Deriving Multi-Agent System Behavior

Ahmed Harbouche ¹, Mohammed Erradi² and Aicha Mokhtari³

¹ LME Lab, Hassiba BenBouali University, Chlef, Algeria
² SIME Research Lab, ENSIAS, Mohammed V Souissi University, Rabat Morocco
³ LRIA Lab, Houari Boumedienne University, Algiers, Algeria
A.harbouche@univ-chlef.dz, erradi@ensias.ma, Aissani_Mokhtari@yahoo.fr

Abstract

The multi-agent systems (MAS) have become a very powerful paradigm in the development of complex computer systems. The modeling of these systems can reduce this complexity during the development phases. The Model-Driven Architecture (MDA) approach can be used to resolve this problem. It allows designers to model their systems at different abstraction levels, providing them with automatic model transformations to incrementally refine abstract models into concrete ones. This paper presents a MDA approach to develop multi-agent system with the derivation of the behavior of a given system agent from its global requirements. The suggested approach is based on the definition of an appropriate requirements meta-model (Computational Independent Meta-Model CIMM) and the definition of a target design meta-model (Platform Independent Meta-model PIMM). The CIM models are specified using UML activity diagram extended with collaborations to describe the system global behavior. The agent’s behavior model (PIM) is in the form of distributed UML state machines. Automatic model transformations between these two models have been designed in order to govern the derivation process. A real application of telediagnosis in neuroscience has been developed using this approach.

Keywords: Model driven engineering, Behavior, Derivation, Model to model transformation, Computational independent model

1. Introduction

The multi-agent systems (MAS) have become a very powerful paradigm in the development of complex and distributed computer systems. This paradigm is used in several areas such as simulation, virtual markets, digital libraries and information systems in general. The multi-agent systems development is an increasingly complex task. Usually the global behavior of such distributed systems is not achieved by a single component but by a set of collaborative agents. Such global behavior can be decomposed into partial behaviors performed by different system agents. A manual decomposition of the behavior can lead to design errors. Therefore an automatic transformation approach is needed to derive the behavior of these agents from the system global behavior. The use of a software engineering approach to support such derivation process may lead to a highly platform independent designs and promotes the reuse of software artifacts where a derived behavior model can be reused on different platforms.

The Model-Driven Architecture (MDA) [1] approach aims to change the focus of software development from code to models. The MDA specifies three levels of models; Computation Independent Models (CIM), Platform Independent Models (PIM) and Platform Specific Models (PSM). The system is therefore represented by different abstract views. Model
transformations are needed between the different levels of abstraction in order to incrementally refine models into final application code. Models are defined in terms of formal meta-models. These meta-models include the concepts needed to describe a system at a certain level of abstraction, and the relationships existing between them. In order to describe the model transformations needed to refine abstract models into more concrete ones, a mapping between their corresponding meta-models must be defined. Thus, applying a MDA approach requires the appropriate meta-models and the corresponding model transformations. Many researches on MDA are focusing on transforming the PIM to PSM models [2-7]. However, less attention has been given to the transformations of the CIM to PIM models especially in the case of distributed applications where specific properties, such as messages exchange and collaborations between the distributed components, need to be addressed.

This paper presents a MDA approach to derive the behavior of the system agents by transforming the CIM model to PIM models. The CIM model describes the global behavior of a multi-agent system in an abstract way. The PIM models represent the local behavior of the agents where each agent is identified by the local behavior of its roles. The appropriate meta-models have been defined at each level of abstraction with the corresponding model transformations. The proposed approach allows designers to model their system using UML activity diagram extended with collaborations concepts included in the CIM meta-model. This initial model (CIM) is then automatically transformed to the behavior of the roles (PIM). A set of rules needs to be defined to govern the transformation during the derivation process. As we are in a distributed context, the derived behaviors of the roles need also to be synchronized. The collaboration of these behaviors should satisfy the initial CIM model.

This paper presents an extension of our previous work which proposes an approach based on the MDA (Model Driven Architecture) to ensure the transition from the Platform Independent Models (PIM) to PSM models. This paper is organized as follows. Section 2 reviews the related works. Section 3 presents the proposed approach, the basic meta-models and the model to model transformation. Section 4 presents the transformation rules. Section 5 presents a case study. Section 6 presents the implementation of the MAS derivation process. Section 6 concludes this paper.

2. Related Work

The MDA specifies three levels of abstractions expressed in terms of models. Many research works on MDA are focusing on the lower levels and transformations between each other. This is the case of the methodologies TROPOS [2], INGENIAS [3] and Silva et al. work [7]. These proposals address especially the PIM to PSM and PSM to code transformations. However, the transformations of the CIM to PIM models have not been addressed. In contrast to these works, our proposal is focused especially on the CIM to PIM transformation with the behavior derivation.

The transformation of the requirements models (CIM) to the design models (PIM) is the first step to quality design of a complex system. The works [8-10] provide a model transformation of the CIM (Computational Independent Model) to PIM (Platform Independent Model). While STREAM (A Strategy for Transition between Requirements Models and Architectural Models) presented in [11] generates an architectural model described in Acme architectural description language from i* requirements model. Although these proposals are similar to ours in the sense of using transformations between requirements and design models, they do not explore the case of distributed applications such as multi-agent systems where specific properties, such as message exchange and collaborations between the distributed components, need to be addressed. We also provide the behavior derivation during the model transformations.
The behavior derivation needs a transformation approach to automatically derive the behavior of the components from the system global requirements. Bochmann [12] suggests an algorithm for the derivation of behavior based on behavior expressions as basic constructs for the system global requirements. The expressions of collaborations describe the sequential, choice, repetition and parallel structures. While they could not include the conditions (guards) embedded in the choice structures and the repetition structure. In addition, the derived behaviors are in the form of expressions. These expressions cannot be verified and validated automatically. Differently from such works, our proposal was designed to allow the behavior derivation based on model transformations. The models provide an abstract representation of collaborations with a clear understanding of the underlying system. UML activity diagrams extended with collaborations are used to describe the multi-agent system requirements (CIM). They allow representing concepts that could not be described using expressions, as done in [12]. This is the case of the conditions and control nodes. The derived behaviors of the system roles (PIM) are UML state machines which could be used for an automatic generation of code. The derived state machines models can be verified and validated using existent tools such as the SPIN model checker [13]. In addition, our proposal generates platform independent models which could be implemented on more than one platform.

The requirements specification (CIM models) describes the system global behavior. To express the system global behavior, UML interaction and communications diagrams [14] are generally used to describe collaborative behavior. However, they are expressed in terms of message exchanges between the various components cooperating at the preliminary stages of development where it is not necessary to have too many details. To represent the multi-agent system global behavior, we suggest the use of UML collaborations as the main blocks of activities for the construction of the CIM models. The collaboration describes a structure of collaborating elements (roles), each performing a specialized function which collectively accomplishes some desired functionality. The collaborations are very appropriate to model the requirements because they provide a structural framework for these requirements that can embody both the role behaviors and the interactions between the roles needed to fulfill the service without going into details of the collaboration.

3. Deriving MAS behavior by CIM to PIM Transformation

Within the MDA context, many software architects understand and argue that CIM level and its transformation to PIM level is the first step to quality design of a complex system [8-10]. In this direction, we define a model-driven approach to derive the behavior of the multi-agent system roles. This approach will allow developers to build more flexible and reusable designs. The main goal of this approach is to define the derivation process (Figure 1). This goal was achieved by:

1. The definition of the requirements meta-model (CIM meta-model) which enable the description of the system at a very high level of abstraction. Models using this meta-model should express the global behavior of a given system.

2. The description of the design meta-model (PIM meta-model) to describe the local behavior of each system role. The PIM models will be automatically generated from the requirements ones.
3. The definition of the model to model transformation which maps the concepts in the CIM meta-model to those in the PIM meta-model. This transformation governs the derivation process.

![Diagram of the Derivation Process]

**Figure 1. The Derivation Process**

### 3.1. The Requirements Meta-Model (CIM Meta-Model)

The definition of the requirements meta-model is aimed to describe the system global behavior. It helps the experts to describe the system in a high level of abstraction. At this stage, no design concepts or information about the final target platform are taken into account. Models at this level must provide a clear picture of the system behavior and the collaboration between its components.

The proposed CIM meta-model (Figure 2) in this paper has been designed to provide the appropriate concepts used to describe a multi-agent system behavior with the relationships between these concepts. It is considered as the source meta-model of the derivation process. It defines the activity diagram meta-model with its main classes and associations while considering collaborations as the basic activities. The collaboration in the CIM meta-model may consist in one or two collaborations or sub-collaborations. A sub-collaboration consists in some actions accomplished by the collaborating roles. The roles describe the different system agent behavior. UML activity diagrams are suitable to represent choreography of collaborations and sub-collaborations. These collaborations are the basic activities in a composite collaboration describing the system global behavior. UML activity diagrams can express sequential behavior, alternative behavior, competing behavior (parallel composition), repetitive behavior, as well as interruptions. At the models level, the MAS global behavior is defined as a composition of activity diagrams and collaborations.
3.2. The Platform Independent Meta-Model (PIM Meta-Model)

To describe the PIM level in the proposed approach, a PIM meta-model (PIMM) was suggested. It is used as the intermediate platform-independent language. This intermediate abstraction level is aimed at bridging the semantic gap between the requirements and platform meta-models, reducing the complexity of the required model transformation. Thus models created at this PIM level provide a more detailed description of the multi-agent system under development. The proposed PIM meta-model has been defined taking into account the concepts: agent, role, organization, activity, action and communication. Agents, roles and organization are used for specifying the system structure; while activity, actions and communications are used for defining the behavior of each role or agent.

The structural part of the proposed PIMM (Figure 3) is an extension of FIPA Modeling TC’s Agent Class Superstructure meta-model (FIPA ACSM) [15]. This meta-model is also based on the superstructure of UML [14] to model the multi-agent system behavior. The main entities of this meta-model are the agents, the roles and the groups.

An agent is an autonomous entity that can interact with other agents in the environment. Conceptually an agent has an internal structure and communication perspectives. An agent is characterized by its roles and its behavior. The behavior defines all the actions of the agent on its environment. This behavior is based on the roles played by the agent. The actions performed by the agent may be communication actions (sending and receiving messages) and actions of the domain application.

The multi-agent system design requires a description of the organization of its agents. The organizational aspect of a multi-agent system is very important to take advantage of social structures as well defined. Our meta-model describes the organization by a set of interacting roles. This organization is based on the Aalaadin methodology [16] where the organization is described by a group. A group is a composition of agents and roles. An agent can belong to
different groups, which implies that an agent can play more than one role and one role can be played by multiple agents in a group.

![Diagram of agent classification and role assignment]

**Figure 3. The Platform Independent Meta-Model (PIMM)**

The concept of role is very important in designing a multi-agent system because it affects the dynamic management of the system. The role is described by its structure and its behavior. The behavior performed by an agent, playing a role assigned previously, is achieved in the form of activities (actions) performed by the role.

The UML state machine meta-model [14] was selected to describe the behavior of the various roles of a multi-agent system. It is used as the target meta-model of the transformation process. It is a platform independent modeling language. Thus, PIM models automatically generated at this level from the CIM ones provide a more detailed description of the behavior of the various roles. The transformations of these models to PSM then to code could be performed and automated. Producing code is out of the scope of this work.
3.3. The Model to Model Transformation

The model to model transformation describes how the requirements (CIM) models are automatically transformed into UML state machines (PIM) models. It governs the derivation process by a set of transformation rules. A rule consists in transforming a concept outlined in the CIM meta-model to a corresponding one in the PIM meta-model. For this purpose, we define the function named \( \text{Transform} (\text{CIM}_\text{Concept}, \text{PIM}_\text{Concept}, \text{Messages}) \). This function performs some relatively complex mappings, although some of them are also quite direct. Some of these transformations are outlined next:

- Each \textit{Controlflow} concept is mapped to a \textit{Transition} concept.
- Each \textit{ControlNode} concept is mapped to a \textit{ControlState} concept.
- Each \textit{Sub-collaboration} concept is mapped to a \textit{CompositeState}. This composite state will hold the actions of the concerned role after transformation.
- Each \textit{Collaboration} concept is mapped to a \textit{CompositeState}. Then the function \textit{Transform} will trigger the transformation of its sub-collaborations.

The \textit{Messages} parameter in this function represents the coordination messages that will be included in the generated composite state to ensure the global coordination among the derived system roles. The result of applying the designed model to model transformation to the initial CIM is UML state machines platform-independent models. These models describe the behavior of the system roles. The corresponding transformation rules will be described below in Section 4.

4. Transformation Rules

The definition of the transformation rules, to govern the derivation process, requires the identification of various relationships between the CIM meta-model and the PIM meta-models. To perform the derivation, we identify several cases of choreography expressed in the CIM meta-model: sequential behavior, alternative (choice composition), competition (parallel composition), repetition as well as interruptions. At the level of collaboration, the starting roles (SR) and terminating roles (TR) are distinguished as in [12] (Figure 4).

![Figure 4. Structure of the Collaboration](image)

**Definition 1:** A starting role is a role that accomplishes an initial action in a collaboration or in one of its sub-collaborations.

**Definition 2:** A terminating role is a role that accomplishes a final action in a collaboration or in one of its sub-collaborations.

The sets of starting (SR), terminating (TR) and participating roles (PR) are calculated using the same techniques as in [12]. These sets are calculated for a collaboration depending on the
sequencing operators used in the activity diagram. Two types of sequencing are distinguished [17, 18]: Strong and Weak sequencing.

**Definition 3:** *Strong sequencing* (Figure 5) implies that all sub-activities of A1 are finished before an activity A2 can begin.

**Definition 4:** *Weak sequencing* (Figure 6) specifies that an activity A2 will be executed after another activity A1. Weak sequencing provides only a local order of activities for each system component and does not imply a global order.

In addition, two coordination messages are introduced in the derivation process to ensure the synchronization between the derived system roles. We use the same kind of coordination messages as introduced in [12]. Each coordinating message contains the parameters: a) Source role (Sr), b) Destination role (Dr) and c) name of state it belongs to (St). The coordination messages are:

1. Flow message for coordinating strong sequencing, named $Flowm(Sr, Dr, St)$.
2. Choice indication message for propagating the choice to a role that doesn't participate in the selected alternative in the choice composition structure, named $Choicem(Sr, Dr, St)$.

We have defined the transformation rules for the different cases of choreography expressed in the CIM meta-model. In the following, we express the rules that make it possible to derive the behavior of the different roles involved in a system global requirements specification for the cases of sequencing and choice composition. Each rule performs the appropriate model to model transformation for a role $r$ participating in a collaboration $C_i$. The transformation of the collaboration concept triggers the transformation of its sub-collaborations in the case where it is composed. This property requires the definition of recursive transformation rules.

**4.1. Strong Sequencing between Two Collaborations**

The flow message $Flowm(Sr, Dr, St)$ is used for synchronizing the strong sequencing between the derived behavior roles.

![Figure 5. Strong Sequencing between Two Collaborations](image)

**Rule 1:** The source collaboration $C_1$ is transformed into the composite state $S_1$, for a terminating role $r$ in $C_1$. This state $S_1$ will hold the actions performed by the role $r$ after transformation and include the actions of sending the coordination messages $Flowm$. The coordination message $Flowm$ is sent by the role $r$ to the starting roles of the target collaboration $C_2$ (except to itself if it is a member of this set of roles).
If \( r \in TR(C_1) \) then Transform\( (C_1, S_1, Send(Flowm(r, r', S_2))) \) \( \forall r' \in (SR(C_2) - r) \);

**Rule 2:** The target collaboration \( C_2 \) is transformed into the composite state \( S_2 \), for a starting role \( r \) in \( C_2 \). This state consists in the actions of receiving the coordination messages \( Flowm \) from the terminating roles of \( C_1 \) (except from itself) and includes the actions performed by \( r \) after transformation.

If \( r \in SR(C_2) \) then Transform\( (C_2, S_2, Receive(Flowm(r', r, S_2))) \) \( \forall r' \in (TR(C_1) - r) \);

**Rule 3:** The collaboration concept is transformed into a composite state, for a role \( r \) not terminating in the source collaboration or not starting in the target collaboration. This state holds the actions performed by the role \( r \).

If \( r \in (PR(C_1) - TR(C_1)) \) or \( r \in (PR(C_2) - SR(C_2)) \) then Transform\( (C_i, S_i) \) \( \forall i = 1, 2 \);

**Rule 4:** The controlflow concept is transformed into a transition, for a participant role \( r \) in the source and the target collaborations. This transition connects the states \( S_1 \) and \( S_2 \) obtained from the transformation of \( C_1 \) and \( C_2 \).

If \( r \in PR(C_1) \) and \( r \in PR(C_2) \) then Transform\( (Controlflow, Transition) \);

4.2. Weak Sequencing between Two Collaborations

In this case, no coordination message is used. To transform the controlflow concept, Rule 4 is applied.

![Figure 6. Weak Sequencing between Two Collaborations](image)

**Rule 5:** The collaboration \( C_i \) is transformed into the composite state \( S_i \), for a participant role \( r \) in the collaboration \( C_i \). This state holds the actions performed by \( r \).

If \( r \in PR(C_i) \) then Transform\( (C_i, S_i) \) \( \forall i = 1, 2 \);

4.3. Choice between Two Collaborations

We consider that the decision is made locally in a given role. Such role is the starting role in the collaboration choice structure (it is the starting role in both collaborations of the choice composition) (Figure 7). The Choice indication message \( Choicem(Sr, Dr, St) \) is used for propagating the choice to a role that doesn't participate in the selected alternative of the choice composition structure.
Rule 6: The collaboration concept $C_i$ is transformed into the composite state $S_i$, for a participant role $r$ in a choice composition and responsible for the choice. This composite state $S_i$ will hold the actions of the concerned role after transformation. As the role is responsible for the choice, it must send in parallel a coordination message $Choicem$ to all roles that do not participate in the selected collaboration $C_i$ alternative but are members of the other alternative $C_i'$.

If $r \in SR(C_i)$ then $\text{Transform}(C_i, S_i, \text{Send}(Choicem(r, r', S_i)))$ 
$\forall r' \in (PR(C_i) \cap PR(C_i'))$ and $\forall i, i'=1,2$ and $i \neq i'$;

Rule 7: The collaboration concept $C_i$ is transformed into the composite state $S_i$, for a participant role $r$ in a choice composition and not responsible for the choice. This composite state $S_i$ will hold the actions of the concerned role.

If $r \in PR(C_i)$ and $r \notin SR(C_i)$ then $\text{Transform}(C_i, S_i)$ $\forall i=1,2$;

Rule 8: For a role $r$ involved in one of the two collaborations of a choice structure, the collaboration concept is transformed into a composite state corresponding to the collaboration in which it doesn't participate. This composite state consists in an action for receiving the coordination message $Choicem$ from the starting role of the choice structure.

If $r \in (PR(C_i) - PR(C_i'))$ $\forall i, i'=1,2$ and $i \neq i'$ then 
$\text{Transform}(C_i, S_i, \text{Receive}(Choicem(r', r, S_i)))$ $\forall r' \in SR(C_i')$;

Rule 9: The Controlflow concept is transformed into a transition, for a participant role $r$ in a collaboration $C_i$. This transition connects the states $S_1$ and $S_2$ obtained from the transformation of $C_i$ and $C_2$ to the control states. When a guard is associated to a controlflow, the guard is associated to the generated transition in the derived model of the starting role.

If $r \in PR(C_i)$ then $\text{Transform}(\text{Controlflow, Transition})$ $\forall i=1,2$;

Rule 10: The DecisionNode concept is transformed into a ChoiceState, for a participant role $r$ in a collaboration $C_i$. This state is connected to the composite states $S_1$ and $S_2$ obtained from the transformation of $C_i$ and $C_2$, by the transitions obtained by application of Rule 9.

If $r \in PR(C_i)$ then $\text{Transform(DecisionNode, ChoiceState)}$ $\forall i=1,2$;
**Rule 11:** The MergeNode concept is transformed into a JunctionState, for a participant role $r$ in a collaboration $C_i$. This state is connected to the composite states $S_1$ and $S_2$ obtained from the transformation of $C_1$ and $C_2$, by the transitions obtained by application of Rule 9.

If $r \in PR(C_i)$ then $\text{Transform}(\text{MergeNode}, \text{JunctionState})$ \quad \forall i=1,2$

### 4.4. Example

An example is shown in this section to illustrate the derivation process. The CIM model of the system global requirements (Figure 8) is described by a composite collaboration. This collaboration is a sequence between the sub-collaboration $C_1$ and a composite collaboration named $C_2$. The collaboration $C_2$ defines a choice between the two sub-collaborations $C_3$ and $C_4$. We assume that each sub-collaboration consists in a single action.

![Diagram](image.png)

**Figure 8. Example of the Model to Model Transformation (Derivation Process)**

The Table 1 shows the calculated sets of the starting, terminating and participants at each collaboration involved in the system global behavior.
### Table 1. The Sets of Starting, Terminating and Participants Roles

<table>
<thead>
<tr>
<th>Choreography case</th>
<th>Starting Roles (SR)</th>
<th>Terminating Roles (TR)</th>
<th>Participants Roles (PR)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sub-collaboration “C₁”</td>
<td>{r1}</td>
<td>{r1}</td>
<td>{r1, r2, r3}</td>
</tr>
<tr>
<td>Sub-collaboration “C₃”</td>
<td>{r1}</td>
<td>{r1, r4}</td>
<td>{r1, r2, r3, r4}</td>
</tr>
<tr>
<td>Sub-collaboration “C₄”</td>
<td>{r1}</td>
<td>{r3}</td>
<td>{r1, r3}</td>
</tr>
<tr>
<td>Collaboration “C₂”</td>
<td>{r1}</td>
<td>{r1, r3, r4}</td>
<td>{r1, r2, r3, r4}</td>
</tr>
</tbody>
</table>

We consider in this example the transformation rules that allow deriving the behavior of the role r₁. The CIM model consists in a weak sequence between C₁ and C₂ and the role r₁ participates in the two collaborations. Therefore the derivation process triggers the rules 4 and 5. The rule 5 realizes the transformation of the actions that it performs at the sub-collaboration C₁ and triggers the transformation of the collaboration C₂. The rule 4 performs the transformation of the control flow in a transition that connects the composite states resulting from the transformation of C₁ and C₂. The collaboration C₂ is a choice composition between the sub-collaborations C₃ and C₄, and the role r₁ is responsible for the choice (it is a starting role in the two sub-collaborations). Therefore the derivation process triggers the rules 6, 9, 10 and 11. The rule 6 performs the transformation of the actions that it performs at each sub-collaboration. On the other hand, it must inform the roles r₂ and r₄ not participating in the collaboration C₄ by sending a coordination message Choicem for indicating the choice of C₄. The rules 9, 10 and 11 perform the transformation of the control flow and the control nodes. The derivation process generates a PIM model describing the r₁ behavior (Figure 8).

### 5. Case Study

In this section, we present an application of telediagnosis in neuroscience. Conventionally, when a patient with a stroke is admitted into a hospital (HA), he will be examined by an emergency doctor. The emergency doctor contacts the EMS (Emergency Medical Services) via a regulating doctor to inform him of the emergency admission of a patient who presents symptoms to suspect a stroke (speech disorders, vision disorders, sensor-motor deficits, coordination disorders, etc.). Within the EMS information system, the regulating doctor creates a medical record based on an initial evaluation using the GCS (Glasgow Coma Score), the hemodynamic status, time of occurrence of signs, time of admission to emergencies, background, anamnesis data and clinical examination, etc.

While remaining in contact with the emergency doctor, the regulating doctor of the EMS calls the neurologist on duty at UHC (University Hospital Center) and initiates a conference call in order to establish the diagnosis. If appropriate, the patient is urgently transferred to the University Hospital Center within an equipped ALS (Advanced Life support) or not-equipped ACA (Ambulance Care Assistance) ambulance depending on the patient health situation.

This application can be modeled by a multi-agent system. The agents can play the roles: Neurologist (UHC), Emergency doctor (HA), Emergency Medical Services (EMS), Advanced Life Support (ALS), Ambulance Care Assistance (ACA), etc. The derivation process follows two steps.

The first step specifies the global requirements model (CIM) describing the system global behavior. The CIM model is described by an activity diagram whose core activities are collaborations and sub-collaborations (Figure 9). It is described by a collaboration consisting in a weak sequence between the Clinical sub-collaboration and the composite collaboration named Clinical-Decision. The Clinical-Decision
collaboration is composed of a weak sequence between the *Para-clinical* sub-collaboration and the *Decision* collaboration. The *Decision* collaboration is itself composed of a weak sequence between the *Decision-Making* sub-collaboration and the *During-Transfers* collaboration. The last one consists in a choice composition between the collaborations: *Supported by HA* and *Transfer* which is also a choice composition. The collaboration *Transfer* defines a choice between the two sub-collaborations *Sending ALS* and *Sending ACA*.

**Figure 9. The System Global Requirements (CIM)**

The second step consists in applying the models transformation rules to the CIM model. These rules generate a state machine (PIM model) describing each role behavior. The derived PIM model includes the messages required for ensuring the global coordination among the different system roles. We consider in this example the transformation rules that allow deriving the behaviors of the equipped ambulance (ALS) and the neurologist (UHC) roles.
The ALS behavior derivation can be summarized in the transformation of *During-Transfers* collaboration because the ALS does not participate in the *clinical, Para-clinical* and *Decision making* sub-collaborations. At the *During-Transfers*, the ALS is a participant role in *Transfer* collaboration but not in the *Supported by HA* sub-collaboration. Therefore the derivation process triggers the rules 7, 8, 9, 10 and 11. The rule 7 realizes the transformation of the *Transfer* collaboration. The rule 8 allows the ALS to receive the coordination message sent by the neurologist in case of choice of the *Supported by HA* sub-collaboration. The reception of this message triggers the continuation of the treatment at this sub-collaboration. The rules 9, 10 and 11 perform the transformation of the control flow and the control nodes. The transformation of the *Transfer* collaboration which is a choice composition is done in similar manner as the previous situation. As the ALS participates in the *Sending ALS* sub-collaboration and not in the *Sending ACA*, the derivation process triggers the rules 7, 8, 9, 10 and 11. These rules accomplish the transformation of the actions performed by the role as a participant at the *Sending ALS* sub-collaboration, allow the reception of the coordination message in case of choice of the *Sending ACA* sub-collaboration and perform the transformation of the control flow and the control nodes respectively. The derivation process generates a PIM model which describes the ALS behavior (Figure 10).

Figure 10. The Derived PIM Describing the ALS Behavior
The neurologist (UHC) behavior derivation is obtained with the same manner as described in the ALS derivation. The (Figure 11) shows the derived PIM model describing the UHC behavior.

Figure 11. The Derived PIM Describing the UHC Behavior

The meta-models and model transformations outlined in the previous sections have been developed using the facilities provided by the Eclipse platform. Eclipse platform is a free open source environment. It offers some widely used implementation of the OMG standard Meta Object Facility (MOF) [19], called Eclipse Modeling Framework (EMF) [20]. The architecture (Figure 12) shows the needed components to implement this development process. The architecture consists in three blocs:

1. The first bloc consists in representing:
   (i) The requirements meta-model (CIMM), UML activity diagram extended with collaborations,
   (ii) The PIM meta-model, UML state machine meta-model, and
   (iii) The CIM model which represents a given system global requirements.

Figure 12. The Derivation Process Architecture
These meta-models and models are created using EMOF (Essential MOF). EMOF allows designers to create, manipulate and store both models and meta-models.

2. The second bloc is the main component of this architecture. It is responsible for the behavior derivation. It consists in: the model to meta-model conformity component, named M2MMC and the Derivation process rules. The M2MMC carries out the compliance of the CIM model with its meta-model. This compliance is achieved by the EMOF. The compliance of the derived behaviors with the PIM meta-model is guaranteed by the Derivation process rules. The Derivation process rules performs the model to model transformation. It realizes the behavior derivation of the different roles from the CIM model. This derivation is designed to generate the different concepts of a state machine for each system role.

There exist several languages for model transformations. Among these languages, we mention the ATL language (ATLAS Transformation Language) [21-23] and the QVT language (Query View Transformation) of the OMG [24, 25]. Both languages exhibit a layered architecture and share some common characteristics as they initially shared the same set of requirements defined in QVT RFP [26]. ATL and QVT languages have a similar operational context [27, 28]. The transformation rules of the derivation process are expressed in ATL language which provides the standard Eclipse solution for model to model transformations.

3. The third bloc consists in the derived UML state machines (PIM) that reflect the local behavior of each role. These models are platform independent models which could be used for an automatic generation of code for multiple platforms.

7. Conclusion

The work presented in this paper offers a model driven approach to derive the behavior of a multi-agents system roles from its global requirements. The proposal presents a high level of abstraction using UML activity diagram extended with collaborations meta-model which allows designers to describe the system global behavior model (CIM model). This CIM meta-model allows representing concepts that could not be described using behavior expressions. The derived behavior of the system roles (PIM models) are UML state machines which could be used for an automatic generation of code. Automatic model transformations from the CIM models to PIM models have been designed to govern the derivation process. They have been implemented using the ATL language "Atlas Transformation Language". We have considered the transformation of the conditions (guards) embedded in the choice structures and the repetition structure. In addition, our transformation approach considers also all control nodes specified the UML activity diagram such as Merge Node and Join Node. The PIM models include the messages required for realizing the collaborations and for ensuring the global coordination among the different system roles. An Emergency Medical collaborative application has been used to test and illustrate the development approach. We plan also to extend the derivation process by including the derivation of the detailed behavior of a sub-collaboration. Such behavior could be described as a sequence diagram. The derivation of a sub-collaboration behavior will be achieved by transforming the sequence diagram into a state machine using a similar model transformations mechanism.
References


Authors

Ahmed Harbouche has been an Assistant professor in Computer Science. He obtained his Magister in 1993 at Houari Boumedienne University (Algiers, Algeria) in the area of Artificial intelligence. He is member of the research team “models engineering” of the department of computer science at Hassiba ben Bouali University (Chlef Algeria). His research interests pertain artificial intelligence, multi-agents systems and Software Engineering.

Mohammed Erradi has been a Professor in Computer Science since 1986. He has been leading the distributed computing and networking research group since 1994 at ENSIAS of Mohammed V-Souissi University (Rabat Morocco). His recent main research interests include Communication Software Engineering, Distributed Collaborative Applications, Security Policies, Reflection and Meta-level Architectures. He obtained his Ph.D. in 1993 at University of Montreal in the area of Communicating Software Engineering under the supervision Professor Gregor Von Bochmann. He is leading, at the present time, TENEMO project (A Collaborative Environment for Neuroscience Tele-diagnosis over a Mobile Platform) funded by a French-Moroccan joint research program (2008-2011). He also is currently the Principal Investigator of a number of research projects grants. Professor Erradi has published more than 60 papers in international conferences and journals. He has organized and chaired four international scientific events and has been a member of the program committee in multiple international conferences.

Aicha Mokhtari is currently a Professor in USTHB Alger’s University, Computer Science Department. Her research focuses on reasoning about knowledge and uncertainty, and its applications to security, web semantic, recommender system and data bases. Her work lies at the boundary of a number of fields. I usually teach data bases in an undergraduate course and knowledge representation and reasoning in a postgraduate course.