles with the Atlas Transformation Language (ATL) is presented and results are compared and discussed.

Keywords: model transformation; GRAI decisional model; UML profile

Introduction

Enterprises must fit their functions and processes to those of others in order to improve their competitiveness and to take advantage of new market opportunities. Both Business Process and Enterprise Modelling techniques and methods have been successfully used by enterprises to integrate their information and manufacturing systems throughout the last few decades (Vernadat 1996). However, the rapid evolution of technology means that enterprises need to advance ever faster, and they must establish collaborative networks, such as Extended and Virtual Enterprises, so as to be able to take advantage of their core competences and to look for others that they do not have (Tae-Young *et al*. 2006).

New problems, such as Business Process Integration or Interoperability, arise in collaborative enterprises. Solving these interoperability problems not only at the code level but also starting from a higher level of abstraction is a challenge that was one of the objectives proposed by INTEROP NoE (INTEROP 2007), and more precisely by Task Group 2 (TG2), in order to search for solutions to achieve interoperability following a model-driven approach (Grangel *et al*. 2005, Chen and Doumeingts 2003).

The aim of TG2 has been to analyse and propose guidelines and methods that can help to solve the interoperability problems of Enterprise Software Applications (ESA) starting out from the enterprise models level and using an approach based on Model Driven Architecture (MDA) (OMG 2003). This method is called Model Driven Interoperability (MDI) (Grangel *et al*. 2006). The work of TG2 first focused on the models and transformations to be performed at the Computation Independent Model (CIM) level from the theoretical point of view. At this level the GRAI[1] method (Doumeingts et *al*. 1993, Berio 2003) was chosen as means to capture the enterprise models and the UML was selected to play an interface role between enterprise models and IT models. To address the feasibility of its proposal, TG2 has tested it with transformation tools. More precisely, from the Business Process Modelling point of view, TG2 first worked on model transformation from GRAI Extended Actigrams to UML Activity Diagrams (Bourey *et al*. 2006).

The paper focuses on the transformation of the decisional model which is an important model at business level. It is organised as follows: section 2 describes the context of the study and section 3 gives an overview of model transformation concepts. In Section 4 the basic constructs of GRAI Grids are presented and two first

[1] GRAI is a French acronym that can be translated into Graph showing Interrelations between Results and Activities

mappings with semantic losses are described and discussed. Then, in Section 5 two UML profiles to avoid semantic losses are defined and they are implemented within a transformation tool presented in Section 6. Finally, Section 7 outlines the main conclusions.

2. Context of the Study

Model Driven Engineering focuses on *models* as the primary artefacts in the development process, with *transformations* as the primary operation on models which is used to map information from one model to another (Bézivin 2004). Model Driven approaches aim to provide large scope solutions to improve software development processes. These kinds of approaches can also be useful for performing model transformations in the Enterprise Modelling context as well as for contributing to solve interoperability problems (Jardim-Gonçalves *et al*. 2006).

Enterprise Modelling is perceived as a prerequisite to achieve integration, and it can also help to solve interoperability problems starting from a higher level of abstraction than code. It is defined by (Vernadat 1996) as the art of externalising enterprise knowledge and know-how, which represents the Enterprise in terms of its organisation and functions (activities, information, resources and organisation units, and system infrastructure and architecture). It is also the process of drawing up models of the whole or part of the enterprise that include its different functions.

Enterprise Modelling is achieved through by using Enterprise Modelling Languages (EML). In this context, there are several formalisms which deal with Enterprise

Modelling, such as GRAI (Doumeingts et *al*. 1993, Berio 2003), CIMOSA (Berio and Vernadat 1999), PERA (Williams 1993), IDEF (IDEF 2007), and so forth. The GRAI methodology, which was created and developed by the research group LAP/GRAI at the University of Bordeaux 1, is a well-known Enterprise Modelling methodology. One of its strengths is that it takes into account both the decisional and the functional aspects together, as well as the informational and business process aspects. All these aspects are considered from both a general and a local point of view. **GRAI Grids** are defined to model the global functional and decisional aspects and GRAI Nets are used for local modelling of decision processes. **GRAI Extended Actigrams** are dedicated to Business Process Modelling. The GRAI methodology has been acknowledged and approved (by the IFIP and the IFAC) as one of the three important Enterprise Modelling methodologies together with CIMOSA and PERA.

However, one of the main weaknesses of EML is the difficulty involved in establishing strong links between enterprise and software models (Grangel *et al*. 2005). This paper addresses one part of this issue (more precisely the Decisional aspects) by defining a transformation of GRAI Grids.

On the other hand, **UML**[2], which has been successfully used to model and develop IT systems in different domains, can also be useful in the context of Enterprise Modelling (Marshall 2000, Eriksson and Penker 2000). It is the most widely known Object Management Group™ (OMG™) adopted specification, as an object-oriented modelling language used to model applications in the context of Software Engineering. For these reasons, UML is a good candidate to establish links between the contexts of Enterprise Modelling and Software Engineering and, therefore, to fill the gap between these two domains.

3. Model Transformations

The objective is to transform a source model *Ma* into a target model *Mb*. One of the most commonly used techniques for model transformation is known as the *Metamodel Approach* (OMG 2003) which is based on the *Model Transformation Pattern* shown in Figure 1. In this approach, the first step consists in defining the source and target metamodels (resp. *MMa* and *MMb*) which define the languages used to build the models (resp. *Ma* and *Mb*). Each model conforms to its metamodel. Then a mapping (*Tab*) between the metamodels is built. This process consists in establishing correspondences between the constructs in each metamodel. This mapping can be defined as a simple table showing the construct matching. For example, in (Cuenca *et al*. 2006) a table for the mapping of UML use cases or DFD onto CIMOSA can be

[2] Model Driven Architecture, MDA, Object Management Group, OMG, UML and XMI are either registered trademarks or trademarks of Object Management Group, Inc. in the United States and/or other countries.

found. This kind of table can be used as a set of specifications to be implemented by using more formal and executable languages (like XSL, general programming languages or transformation dedicated languages such as ATL (Jouault *et al.* 2006)). In this case the language that is used conforms to its metamodel *MMt*. By using an executable language it is possible to perform the transformation *Tab* from any input model *Ma* conforming to *MMa* in order to generate the corresponding target model *Mb* conforming to *MMb*. The transformation pattern in Figure 1 is obviously not limited to a one-to-one model transformation, but it can be adapted in a Model-weaving context (Milewski and Roberts 2005).

Figure 1. Model Transformation Pattern from (Allilaire *et al*. 2006)

In our study *MMa* is the GRAI Grid metamodel and will be described in the following section. *MMb* is the UML metamodel which is fully defined in (OMG 2009). Two target models will be considered: Use Cases and Activities. Simple mappings from GRAI Grids to Use Cases and Activities are presented in sections 5.2 and 5.3. Since these mappings introduce semantic losses, the target UML metamodel is extended in sections 5.6 and 5.7 by defining two dedicated Profiles. In this framework, we present in the following sections some experiences on model

transformations in order to understand how enterprises that use different Enterprise

Modelling formalisms can interoperate following a model-driven approach.

4. GRAI Grids

Within the GRAI methodology (Doumeingts *et al*. 1993, 1998, Berio 2003), three formalisms have been defined: **GRAI Grids** are defined to model the comprehensive functional and decisional aspects, and GRAI Nets are used for local modelling of decision making processes. **GRAI Extended Actigrams** are dedicated to Business Process Modelling. This paper addresses one part of the transformation of GRAI models and, more precisely, the transformation of GRAI Grids into UML models. Transformations of GRAI Extended Actigrams into UML Activity Diagrams with semantic losses was studied in (Grangel *et al.* 2007, Bourey *et al.* 2006) and a UML profile for preserving semantics was proposed in (Grangel *et al*. 2008).

GRAI Grids are intended to provide a general cartography of periodic decision making inside an enterprise. Thanks to a matrix representation, they emphasise the temporal scope of a decision as well as the functional analysis of the enterprise.

The **rows** define the **Decisional Levels** (for instance, Strategic, Tactical, Operational) and are characterised by both a **Horizon** and a **Period**. The Horizon defines the time limit for reaching the objectives. The Period is a subdivision of the Horizon and defines the period of time beyond which the decision could be reconsidered in order to reach the objective at the end of the Horizon.

The **columns** are macroscopic **Functions** of the enterprise. Two kinds of grids are proposed. The first one is made up of **Functional Grids** in which the Functions (i.e. columns) are the main functional activities of the enterprise such as *Product Management*, *Design Management*, *Sales Management*, and so forth. The second category of Grids consists of **Control Grids**, in which the three main functions are *To Manage Products*, *To Manage Resources* and *To Plan* (i.e. to synchronise Products and Resources). Each function can be broken down into **Subfunctions**: for example,

the function *To Manage Products* can be split into *To Manage Procurement* and *To Manage Materials*. Two additional columns representing the **External** and **Internal Information Sources** are added on each grid border. They are used as interfaces with both external and internal Information Systems as well as with Physical Systems or the environment. The intersection of a row and a column defines a cell called a **Decision Centre** where decisions are made. This model element is the central construct of this formalism. Two kinds of **Flows** are used to connect Decisional Centres:

- **Information Flows**: they are depicted by thin arrows and represent simple pieces of information needed by a Decision Centre.
- **Decision Frames**: they are depicted by broad arrows going from a Decision Centre (source) that imposes the Decision Frame on another Decision Centre (target). They represent the decisional information the target must take into account in order to make its own decisions. Therefore, they introduce a dependency between the source Decision Centre that imposes the Decision Frame on the target Decision Centre. A Decision Frame is made of at least:
    - **Objectives** to be reached by the target Decision Centre,
    - **Decision Variables** which are parameters that the receiver of the Decision Frame can act on in order to reach the objectives,
    - **Constraints** on the Decision Variables,
    - **Criteria** used to choose among the Decision Variables.

Figure 2 shows an example of a GRAI Grid. In order to describe how each decision is made, each Decision Centre is detailed by a **GRAI Net**. A GRAI net is an extended version of GRAI Extended Actigrams intended to provide a precise description of the process of decision-making.

Figure 2. Example of a GRAI Grid

An example of GRAI Net is given on Figure 3. This process is composed of both execution and decisions activities and must comply with the received decision frame in terms of objectives, decision variables, constraints and criteria. Execution activities like 'Compare achieved/budget' on Figure 3 are represented horizontally

while decision activities like 'Accelerate the orders in progress' are represented vertically. GRAI nets are therefore represented by using a process-oriented formalism. GRAI nets are not presented in greater detail in this paper but a thorough description of them can be found in (Doumeingts *et al.* 1998) and (Berio 2003).

Figure 3. Example of a GRAI Net

From this first informal description, the first task was to formalise the GRAI Grid modelling language by means of its Metamodel. The two main class diagrams of this metamodel are presented in Figure 4 and Figure 5. OCL constraints (OMG 2006) were defined to ensure the consistency of the formalism described by this metamodel.

Figure 4. GRAI Grid Metamodel: classes, generalisations and structural associations

Figure 5. GRAI Grid Metamodel: Nodes and Flows

5. GRAI Grid Transformation

5.1 Mapping Proposals

The objective is to establish a bridge from the Enterprise Modelling domain to the Software Engineering domain and, more precisely, from the comprehensive modelling of decision-making represented by means of GRAI Grids to another formalism belonging to the Software Engineering domain. Due to its broad coverage and use, UML was chosen as the target formalism for this study. The method to establish this bridge consists in:

1) **defining a first mapping** from GRAI Grids to UML by finding the basic concepts of UML that are the closest to the GRAI Grid concepts; since the central component of a GRAI Grid is the Decision Centre, the first step consists in choosing the UML target for the Decision Centre and then defining the other related UML targets in order to obtain a consistent model from the point of view of UML semantics;

2) **filling the semantic gap** between the source and target concepts in order to completely define a mapping;
3) **implementing the mapping** within a tool using a Model Transformation Language.

A first study on this mapping was proposed by (Darras 2004), but the results partially covered the first step and were based on UML 1.5. In the following, the three steps will be covered and the latest UML adopted specification will be used (i.e. UML 2.2 (OMG 2009)). In order to define a consistent mapping from GRAI Grids onto UML, two viewpoints can be considered. The first one involves taking the consistency of relations between modelling components into account: if a GRAI Grid source construct SC1 is mapped onto a target UML construct TC1, and SC1 is linked to a second source construct SC2, then SC2 must be mapped onto a UML target construct TC2 that is linkable to TC1 and complies with the UML metamodel. The second viewpoint consists in taking the purposes of the decision-making modelling into account and then finding the most appropriate UML constructs as targets for the mapping definition.

From the first viewpoint, to ensure the consistency between the relations throughout the mapping definition, a bottom-up approach can be considered starting with the mapping of GRAI Nets. Actually, GRAI Nets are extensions of GRAI Extended Actigrams and can therefore easily be transformed into UML Activity Diagrams as proposed in (Grangel *et al.* 2007, 2008). As previously mentioned, a GRAI Net is associated to a Decision Centre, which is the central construct of a GRAI Grid. Therefore, GRAI Decision Centres must be mapped onto a UML construct so that it can be expanded into UML Activity Diagrams. This problem of defining a consistent mapping respecting the hierarchical breakdown of GRAI Grids into GRAI Nets is represented in Figure 6.

Therefore it is necessary to navigate into the UML metamodel to find which UML concepts can be broken down into Activities. An excerpt of the UML metamodel is shown in Figure 7. On this excerpt we can notice that an *Activity* is a specialisation of the abstract metaclass *Behavior*. Moreover an *Activity* is composed of *Activity Node*s that are transitively specialised into *CallBehaviorActions* that are associated to the *Behavior* they call. In other words, that means that an activity can contain *CallBehaviorActions* calling activities that can be considered as sub-activities. Therefore *Activity* is a candidate as target of a grid and thus its *CallBehaviourActions* are possible targets for transformation of Decision Centres. In this case GRAI grids will be represented on Activity Diagrams. On the Figure 7, we can also see that the *UseCase* metaclass is a specialisation of the abstract metaclass *BehavioredClassifier*. This metaclass has a composition relation with the *Behavior* metaclass. By using the specialisation relationship between *Activity* and *Behavior*, that means that a *Use case* can be broken down into an *Activity*. Therefore *Use Cases* can be used as target for transformation of Decision Centres. Moreover, Use cases are packageable elements and use case diagrams can contains packages. Therefore GRAI grids can be mapped onto packages and represented on Use Case Diagrams. Figure 8 shows the summary of these two proposals

Figure 6. Problem of defining consistent mappings

Figure 7. Excerpt of the UML metamodel

Figure 8. Proposals for Decision Centre mapping

From the second viewpoint, we must come back to the main purposes of defining GRAI Grids. This formalism is intended as a way to model a comprehensive cartography of the decision-making within the enterprise. With a Software

Engineering approach, a GRAI Grid (and more precisely a Decision Centre) can be considered from two points of view:

- from a **requirement-oriented** point of view: each Decision Centre is considered to • a requirement on how a decision has to be made. In this case, the most appropriated UML construct for the mapping of a Decision Centre is a **Use Case**.
- from a **process-oriented** point of view: in this case, the focus is more on the flows (Decision Frames and Information Flows) and the schedule of decision-making. Therefore, **UML Activity Diagrams** are good candidates for the translation of GRAI Grids.

These two last proposals are consistent with the mapping of GRAI Nets onto UML Activity Diagrams as well as with the bottom-up approach presented earlier. The following two sections will detail each mapping proposal. In order to fill the semantic gap, these mappings will be completed by the Profile definitions discussed in Section 5.4. Section 5.5 will recall the main components of a UML Profile and Sections 5.6 and 5.7 will describe the proposed profiles.

5.2 First Mapping onto UML Use Case Diagram

In this case, GRAI Grids are considered to be a set of requirements on how a decision has to be made. Each requirement is represented by a Decision Centre which can be mapped onto a UML *Use Case* since they are both used to capture the requirements of a system, that is, what a system is supposed to do (OMG 2009). Since Use cases are packageable elements, the Function a Decision Centre belongs to can be mapped onto a UML *Package* used as a Use Case container. If a Function is broken down, each subfunction is mapped onto a nested *Package*. Consequently, a GRAI Grid is also mapped onto a UML *Package* and a GRAI Decisional Model is mapped onto a UML *Model* in order to maintain the consistency of the structural hierarchy of the model. However the bi dimensional structure of the GRAI grids is not kept: a Decision Centre belongs to both a line (i.e. Decisional Levels with temporal aspects) and a

column (i.e. functional breakdown). On the UML side, a package is a container for use cases, because use cases are packageable elements, but a use case belongs to zero or to one and only one package. The consequence is the following: a mapping of GRAI grids onto UML using use cases has to give the priority to one of the both dimensions: temporal or functional dimension. In this paper, the functional breakdown has been chosen: packages will be used as target for the transformations of the functions and temporal information carried by the lines representing the Decisional Levels of the grids will be considered as secondary information. Nevertheless the approach presented in this paper could be used for the dual approach giving priority to the line and considering the functional breakdown as a secondary aspect.

External and internal Information Sources are considered as information providers: they can be mapped onto *Actors* belonging to a specific *Package*. Finally, Information Flow and Decision Frames are directed relations than can be mapped onto *Dependencies*. Since a UML Dependency is defined from a client to a supplier (OMG 2009), it will be directed from the target of the Information Source or Decision Frame to its source in order to keep the logical dependency of the initial information and decision flows. As a graphical consequence, the resulting UML arrow points in the opposite direction to the GRAI arrow. A summary of the proposed mapping is shown in Figure 9.

Figure 9. First Mapping from GRAI Grid onto UML Use Case Diagram

5.3 First Mapping onto UML Activity Diagram

The second approach consists in mapping GRAI Grids onto UML *Activity*. In order to ensure the consistency of refinement of the Decision Centre into GRAI Nets, Decision

Centres are mapped onto *CallBehaviorActions*. The called behaviour is then the *Activit*y corresponding to its GRAI Net. The Function a Decision Centre belongs to is mapped onto an *Activit*y *Partition* graphically represented by swimlanes. Indeed, as mentioned into the UML Superstructure (OMG 2009): "They *[Activity Partitions]* often correspond to organizational units in a business model". The mapping of Functions onto *Activity Partition*s is then natural. Since it is possible to share elements between *Activity Partitions*, UML makes it possible to build multidimensional activity structures. However we will only keep the vertical functional view as privileged view. The two justifications are the following:

- since we have given the priority to the functional breakdown for the mapping of grids onto use case diagram, it will be easier to compare the results produced by the both approaches.
- even if we choose a bidimensional structure taking both the lines and the columns of the grid into account by creating both horizontal and vertical *Activity Partitions*, temporal information about horizon and period will be lost in this first mapping (see section 5.4).

This mapping is consistent with the GRAI Function breakdown since a UML *Activity Partition* can contain other *Activity Partitions*. GRAI Grids are mapped onto UML *Activities*. The Decision Frames and Information Flows are mapped onto both *Object Flows* and connected *Input/Output Pins*. Actually, since Decision Frames and Information Flows carry information, UML *Object Flows* are natural candidates as targets for the mapping of these source elements. But since UML rules impose that *Object Flows* must be connected to actions through pins (in the UML metamodel, *Call Behavior Actions* are specialisations of the abstract *Action* metaclass), it is compulsory to create also input and output pins for connecting the object flows to its corresponding input and output actions. Internal and External Information Sources are

mapped onto *Central Buffer Node*s belonging to two specific *Activity Partitions*. A summary of the proposed mapping is shown in Figure 10

Figure 10. First Mapping from GRAI Grid onto UML Activity Diagram

5.4 Discussion

These two mappings are obviously incomplete and some GRAI concepts and properties of concept, such as Horizon and Period as well as the attributes of Decision Frames, are missing. Moreover different GRAI concepts are mapped onto the same UML basic construct and, for example, in the case of the second proposed mapping, this makes it impossible both to know if a UML flows comes from a GRAI Information Flow or a Decision Frame and to write a backward transformation based on this mapping. Therefore it is needed to keep information about the source model when the target model is generated and thus to preserve the semantics of the source model. Two main approaches for solving this problem can be investigated. The first one consists in enriching the set of constructs of the target modelling language and then keeping additional information in the target model. The second one consists in keeping the additional semantics 'outside' the target model, for example, by storing applied transformation rules in a log file or by using a third linked model capturing

the semantic gaps. In this paper, only the first approach is investigated through the definition of UML profiles presented in the next sections.

5.5 UML Profile Definition

A profile is a specific version of UML. Generally, a profile is first defined by means of a domain model which represents the new concepts and their relationships as well as a description of their semantics. Then the mapping of these new concepts onto UML constructs is defined through a set of extension elements applied to the UML basic constructs. Therefore, a UML Profile can be considered as being a lightweight extension mechanism that adapts the UML Metamodel to one Specific Modelling Domain. A typical UML Profile is made up of stereotypes, tagged values and constraints (OMG 2009):

- **Stereotypes**: according to the UML 2.2 specification, these are specialisations of the metaclass *Class* and define how an existing metaclass may be extended. Each stereotype may extend one or more metaclasses of the UML Metamodel through extensions as part of a profile.
- **Tagged Values**: these are properties of a stereotype and are standard meta-attributes.
- **Constraints**: these are conditions or restrictions expressed in natural language text or, better, in a machine-readable language such as OCL (OMG 2006) that are classically used in UML to define the usage of a model element, and in the case of a profile definition, of a stereotype or of its tagged values.

The profile definitions presented in the next two sections are devoted only to the transformation of GRAI Grids into Use Case Diagrams (called 'UML UCD' in the following) and into Activity Diagrams (called 'UML AD' in the following). They are part of a more general on-going work that aims to define a complete specialisation of UML for bridging all the GRAI formalisms (Extended Actigrams, Grids and Nets) with UML. Since the objective is both to transform GRAI Grids and to define UML profiles, the starting domain model for these profile definitions is the GRAI Grid metamodel presented in Section 4. As mentioned in the previous sections, one of the

main problems of the first mappings that are presented is the loss of information. In the following sections, an approach based on the definition of UML Profiles is presented to tackle this problem when UML is considered as the target model. These profiles are defined to enrich the first mappings summarised in Figure 9 and 10: each stereotype presented in the following sections will extend the basic UML elements in the right-hand columns of Figure 9 and 10. These profiles are called respectively '**UML Profile for GRAIGrid2UCD**' and '**UML Profile for GRAIGrid2AD**'.

5.6 Profile for Transformation into UML Use Case Diagram

For this profile (shown in Figure 11), five stereotypes were created to extend the *Package* metaclass:

- **graiDecisionalModel** for the root element,
- **graiGrid** for the GRAI Grid structure,
- **graiFunction** for the columns associated to GRAI Functions or Subfunctions,
- **graiExternalInformation** for the left-most column of a GRAI Grid,
- **graiInternalInformation** for the right-most column of a GRAI Grid.

Figure 11. UML Profile for Transformation of a GRAI Grid into a UML UC Diagram

The *Use Case* metaclass was extended by the stereotype **graiDecisionCentre**. Since the rows (i.e. the Level) of GRAI Grids have no corresponding concepts in UML, their properties *horizon* and *period* have to be copied onto all UML elements corresponding to row members. Therefore, two attributes (i.e. tagged values) *horizon* and *period* were added to the stereotype graiDecisionCentre. This method was also used for the stereotype **graiInformationSource** to extend the metaclass *Actor*.

Finally, the *Dependency* metaclass was extended by two stereotypes to take completely into account the two types of flows used in GRAI Grids:

- **graiInformationFlow**, which possesses one attribute named 'type'with which to identify its nature (i.e. Internal or External),
- **graiDecisionFrame**, containing four attributes to store the characteristics of a Decision Frame (i.e. objectives, decision variables, constraints and criteria).

5.7 Profile for Transformation into a UML Activity Diagram

As for the previous profile, nine stereotypes were defined (Figure 12) to fill the semantic gap between the GRAI Grid concepts and the basic constructs of UML. The main differences concern the extended UML metaclasses:

- **graiDecisionalModel** extends the metaclass *Package*,
- **graiGrid** is now an extension of *Activity*,
- **graiFunction**, **graiExternalInformation** and **graiInternalInformation** extend the *ActivityPartition* metaclass,
- **graiDecisionCentre** is an extension of the *CallBehaviorAction* metaclass,
- **graiInformationSource** is used to extend *CentralBufferNode*,
- **graiDecisionFrame** and **graiInformationFlow** are extensions of *ObjectFlow*.

Figure 12. UML Profile for Transformation of a GRAI Grid into a UML Activity Diagram

5.8 New Mappings

Taking these two profile definitions into account, two new mappings are proposed on Figure 13 and 14 for the 7 fundamental GRAI Grid concepts. These mappings are presented as tables where:

- the first column contains the GRAI Grid source concepts;
- the 'Conditions' column of the Figure 14 gives the conditions for generating different UML targets. For example, the Figure 14 shows that the transformation of Information Flow will depends on the fact that its source and target are Decision Centre or not;
- the 'UML Targets' column shows the UML elements corresponding to the GRAI source concepts. For some GRAI source concepts only one UML element will be generated. For example, a GRAI Decision Centre will be

transformed into a UML Use Case. But for other GRAI concepts, several UML elements will be generated. For example, in the case of transformation into a Use Case Diagram, a grid will generate 3 packages: one for the grid itself containing the two others associated to External Information and Internal Information;

- the 'Stereotypes' column shows the stereotypes that will be applied to the corresponding UML targets;
- the 'Tagged Values' column shows the names of the tagged values defined on the stereotyped UML targets with the possible values in case of enumerated types (for example, the tagged value 'type' defined on the 'graiInformationFlow' stereotype can be 'Internal' or 'External')

Figure 13. Mapping from GRAI Grid onto a UML Use Case Diagram using a profile

Figure 14. Mapping from GRAI Grid onto a UML Activity Diagram using a profile

6. Application

In order to demonstrate the feasibility of the implementation of the two complete mappings using UML profiles presented in the previous sections, a model transformation language was used. This section presents first an overview of the language used and then a description of the results thus obtained.

6.1 ATL Overview

Atlas Transformation Language (ATL) is a Model Transformation Language that has been developed and specified by the ATLAS INRIA & LINA research group (LINA 2009), as an answer to the OMG's MOF/QVT Request For Proposal (RFP) (Jouault *et al*. 2006). ATL is a hybrid of declarative and imperative languages based on OCL (OMG 2006). The preferred style of transformation writing is declarative, which means mappings can be expressed by rules. However, imperative constructs are provided so that some mappings, which are too complex to be handled declaratively, can be specified within rules by using imperative helpers.

A **rule** describes the transformation from a source model to a target model by relating metamodels. Each rule contains a unique name. It is introduced by the keyword 'rule' that is followed by the rule's name.

In the **source pattern**, rules declare which element type of the source model has to be transformed. It consists of the keyword 'from', a source variable declaration and an optional precondition. This precondition is expressed using an OCL expression that restricts the rule triggering to elements of the source model that satisfy this precondition. When no filter is specified, the rule is executed for each input element whose type corresponds to the in variable declaration.

A **first optional** section introduced by the keyword 'using' can be used to declare local variables.

In the **target pattern**, rules declare to which element(s) of the target model the source pattern has to be transformed to. It may contain one or several target pattern elements. A target pattern element starts with the keyword 'to' and consists of a variable declaration and a sequence of bindings (assignments).

A **second optional section** of an ATL matched rule is the 'do' section. This section enables to specify a sequence of ATL imperative statements that will be executed once the initialization of the target model elements generated by the rule has completed. This imperative block can be used to initialise some model element features that have not been initialised using the declarative bindings, or to modify some already initialized features. This section will be used in the following to apply stereotype to a target element. The general structure of a rule is shown in the following code.

```
rule <ruleName> {
  from <sourceVariable> : <sourceMetaModel>!<sourceElement>
            [(<precondition>)]

  [using <local variable declaration>]

  to <targetVariable> : <targetMetaModel>!<targetElement> (<assignments>)

  [do {<imperative statements>}]
}--end of the rule
```

Eight ATL rules were written in order to implement the complete mapping of the seven basic constructs of GRAI Grids onto UML Use Case Diagrams: two rules were needed for the transformations of GRAI functions, depending on their potential breakdown into subfunctions.

Ten rules were developed for the transformation into UML Activity Diagrams, including two for GRAI Functions and three for GRAI information flows according to the type of their source and target (Decision Centre or Information Source); indeed, to conform to the UML Metamodel, it is necessary to manage the input/output *Pins* of *CallBehaviorActions* for UML object flows.

The following sections present three kinds of rules: simple rule, rule with helper and rule with application profile.

6.2 Simple Rule

The first rule presented deals with the transformation of a GRAI Function into a UML Package without using the proposed Profile.

```
rule GraiFunction2UmlPackage {
  from source_GraiFunction : GraiGridMetaModel!Function

  to target_UmlPackage : UML2!Package (
     --the name is the same
     name <-source_GraiFunction.name,

    -- the subfunctions of the current GRAI function
    -- will be elements of the generated UML package
     packagedElement <-source_GraiFunction.subFunction )

     --end of 'to' section
} --end of the rule
```

This rule simply copies the name of the source GRAI Function to the target UML Package. It also specifies that the results of the transformation of the subfunctions of the current GRAI Function will be elements belonging to the generated UML package.

6.3 Rule with Helper

The rule presented in this section deals with the transformation of GRAI Information Flow. As presented in the previous section the transformation depends on the fact that the GRAI Information Flow is connected or not to a Decision Centre. This condition will be a precondition of the transformation rule. In order to increase the readability and reusability, a helper is defined on the source element 'Information Flow' (see code herafter). This helper, named 'isComingFromADecisionCentre', uses the association linking an Information Flow to its source through the association end 'source' of the association 'from' linking a Flow to a Node (see Figure 5). In this helper, 'self' references the current InformationFlow on which the helper is invoked. Therefore 'self.source' references the instance of Node which is the source of the current Information Flow. On this Node, the OCL operation 'oclIsKindOf' in invoked to test if the node is a kind of Decision Centre. The same type of helper has been defined to test if an Information Flow has a Decision Centre as target.

```
helper context GraiGridMetaModel!InformationFlow
def : isComingFromADecisionCentre() : Boolean =
      self.source.oclIsKindOf(GraiGridMetaModel!DecisionCentre) ;
```

The same type of helper has been defined to test if an Information Flow has a Decision Centre as target.

```
helper context GraiGridMetaModel!InformationFlow
def : isGoingToADecisionCentre() : Boolean =
      self.target.oclIsKindOf(GraiGridMetaModel!DecisionCentre) ;
```

These two helpers have been used to define the preconditions of the rules implementing the mapping of Information Flows. Hereafter the source code corresponding to the mapping for Information Flows going from a Decision Centre to another one is given.

```
rule GraiInformationFlowDCtoDC2UmlObjectFlow {
  from source_GraiInformationFlow : GraiGridMetaModel!InformationFlow
```

```
      (source_GraiInformationFlow.isComingFromADecisionCentre()
       and
       source_GraiInformationFlow.isGoingToADecisionCentre()
       ) --end of 'from' section
  to target_UmlInputPin : UML2!InputPin (
        name <-'IP_IF_'+source_GraiInformationFlow.id.toString()
       ),
       target_UmlOutputPin : UML2!OutputPin (
        name <-'OP_IF_'+source_GraiInformationFlow.id.toString()
       ),
       target_UmlObjectFlow : UML2!ObjectFlow (
        --copy the name
        name <-source_GraiInformationFlow.name,
        --assign to the main activity
        activity <-source_GraiInformationFlow.graiGrid
  )--end of 'to' section
do {
--code to link the generated Pins and ObjectFlow to the actions
--that are the result of the transformation of decision centres
--Not detailed in this paper
   }--end of 'do' section
}--end of the rule
```

In this rule, the 'to' section describes the binding between the source and the target properties. As described in the Figure 14, this rule creates three UML target elements: an Input Pin, an Output Pin and an Object Flow. First, the Input Pin and the Output Pin are created. Their names are built by prefixing the name of the Information Flow by 'IP_IF_' and 'OP_IF_'. Then the Object Flow is created. Its name is set to the name of the Information Flow and the activity this generated Object Flow belongs to is set to the activity resulting from the transformation of the GRAI Grid the Information Flow belongs to. The 'do' section, not described here, contains the code making the links between the generated elements.

6.4 Applying UML Profiles

ATL makes it possible to use UML Profiles. The method to use profile with ATL is made up of four steps.

1) The first one consists in defining the profile with an UML tool (IBM ® Rational Software Modeler ® (IBM 2009) or MagicDraw (MagicDraw 2009) for example).
2) In the second step, the profile is applied to the generated UML model. For example, in order to apply the profile called 'GRAIGrid2UCD' (GRAI Grid to Use Case Diagram) to a target UML model, the following statement must be added in the 'do' section of the rule creating the UML model:

```
target_UmlModel.applyProfile(UML2!Profile.allInstances()->
        select(e | e.name = 'GRAIGrid2UCD').first());
```

3) The third step consists in applying stereotypes to the elements of the UML target model for which we want to keep additional semantics coming from the source model. The 'applyStereotype' method is invoked on the target element with an instance of the metaclass *Stereotype* as parameter. To get it, the 'getApplicableStereotype' method is invoked with the name of the stereotype to apply. For example, to apply the stereotype 'graiDecisionCentre' to the target UML Use Case called 'target_UmlUseCase' obtained after transformation of a GRAI Decision Centre, the following code must be added in the 'do' section of the rule creating a Use Case from a GRAI Decision Centre:

```
target_UmlUseCase.applyStereotype(
        target_UmlUseCase.getApplicableStereotype(
            'GRAIGrid2UCD::graiDecisionCentre'));
```

4) Finally, for concerned target elements, tagged values of stereotyped UML model elements are set using the 'setValue' method. This method is invoked on a UML element and has three parameters: (1) the stereotype, (2) the name of the tagged value and (3) its value. For example, to assign a value to the tagged values 'horizon' and 'period' of a Use Case obtained after transformation of a GRAI Decision Centre according to its nature, the following code must be added in the 'do' section of the rule creating an Use Case from a GRAI Decision Centre, after the application of the stereotype 'graiDecisionCentre':

```
target_UmlUseCase.setValue(
         target_UmlUseCase.getAppliedStereotype(
                   'GRAIGrid2UCD::graiDecisionCentre'),
        'horizon',
         source_GraiDecisionCentre.level.horizon);
target_UmlUseCase.setValue(
         target_UmlUseCase.getAppliedStereotype(
                   'GRAIGrid2UCD::graiDecisionCentre'),
```

```
'period',
 source_GraiDecisionCentre.level.period);
```

6.5 Discussion

Figure 15 and Figure 16 show the results obtained after the transformation of the GRAI Grid example of Figure 2 into respectively Use Case Diagram and Activity Diagram. Only the two left-most columns (i.e. *External Information and To Manage Products*, including its two subfunctions *To Manage Procurement* and *To Manage Materials*) and the three top-most rows are illustrated. These models were obtained by using first the ATL Development Tools (IDE developed for ATL on top of Eclipse) (Allilaire and Idrissi 2004) and then by importing the result into IBM Rational Software Modeler (IBM 2009) for visualising purposes.

Figure 15. Excerpt from the Result of Transformation into UML Use Case Diagram

Figure 16. Excerpt from the Result of Transformation into UML Activity Diagram

The models thus obtained contains the same information and semantics as the initial Grid, but these UML models clearly show that the GRAI Grid formalism provides models that are more readable and concise and therefore are better adapted for a general overview of decisional flows and functions at the Business level. Actually, the transformations of GRAI Grids into UML are not performed for simple translation purposes, but mainly to bridge models from the Enterprise Modelling domain (i.e. Business Level) with IT models. The two obtained UML models represent two complementary viewpoints on the decision making. The use case viewpoint can be used as a general view of both the services and their dependencies to be implemented into an information system. In contrast, the activity diagram is more process oriented: it highlights the sequences of decision activities. Moreover it is possible to refine the proposed profile for activity diagram by defining a stereotype

extending the *Partition* UML metaclass in order to represent explicitly the Decision Level concept (i.e. lines) of GRAI grids. This stereotype contains two properties (tagged values) for representing horizon and period. It makes it possible to use the multidimensional representation of the resulting activity diagram by assigning each *CallBehaviorAction* to both a vertical swim lane corresponding to the partition associated to a GRAI function and a horizontal swim lane corresponding to the partition associated to a GRAI Decision Level. This way, the obtained activity diagram is closer to the structure of the original GRAI Grid.

This experiment has shown how it is possible i) to define UML profiles for filling the semantic gap between GRAI Grids and UML Use Case Diagrams or Activity Diagram, and ii) to implement the profile-based mapping using a transformation language.

In contrast to the work presented by (Darras 2004), this paper has completely defined both the profiles and the mappings. Moreover, a complete implementation has been also carried out to validate the mapping proposals. This implementation has emphasised some inconsistencies of the mapping proposed in (Darras 2004), for instance concerning the direction of dependencies between use cases. Other mapping solutions have been also proposed in this paper dealing with Information Sources or Functions.

7. Conclusion

This paper has focused on the transformation of GRAI Grids into UML. The first main contribution of this work is related to the definition of the metamodel of the GRAI Grid. This metamodel has been used to define a first mapping onto UML Uses Cases and Activity Diagrams. In order to fill the semantic gaps between GRAI Grids and UML, two profiles have been proposed and implemented by means of a

transformation language. This implementation has demonstrated the feasibility of the whole approach. This approach, which is not limited to GRAI Grids, makes it possible to establish a bridge between dedicated Enterprise Modelling Languages and a language that is widely used in the Software Engineering domain. It can be used within a vertical MDA approach to link the CIM and PIM levels. But this approach can also be used in a horizontal manner in order to transform enterprise models that are expressed using different formalisms into models using a single language in order to make it easier to define the interoperability needs between the enterprise systems.

The work currently in progress deals with a complete integration of the profiles defined for the semantic lossless transformations of the three GRAI formalisms (i.e. Grids, Nets and Extended Actigrams) into a single UML profile. This profile will take both the general and the local aspects of the decision-making, as well as business process modelling, into account. Future research will focus on the transformation of the other model types at the Enterprise Modelling Level (organisation, business date, services, and so forth) according to both vertical (i.e. MDA-based) and horizontal (i.e. for solving interoperability problems) approaches. At last model weaving techniques are currently studied as alternative solutions to preserve semantics during the transformation process.

Acknowledgements. This work was initiated and partially funded by the EC within the 6th FP, INTEROP Network of Excellence (INTEROP 2007). The authors were indebted to TG2. It was also partially funded by the ASICOM project. This project started in April 2008 was approved by two French poles of competitiveness: PICOM in Trade Industries domain and Nov@log in Logistics domain.

References

Allilaire, F., *et al.*, 2006. ATL -Eclipse Support for Model Transformation. *In: Proceedings of the Eclipse Technology eXchange Workshop (eTX) at the ECOOP 2006 Conference, Nantes, France.*

Allilaire, F. and Idrissi, T., 2004. ADT -Eclipse Devopment Tool for ATL. *In: Proceedings of EWMDA-2, Kent*.

Berio, G., Project UEML, WP3, Deliverable D3.3, Requirements Analysis: initial core constructs and architecture, Annex2.2, GRAI metamodelling v2.5. , 2003, Technical report.

Berio, G. and Vernadat, F.B., 1999. New developments in enterprise modelling using CIMOSA. *Comput. Ind.*, 40 (2-3), 99–114.

Bézivin, J., 2004. In Search of a Basic Principle for Model Driven Engineering. *CEPIS, UPGRADE, The European Journal for the Informatics Professional*, V (2), 21–24 Available from: http://www.upgrade-cepis.org/issues/2004/2/up5-2Bezivin.pdf.

Bourey, J.P., *et al.*, 2006. INTEROP NoE: Deliverable DTG2.2: Report on Model Interoperability. [online] Available from: http://interopvlab.eu/ei public deliverables/interop-noe-deliverables/tg2-model-driveninteroperability [Accessed 07 September 2009].

Chen, D. and Doumeingts, G., 2003. European initiatives to develop interoperability of enterprise applications-basic concepts, framework and roadmap. *Annual Reviews in Control*, 27 (2), 153–162.

Cuenca, L., Ortiz, A., and Vernadat, F., 2006. From UML or DFD models to CIMOSA partial models and enterprise components. *International Journal of Computer Integrated Manufacturing*, 19 (3), 248–263.

Darras, F., 2004. Proposal of a formal framework for the design and use of an Enterprise Resource Planning software. Thesis (PhD). Institut Polytechnique de Toulouse.

Doumeingts, G., *et al.*, 1993. GIM (GRAI Integrated Methodology) and its Evolutions -A Methodology to Design and Specify Advanced Manufacturing Systems. *In*: H. Yoshikawa and J. Goossenaerts, eds. *DIISM '93: Proceedings of the JSPE/IFIP TC5/WG5.3 Workshop on the Design of Information Infrastructure Systems for Manufacturing*, Vol. B-14 of *IFIP Transactions* North-Holland, 101–120.

Doumeingts, G., Vallespir, B., and Chen, D., 1998. Decisional modelling using the GRAI Grid. *In: International Handbook on Information Systems.*, 313–337 Springer-Verlag.

Eriksson, H. and Penker, M., 2000. *Business Modeling with UML: Business Patterns at Work*. J. Wiley.

Grangel, R., *et al.*, 2007. Transforming GRAI Extended Actigrams into UML Activity Diagrams: a First Step to Model Driven Interoperability. *In*: R.J. Gonçalves, J. Muller, K. Mertins and M. Zelm, eds. *3rd International Conference on Interoperability for Enterprise Software and Applications, Enterprise Interoperability II*, *New Challenges and Approaches*, March. ISBN 978-1-84628-857-9 Springer, 447–458.

Grangel, R., Bourey, J.P., and Berre, A., 2006. Solving Problems in the Parametrisation of ERPs using a Model-Driven Approach. *In*: G. Doumeingts, J. Muller, G. Morel and B. Vallespir, eds. *Enterprise Interoperability. New Challenges and Approaches, Interoperability for Enterprise Software and*

Applications Conference (I-ESA'06), March. ISBN 978-1-84628-713-8 Springer, 91–101.

Grangel, R., *et al.*, 2005. Enterprise Modelling, an overview focused on software generation. *In*: H. Panetto, ed. *Interoperability of Enterprise Software and Applications Workshops of the INTEROP-ESA International Conference EI2N, WSI, ISIDI and IEHENA 2005* Hermes Science Publishing, 65–76.

Grangel, R., Cutting-Decelle, A.F., and Bourey, J.P., 2008. A UML profile for transforming GRAI Extended Actigrams into UML. *In: 17th IFAC World Congress*, July., Seoul (Korea).

IBM, 2009. IBM Rational Software Modeler Development Platform 7.5. [online] Available from: http://www-01.ibm.com/software/awdtools/modeler/swmodeler/ [Accessed 07 September 2009].

IDEF, 2007. Integrated DEFinition Methods. [online] Available from: http://www.idef.com/ [Accessed 07 September 2009].

INTEROP, 2007. Interoperability Research for Networked Enterprises Applicationsand Software NoE (IST-2003-508011). [online] Available from: http://interop-vlab.eu/ [Accessed 07 September 2009].

Jardim-Gonçalves, R., Grilo, A., and Steiger-Garcao, A., 2006. Challenging the interoperability between computers in industry with MDA and SOA. *Comput. Ind.*, 57 (8), 679–689.

Jouault, F., *et al.*, 2006. ATL: a QVT-like transformation language.. *In*: P.L. Tarr and W.R. Cook, eds. *OOPSLA Companion* ACM, 719–720.

LINA, 2009. research laboratory. [online] Available from: http://www.lina.univnantes.fr/ [Accessed 07 September 2009].

MagicDraw, 2009. Magic Draw. [online] Available from: http://www.magicdraw.com/ [Accessed 07 September 2009].

Marshall, C., 2000. *Enterprise Modeling with UML. Designing Successful Software Through Business Analysis*. Addison Wesley.

Milewski, M. and Roberts, G., 2005. The model weaving description language (MWDL)-Towards a formal aspect oriented language for MDA model transformations. *In*: *Proceedings of the 1st Workshop on Models and Aspects - Handling Crosscutting Concerns in MDSD, in conjunction with the 19th European Conference on Object-Oriented Programming*.

OMG, 2003. MDA Guide Version 1.0.1. [online] Available from: http://www.omg.org/docs/omg/03-06-01.pdf [Accessed 07 September 2009].

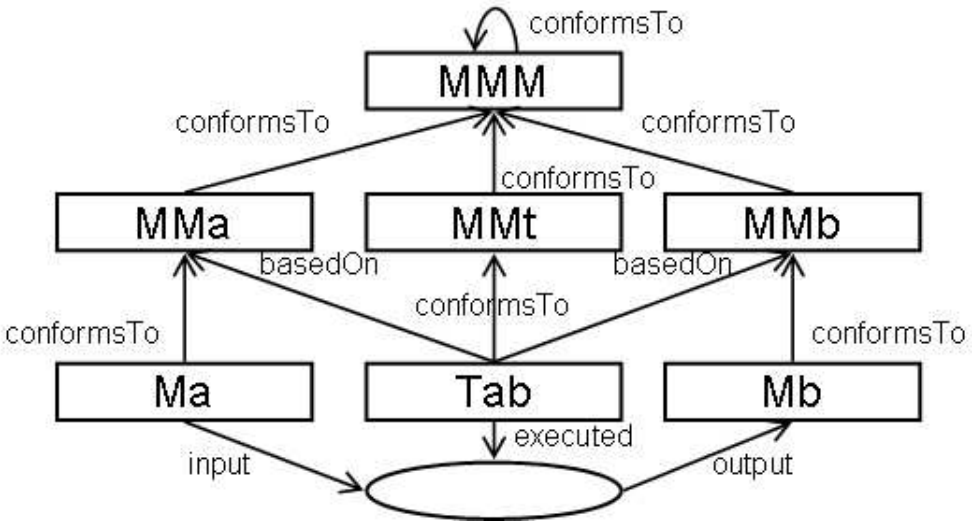
OMG, 2006. Object Constraint Language 2.0. [online] Available from: http://www.omg.org/spec/OCL/2.0/ [Accessed 07 September 2009].

OMG, 2009. Unified Modeling Language: Superstructure, version 2.2. [online] Available from: http://www.omg.org/spec/UML/2.2/ [Accessed 07 September 2009].

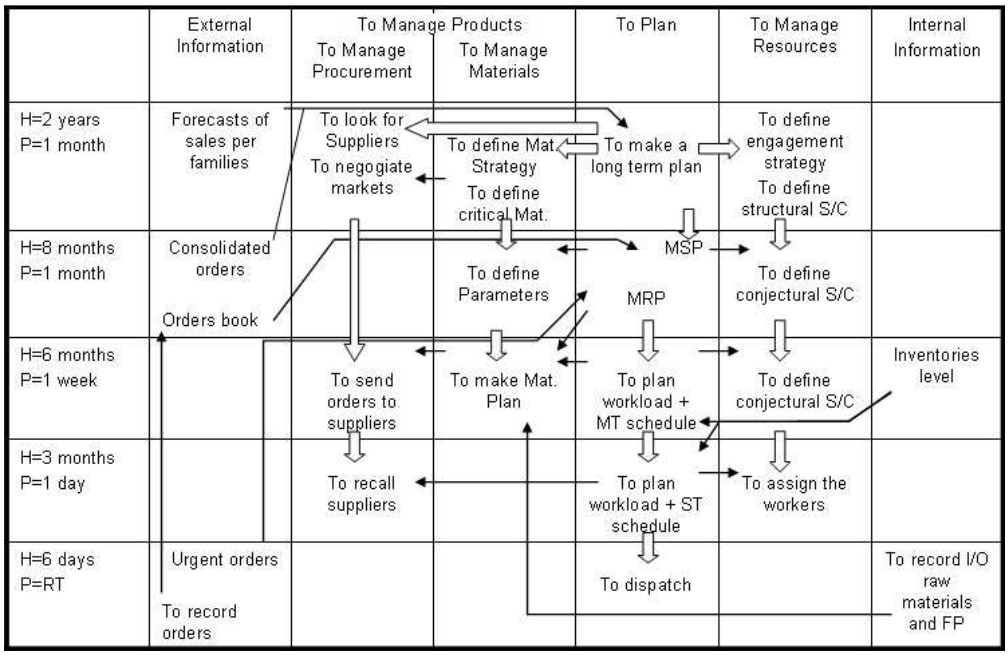
Tae-Young, K., *et al.*, 2006. A modeling framework for agile and interoperable virtual enterprises. *Comput. Ind.*, 57 (3), 204–217.

Vernadat, F.B., 1996. *Enterprise Modeling and Integration: Principles and Applications*. Chapman and Hall.

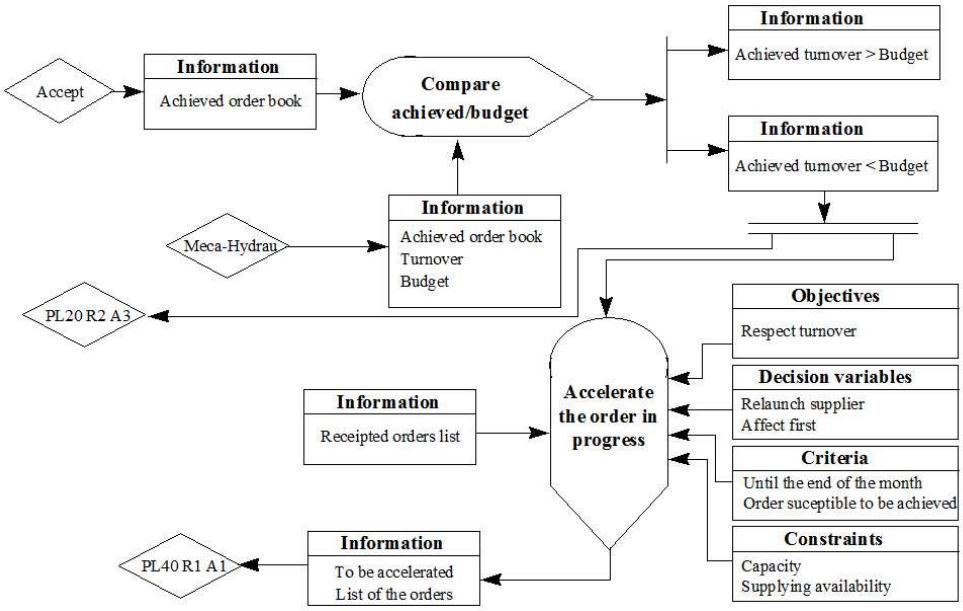
Williams, T.J., 1993. The Purdue Enterprise Reference Architecture. *In: Proceedings of the Workshop on Design of Information Infrastructure Systems for Manufacturing*, November. Elsevier.



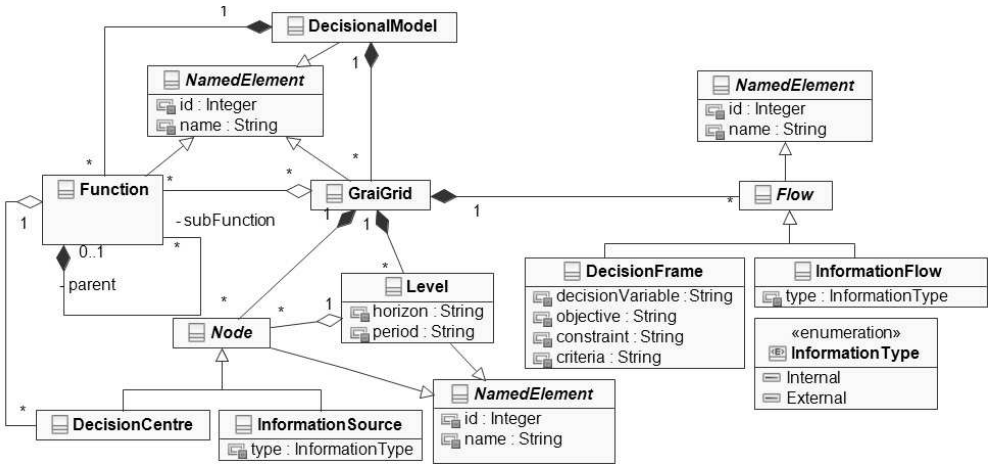
Model Transformation Pattern from (Allilaire et al. 2006)
188x99mm (72 x 72 DPI)



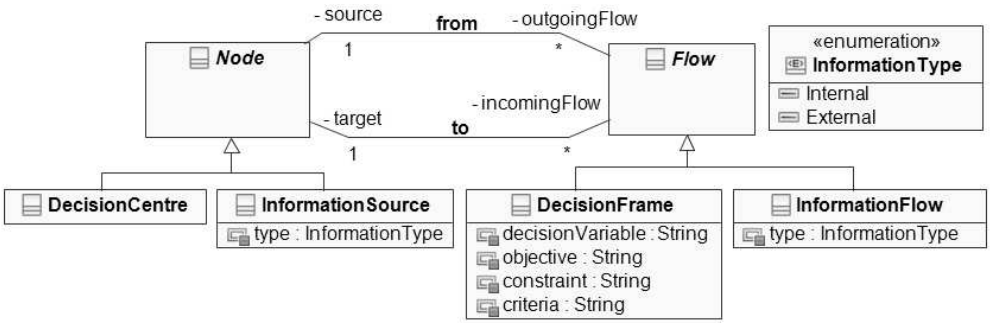
Example of a GRAI Grid
263x170mm (72 x 72 DPI)



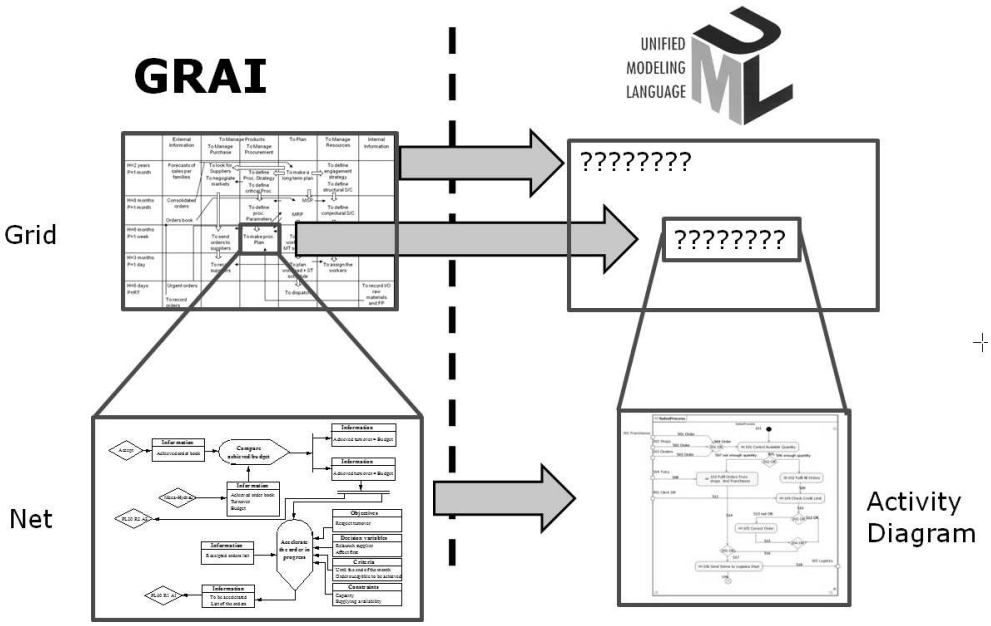
Example of GRAI Net
261x161mm (96 x 96 DPI)



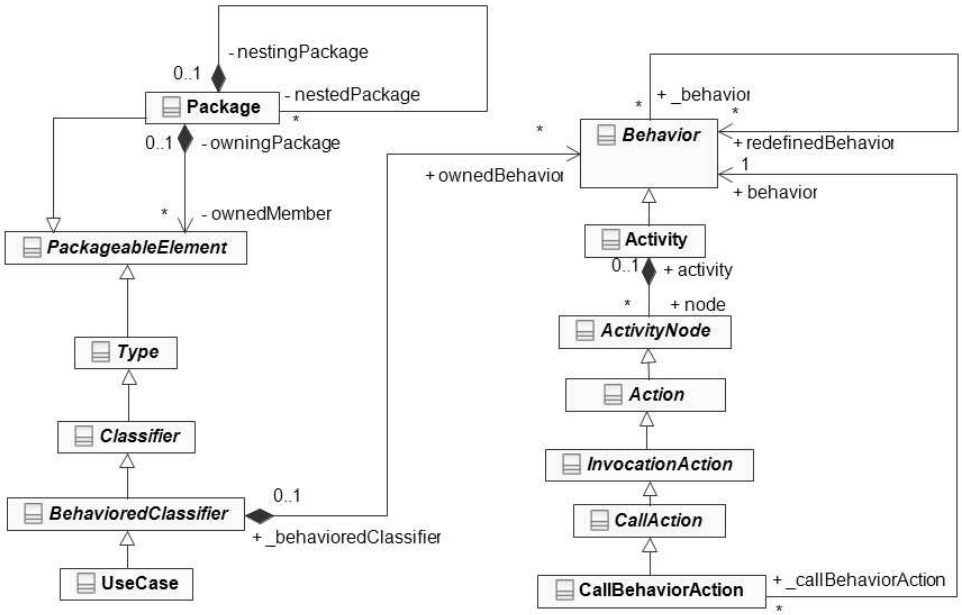
GRAI Grid Metamodel: classes, generalisations and structural associations
259x118mm (96 x 96 DPI)



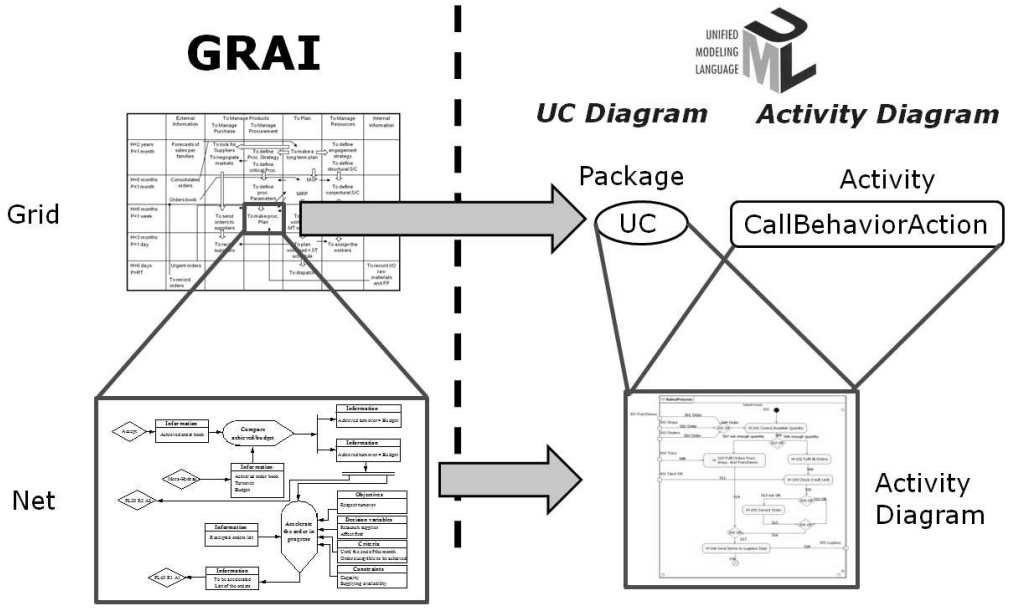
GRAI Grid Metamodel: Nodes and Flows
223x71mm (96 x 96 DPI)



Problem of defining consistent mappings
318x198mm (96 x 96 DPI)



Excerpt of the UML metamodel
235x150mm (96 x 96 DPI)



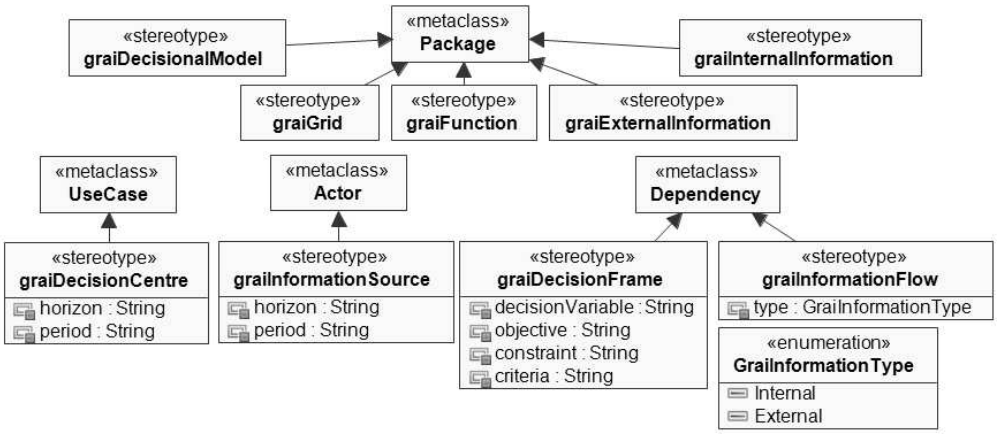
Proposals for Decision Centre mapping
316x192mm (96 x 96 DPI)

GRAI Grid	UML UC Diagram
DecisionalModel	Model
Grid	Package
Function	Package
DecisionCentre	UseCase
InformationSource	Actor
DecisionFrame	Dependency
InformationFlow	Dependency

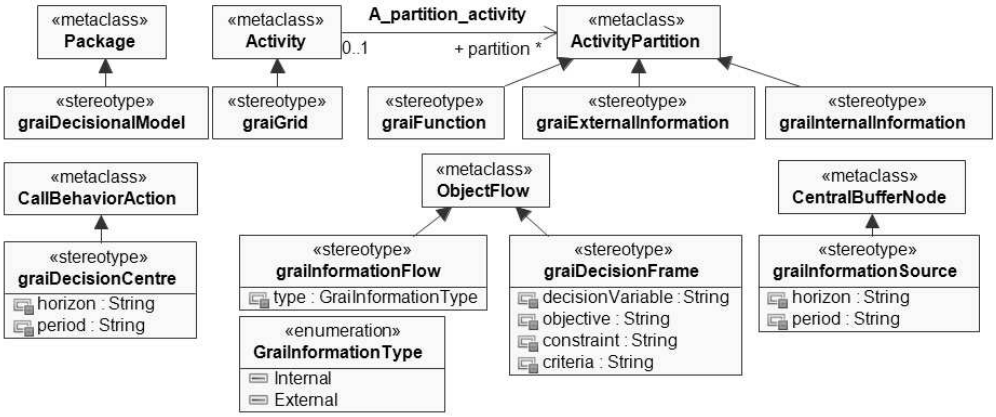
First Mapping from GRAI Grid onto UML Use Case Diagram
143x75mm (96 x 96 DPI)

GRAI Grid	UML Activity Diagram
DecisionalModel	Model
Grid	Activity
Function	ActivityPartition
DecisionCentre	CallBehaviourAction
InformationSource	CentralBufferNode
DecisionFrame	ObjectFlow + Pins
InformationFlow	ObjectFlow + Pins

First Mapping from GRAI Grid onto UML Activity Diagram
160x76mm (96 x 96 DPI)



UML Profile for Transformation of GRAI Grid into UML UC Diagram
229x101mm (96 x 96 DPI)



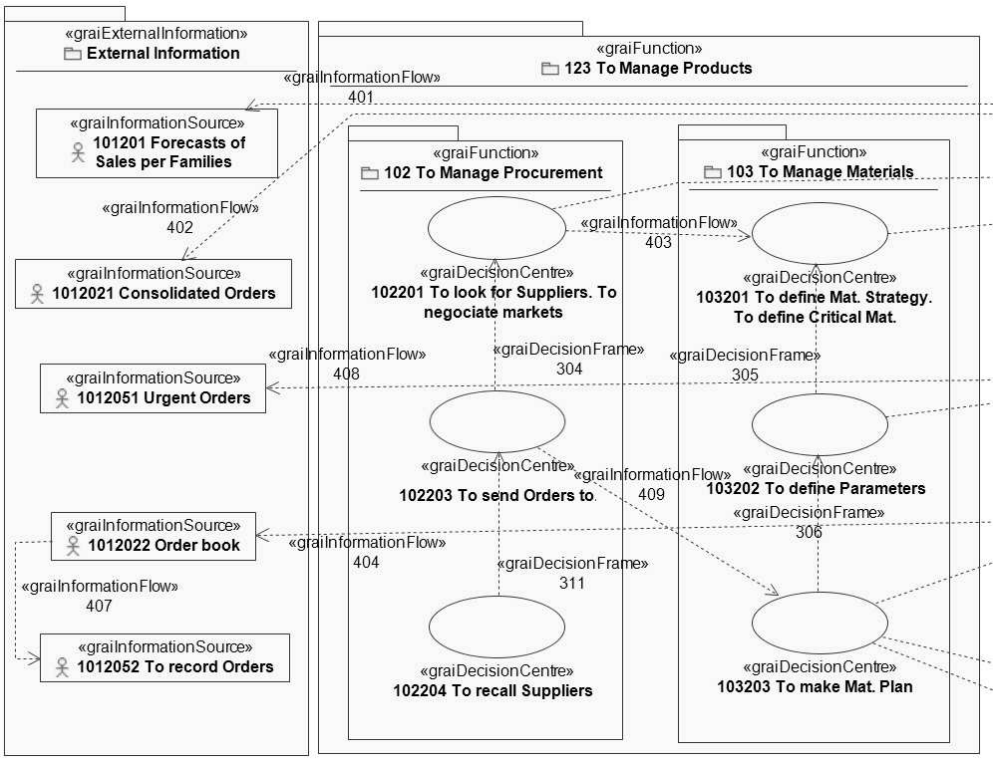
UML Profile for Transformation of GRAI Grid into UML Activity Diagram
244x104mm (96 x 96 DPI)

GRAI Grid Source Concepts	UML Targets	Stereotypes	Tagged Values
DecisionalModel	Model		
	Package	graiDecisionalModel	
GraiGrid	Package	graiGrid	
	Package	graiExternalInformation	
	Package	graiInternalInformation	
Function	Package	graiFunction	
DecisionCentre	UseCase	graiDecisionCentre	horizon period
DecisionFrame	Dependency	graiDecisionFrame	constraint criteria decisionVariable objective
InformationSource	Actor	graiInformationSource	horizon period
InformationFlow	Dependency	graiInformationFlow	type=Internal/external

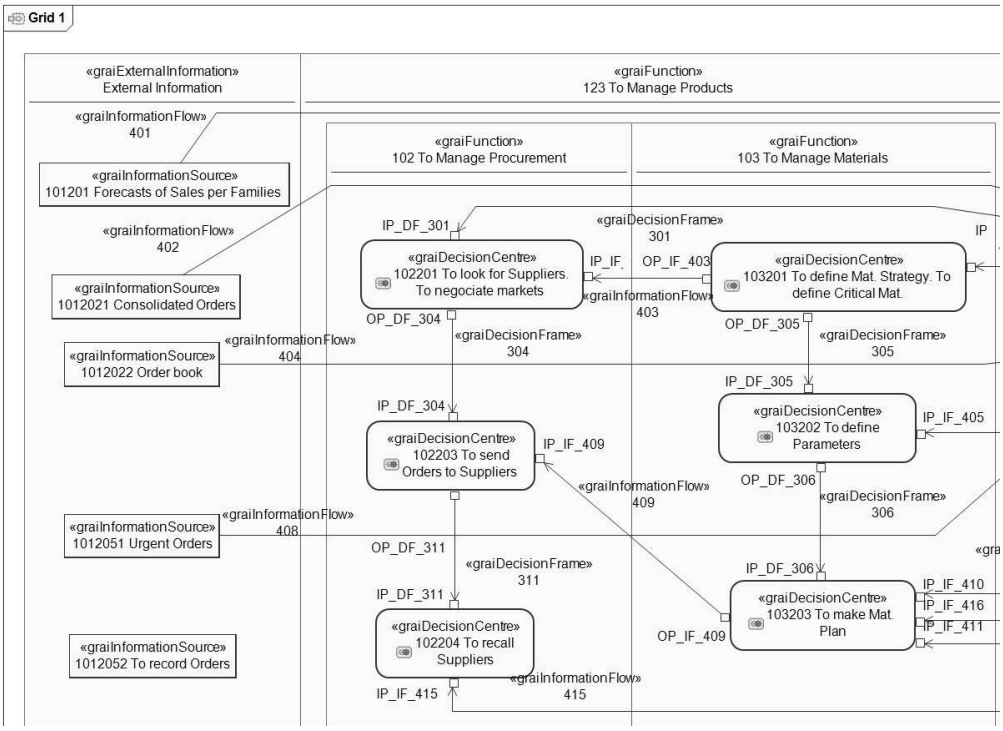
Mapping from GRAI Grid onto a UML Use Case Diagram using profile
241x133mm (96 x 96 DPI)

GRAI Grid Source Concepts	Conditions	UML Targets	Stereotypes	Tagged Values
DecisionalModel		Model		
		Package	graiDecisionalModel	
GraiGrid		Activity	graiGrid	
		ActivityPartition	graiExternalInformation	
		ActivityPartition	graiInternalInformation	
Function		ActivityPartition	graiFunction	
DecisionCentre		CallBehaviourAction	graiDecisionCentre	horizon period
DecisionFrame		ObjectFlow	graiDecisionFrame	constraint criteria decisionVariable objective
		InputPin		
		OutputPin		
InformationSource		CentralBufferNode	graiInformationSource	horizon period
InformationFlow	from Decision Center to Decision Center	ObjectFlow	graiInformationFlow	type=Internal/external
		InputPin		
		OutputPin		
	from Decision Center to Information Source	ObjectFlow	graiInformationFlow	type=Internal/external
		OutputPin		
	from Information Source to Decision Center	ObjectFlow	graiInformationFlow	type=Internal/external
		InputPin		

Mapping from GRAI Grid onto UML Activity Diagram using profile
291x199mm (96 x 96 DPI)



Excerpt from the Result of Transformation into UML Use Case Diagram
265x206mm (96 x 96 DPI)



Excerpt from the Result of Transformation into UML Activity Diagram
317x230mm (96 x 96 DPI)