MaCMAS/UML: A Methodology Fragment for the Analysis Stage of Large Complex/Complicated Multi–Agent Systems* WORKING DOCUMENT v.2.1 (19-June-04)

Joaquín Peña and Rafael Corchuelo

The Distributed Group
University of Seville
Avda. de la Reina Mercedes, s/n. Sevilla 41.012 (Spain)
Phone: +34 954 55 38 65, Fax: +34 954 55 71 39
E–mail: joaquinp@us.es, web page: www.tdg-seville.info

Abstract. The main purpose of Agent–Oriented Software Engineering is to provide mechanisms to conquer complexity. The design phase requires an understanding of the Multi–Agent System which allows us to implement it. As a matter of fact, most complexity issues must be properly managed at the analysis stage. In this paper, we propose a methodology fragment for the analysis stage which is specially tailored to deal with complexity. Our main contribution is the use of abstraction, decomposition and composition techniques to conquer complexity and their integration in the software process. We conquer complexity producing a layered description of the system which is obtained iteratively, incrementally and systematically.
keywords: Complex systems, Multiparty Interactions, Abstraction, Decomposition, Composition, UML.

1 Future Extensions

This document is in continuous extension and may contain some inaccuracies. We are currently working in the following extensions:

- Integration of our software process with some methodologies.
- Improvements on the composition and reuse facilities to: (i) improve reuse of yet developed agents and models by means of abstracting their models to meet the requirements of the new system.
- We are improving the UML graphical notation, meta-models, and the case study.

Please, check for updates at *http://tdg-seville.info/joaquinp/MacMAS/documentation.htm*

* The work reported in this article was partially supported by the Spanish Ministry of Science and Technology under grant TIC2003- 02737-C02-01.

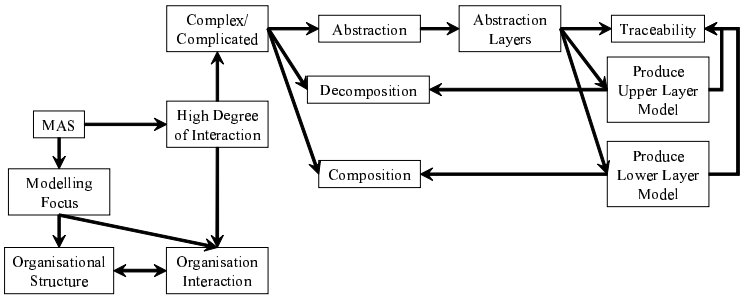


Fig. 1. Conceptual Map of Motivation

2 Introduction and Motivation

2.1 Dealing with Large Complex/Complicated Multi–Agent Systems

Complexity is one of the main problems of current software development. The Agent–Oriented Software Engineering is a new paradigm which basing on agency promises to increase the complexity of software systems we will able to engineer and implement. This problem is well known in the traditional software engineering field. G. Booch, identified the three vertebral principles of software construction in order to cope with the overgrowing complexity which where adapted to the AOSE field in [6,5] as follows: (i)**Abstraction:** it is based on defining simplified models of the system that emphasises some details avoiding others. It is interesting since it limits the designer scope of interest and the attention can be focused on the most important details at a given time, (ii) **Decomposition:** it is based on the principle "divide and conquer". It helps to limit the designer scope to a portion of the problem, and (iii) **Composition:** it consists on identifying and managing the inter–relationships between the various subsystems in the problem. It makes possible to group together various basic components and treat them as higher–level units of analysis, and, provides means of describing the high-level relationships between several units.

To properly deal with complexity we should apply these principles carefully performing a layered description of the system. Unfortunately, by the best of our knowledge, composition and decomposition principles have not been conscientiously applied in the AOSE field and most approaches perform a plain model of the system. In this paper, we present the MaCMAS/UML (Methodology for Analysing Complex Multi–Agent Systems) which is a methodology fragment limited to the analysis stage specially tailored to deal with complexity by means of the three principles. It improves on others in that we provide a layered description and the techniques to support it: each model in an abstraction layer can be decomposed to promote to a lower layer or composed/abstracted to promotes to a higher layer. Our approach also provides: (i) traceability between

layers, between requirement and analysis and between analysis and design, (ii) reuse mechanisms, (iii) modular descriptions, and (iv) means for dealing with open system where we know interactions patterns at modelling time but not the concrete agents who participate on them.

2.2 Focusing the Modelling Process: Structure *vs.* Interaction

The organisational metaphor has been proved one of the most appropriate tools to engineer predictable Information Multi-Agent Systems (hereafter MAS). This metaphor is used by many researches to guide the analysis and design of MASs, e.g. [9,12,16]. In Agent Oriented Software Engineering (hereafter AOSE) "organisation" is a polysemous term that must be treated carefully. A MAS organisation can be observed from two different point of views [2]:

Acquaintance point of view: it shows us the organisation as the set of interaction relationships between agents.

Structural point of view: The later shows us agents as artifacts that belong to sub-organisations, groups, teams. In this view agents are also structured into hierarchical structures showing the social structure of the system.

Notice that both views may relate, but they show the organisation from radically different point of views: a relationship between several agents in one of them, do not necessarily implies a relationship in the other. For example, the group of teachers of a subject are grouped in a team because they teach the same subject, but it does not necessarily implies any acquaintance relationship between them regarding the subject.

Furthermore, many authors agree on that the main source of complexity[1] of MASs is consequence their interacting nature of agents, e.g. [5,8]:

> *Complexity is caused by the collective behaviour of many basic interacting agents.* James Odell [8]

As a matter of fact, to properly conquering complexity, modelling process should be focused on interactions with out taking into account structural constraints that may further complicate our task. As we show in this document, acquaintance organisation can be modelled independently from structural organisation. Later, the set of acquaintance relationships can be mapped over the structural organisation at design stage. Many researches perform the modelling process taking into account both organisation views in parallel or do not clearly distinguish between them what changes the attention to issues that are not the main source of complexity. We think that to properly deal with complexity we must focus on acquaintance organization to map it later onto the structural organization. We think that the modelling process must be focused on interactions and should base on previous principles to properly conquer complexity.

[1] *Complexity* is a vague term that must be further described. See Section **??** for a detailed discussion.

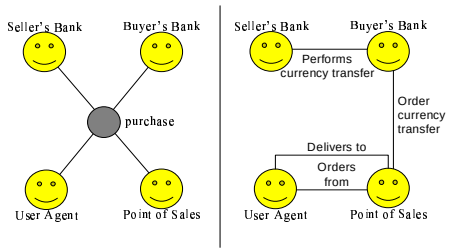


Fig. 2. Multiparty organisation relationships *vs.* biparty relationships

This paper is organised as follows: In Section 3, we present the related work. In Section 5, we describe the models and metamodels of our methodology fragment. In Section 6, we detail the software process we propose and the decomposition and composition techniques. In Section 7, we show a case study to illustrate our methodology and we show the UML2.0 notation. And finally, in Section 8, we summarise the main features of the MaCMAS/UML methodology fragment.

3 Related Work

Abstraction: Multiparty relationships (relationships that can involve an arbitrary number of roles) provides means for more abstract interaction models than biparty links such as associations. For example, as it is depicted in Figure 2(left) a purchase where a Buyer's Bank a Seller's Bank, an User Agent and a Point of Sales Agent participates can be represented as a conceptually atomic social relationship at some level of abstraction. If it is represented by means of biparty links we are forced to divide it mentally (Figure 2(right)), thus decreasing the level of abstraction. In large MAS, this implies to decrease the level of abstraction of models from the beginning, which in consequence decrease our capacity of dealing with complex systems.

Most AOSE approaches, e.g. Tropos, Prometheus, MaSE, PASSI, AUML notation and MESSAGE (see [1] for a summary of these approaches), model interactions between roles as binary UML associations which force designers to decompose mentally multiparty interactions not allowing abstraction. GAIA is the only methodology that provides multiparty interactions, i.e. protocols [16, pag. 32]. Unfortunately, GAIA does not provide an UML–based notation. Some of the previous approaches provide multiparty relationships in some sense. AUML provides multiparty interactions, i.e. nested protocols, used to represent the behaviour aspect (how interactions sequence) of an organisation but not at the acquaintance aspects (interaction relationships) in their organisation models [12]. MESSAGE borrows the multiparty interaction concept from GAIA and provides an UML notation. Unfortunately, MESSAGE's interactions are not usually used but tasks and workflows.

Furthermore, using multiparty links to represent the acquaintance organisation requires for tailored tools to represent their sequences of execution not limited to biparty primitives such as messages as most approaches do, e.g. [1]. GAIA represents such order by means of regular expressions assigned to each role. Unfortunately, they do not provide an UML–based graphical notation nor a way of representing the whole behaviour using a single model which may be easily understood than distributed regular expressions–based description. MESSAGE does not represent the sequences of interactions or acquaintance relations. Although MESSAGE workflows represent the sequence of tasks that a single agent performs, authors do not describe how interactions, acquaintance relations and workflow relate.

Finally, by the best of our knowledge current methodologies do not provide layered descriptions. The only approaches that covers a layered description are [4,12]. Unfortunately, both approaches base only on recursive agent abstractions not providing decomposition and composition techniques which limit their ability to deal with complex systems.

Decomposition: Role modelling techniques are widely accepted and allows us to decompose a MAS into a set of organisations that can be superposed in the design stage. All methodologies support role models but techniques to guide such decomposition are needed. In this paper, we base on two techniques: (i) functional decomposition by sub–goals of the system and (ii) dependency analysis techniques which allow identifying group of roles that loosely depend on others. The former has been identified by most authors but not developed. Unfortunately, by the best of our knowledge GAIA and MESSAGE are the only methodologies that identify the later but without describing how to perform it.

Composition: Decomposing the system into a set of orthogonal problems it is usually impossible. Hence, isolated models loose some features in the decomposition process (the whole is greater than the sum of its parts). For example, given a book store MAS where we search for books and pay these books later, we can separate the search and the payment problems into different organisations. But, for example, the restriction on the book dealers we can search (only those which accept our credit card) can not be modelled in either sub–organisation. Thus, the decomposition of a MAS into a set of role models requires a mechanisms that, although we describe each of them isolated, allows us to later relate them and identify the features loosed as consequence of decomposition. By the best of our knowledge there not exists any methodology that provides techniques to perform the composition of organisation models.

4 Applicability Context

In the field of complex organisational knowledge exchange, decision-making, strategy, and policy-making, Snowden *et al.* proposed the Cynefin Framework which clarifies complexity term providing a taxonomy of a knowledge–based organisation regarding complexity and predictability [15]. This taxonomy divides an organisation into the following domains:

1) Ordered Domain: Stable cause and effect relationship exist. In this domain the behaviour of the organization can be established as a cause/effect chain. It represents the predictable part of the system. This domain is further divided into:

- **1.2) Known Domain:** Every relationship between cause and effect are known. The part of a MAS in this domain is clearly predictable and can be easily modelled.
- **1.1) Knowable Domain:** While stable cause and effect relationships exist in this domain, they may not be fully known. In general, relationships are separated over time and space in chains that are difficult to fully understand. The only issue is whether we can afford the time and resources to move from the knowable to the known domain.

2) Un-ordered: Un–Stable cause and effect relationship exist between interactions in the system. It represents the unpredictable part of the system. This domain is also further divided into:

- **2.1) Complex Domain:** There are cause and effect relationships between agents, but both the number of agents and the number of relationships defy categorization or analytic techniques. Unfortunately, relationships between cause and effect exist but they can not be predicted. This domain presents retrospective coherence. That is to say, coherence can be only established by analysing past history of the system. Unfortunately, future directions, although coherent, can not be predicted.
- **2.2) Chaos Domain:** There are no perceivable relationships between cause and effect, and the system is turbulent; we do not have the response time to investigate change

Our approach is specially tailored to deal with complicated organisations at the "Ordered Knowable Domain" bringing it to the Known Domain. Regarding Complex domain, some of the mechanisms we propose could be also useful to model past patterns with their preconditions and retrospective history in order to infer the rules that will induce to emerge future desirable interactions patterns.

For these domains, we focus only on the analysis stage of multi–agent systems where exist *Correlation* between agents (exists joint information) and whose agents acts *Coordinately* (implies a causal process where communication between agents exists either directly or indirectly through the environment). We take into account agent's and system's goals thus covering system that coordinates by *Contention* (agents that coordinate with contradictory goals) or by *Cooperation* (agents with non–contradictory goals). The kind of system we focus also must present a certain degree of *Congruence* (agents goals fulfil system goals even when a *Contention* mechanisms exits). Hence, in the kind of system we focus agents must relates *Coherently* (the relation among the agents that yields *Congruence* is *Coherence*). All the Co–X terms we use can be found [11]). Thus, we focus on complex reliable information systems with a predictable behaviour. The modelling artifact, techniques and guidelines we provide are also applicable to open system where we know interactions patterns at modelling time but not the concrete agents who participate on them.

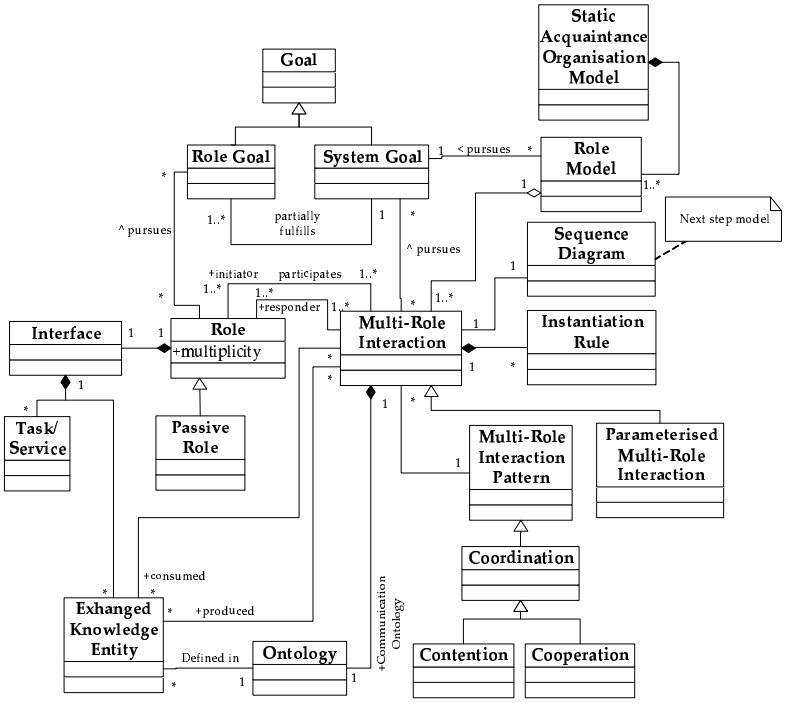


Fig. 3. Metamodel of Role Models

We focus on system where the acquaintance relations between agents are not pre–established in the real organisation where the system has to be deployed. If preestablished acquaintance organisation which does not present mistakes, exits, we can start the modelling process from it not needing most features proposed in this document.

5 The MaCMAS/UML Models and Metamodels

A MAS Metamodel describes the elements/concepts and their relationships that a certain methodology use in their models. It is a crucial element of a methodology since it describes concisely and without ambiguity how models for a certain system can be built (instantiated) and the constraints in the use of their constituents' elements. Metamodels are more crucial, if possible, when defining a methodology fragment as it is our case. A Methodology fragment can not be used without integrating it with other fragments or methodologies. In this scene, establishing the relations between elements in metamodels taken into account represents a good starting point to later integrating software processes, notation, and so on.

The MaCMAS/UML modelling process is focused on interactions since they are the main source of complexity. In order to represent interactions abstractly

we propose *multi–Role Interactions* (mRI) [14,13]. mRIs are first class modelling elements in our models and are used as the minimum building block at static and dynamic aspects of modelling. Their use is crucial for performing an incremental layered modelling approach since mRIs can be described internally by means of finer–grain mRIs, or several of them can be abstracted by coarse–grain one. The main advantages of our approach regarding models/metamodels are:

1. Interactions are the central concept of our fragment, and thus, of all metamodels we propose. It improves our ability to deal with systems with a high interaction degree.
2. An interaction can relate an arbitrary number of roles (called multi-Role Interactions) and can be refined by several finer–grain interactions. Both facts allow us to perform a layered description of the system which improves our ability to deal with complexity through an iterative and incremental process.
3. Interactions correlates with systems goals. It allows establishing a clear traceability between requirements and final models produced by our approach.
4. Interactions can be reused at any level of abstraction.

An mRI is an *institutionalised pattern of interaction* that represents abstractedly the fulfillment of a system goal without detailing how this achievement is carried out. Thus, using mRI as the minimum modelling element we do not have to take into account all the details required to fulfill a complex system goal nor the messages that are exchanged at stages where these details have not been identified clearly or even are not known (we do not use message sequences in our fragment but we leave it for next fragments). This direct correlation between system goals and mRIs allows us to establish a clear traceability between goal-oriented requirement documents and analysis models and it also improves our ability to model congruent systems [11]. As it is shown in the Meta Model of Figure 3, an mRI is related with the following modelling artifacts:

1. the system goal it fulfills,
2. the knowledge consumed and produced by the interaction,
3. the interaction ontology describing the knowledge that is processed in the interaction.
4. the coordination mechanism it follows: contention or cooperation,
5. an instantiation rule which shows the constraints over the agents that may play the roles in the interaction.
6. and the roles that participate on it (notice that we do not include the agent concept since we leave it for the next fragment where role must be mapped onto agents):
   (a) the goals of roles,
   (b) the cardinality of roles (number of agents that may play it),
   (c) the initiator/s role/s, the role interfaces: knowledge and services provided by each role,
   (d) a guard for each role that represent if the agent playing a role wants to participate in the mRI or not.

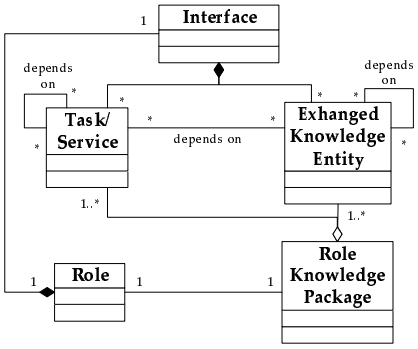


Fig. 4. Knowledge Dependencies Meta-Model

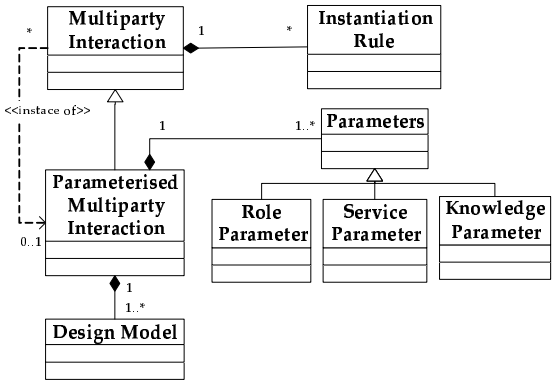


Fig. 5. Parameterised mRI Metamodel

mRIs are represented with UML 2.0 collaborations [10, pag. 132] as all the models we propose. We propose three views of the acquaintance organisation: Two for representing the static and dynamic aspects of the organisation and a third for representing the relation between models in different abstraction layers. We use the following models:

a) Static Acquaintance Organisation View: It shows the interaction relationships between roles in the system statically and the knowledge processed by them. It comprises the following UML models:

Role Models: It is described in metamodel of Figure 3. It shows a set of agent roles collaborating by means of several mRIs. Notice that a role model may also contain a single mRI.

Ontology: It shows the ontology shared by roles in a role model. It is used to add semantics to the knowledge owned and exchanged by roles (see [3] for an UML–based ontology notation).

Knowledge view: Its metamodel is depicted in Figure 4. It shows the knowledge processed by each role in an mRI and their dependencies.

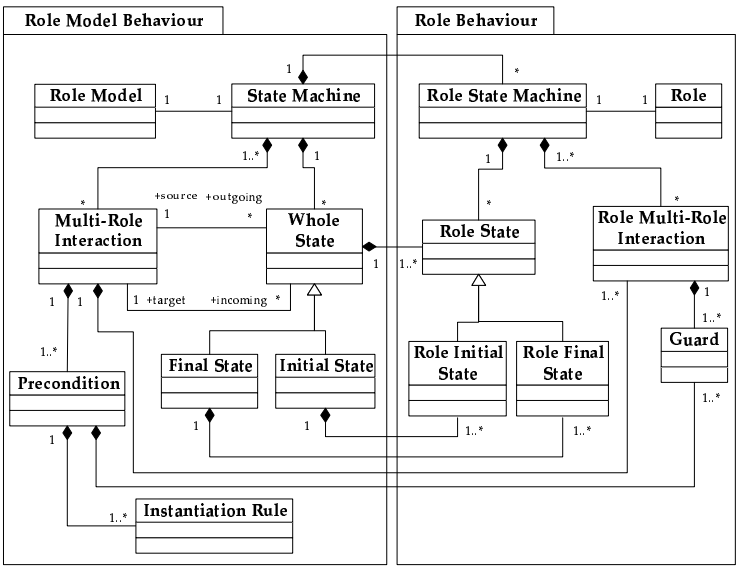


Fig. 6. Behaviour Organisation View Meta-Model: Role and Role Model Behaviour

It consists of a set of UML 2.0 packages, one for each role, where we place the knowledge and services each role provides and their dependencies. It is used to understand how the interaction is carried out with out entering into details and to perform dependency decomposition (see [13] and Section 6.2 for further details).

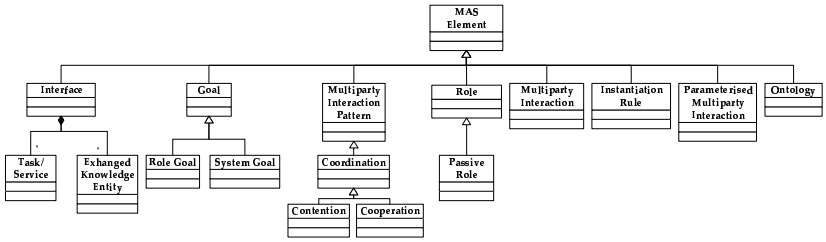
Parameterised mRIs: Its metamodel is depicted in Figure 5. Some interactions patterns can be generalised to reuse them. Roles, knowledge and services can be parameterised which allows to reuse them with concrete roles/agents and certain knowledge and services. It is represented using UML 2.0 collaboration templates [10, pag. 509]. The instances of parameterised mRIs can be used on Role Models.

b) Behaviour of Acquaintance Organisation View: The behavioral aspect of an organisation shows how the interactions in a certain role model sequence. It is represented by two equivalent models (the metamodel of both views is depicted in Figure 6):

Behaviour of a role: It represents separately the behaviour of each role in a role model showing how the mRIs of the role sequence. It is represented using UML 2.0 ProtocolStateMachines [10, pag. 422]. It is used to focus on a certain role ignoring others.

Behaviour of a role model: It represents the order of mRIs in a role model with a centralised description. It is represented using UML 2.0 StateMachines [10, pag. 446]. It is used to understand easilily the whole behaviour of a sub–organisation.

c) Traceability view: This model shows how models in different abstraction layers relate. As it is shown in the meta–model of Figures 7, it shows how



MAS Elements Meta-Model

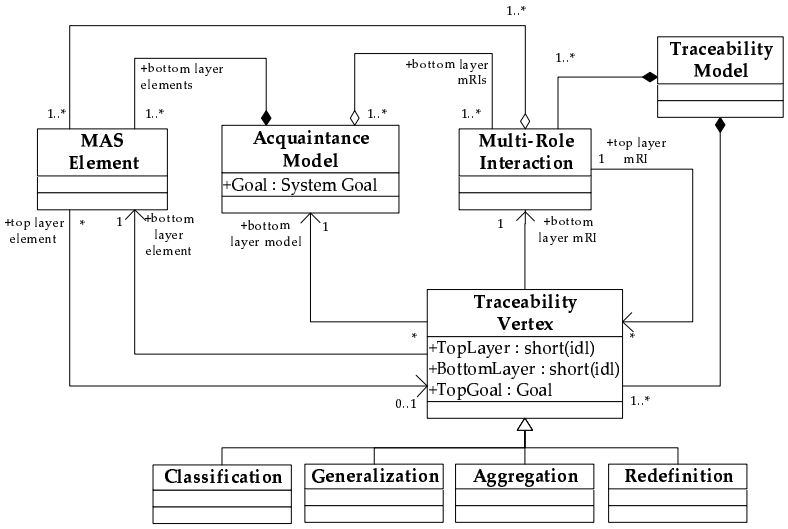


Fig. 7. Traceability Meta-Model

mRIs are abstracted, composed or decomposed by means of *classification* (when a parameterised mRI is instantiated), *aggregation*, *generalisation* or *redefinition*. Notice that we usually shows only the relations between interactions beacuase they are the focus of modelling, but all the elements that conform an mRI can be also related. Finally, since an mRI presents a direct correlation with system goals, traceability models clearly shows how a certain requirement system goal is refined and materialised.

6 The MaCMAS/UML Process Description

Our approach is focused on the acquaintance organisation analysis stage and starts when goals have been already identified in the requirements stage. It finishes when we obtain the sufficient level of detail to understand the acquaintance organisation and map it onto certain organisation structure (*Structural Organisation Analysis* stage of Figure 8).

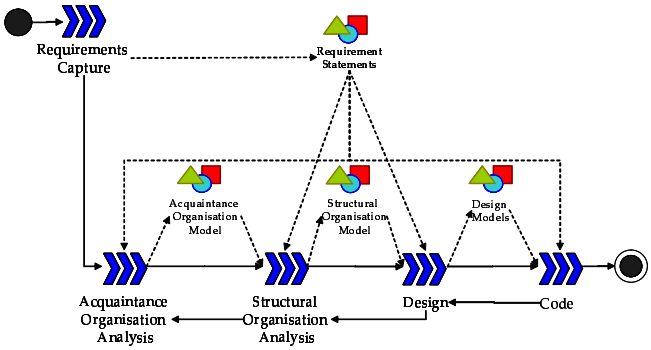


Fig. 8. Overall Process

Table 1. Summary of SPEM stereotypes we use

Stereotype	Name	Description
	Phase	Represents a phase of a software process
	Work Definition	Represents a complex task
	Activity	Represents each task in a Work Definition
	Work Product	Represents the results of software process
	Document	Represents a document generated in the process
	UML model	Represents UML models used in the process
	MAS Model Element	Represents elements of the UML models

We describe the software process of our approach using SPEM. The SPEM metamodel is used to describe a concrete software development process or a family of related software development processes. It is an UML profile that use a set of stereotypes (see table 1 for a summary of them). For further details see http://www.omg.org/technology/documents/formal/spem.htm.

In Figure 9, we show an overview of the *Acquaintance Organisation Analysis* stage. The first step is to produce an initial set of role models for higher level requirement goals which conforms the *Initial Static Acquaintance Organisation Model* (*Build a role model for each goal* work definition). This first set of models provides a superficial understanding of the system to be built which is augmented through an incrementally and iterative process. Then, three tasks are performed in parallel:

***Layer Completion* work definition (Layer-C-WD):** It works with one or several models in order to produce one or more abstract or refined models. New models are placed in upper layers or lower layers, thus performing a top–down or bottom–up design as needed.

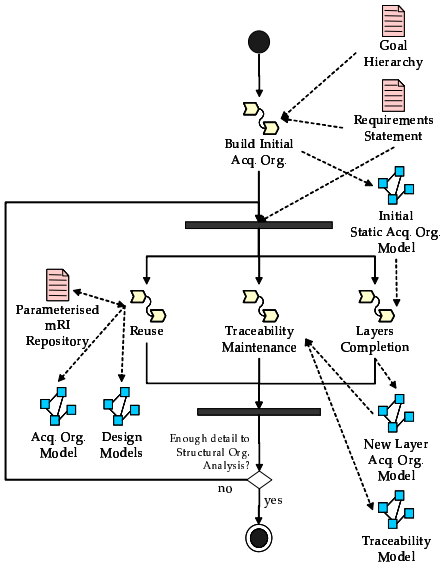


Fig. 9. Acquaintance Organisation Process

***Traceability Maintenance* work definition (Traceability-M-WD):** It updates the *traceability model* which document the relations between models in different layers and requirements system goals.

***Reuse* work definition (Reuse-WD):** It instantiates *Parameterised mRIs* stored in the Interaction Patterns Repository when appropriate. It finish the modelling process in this stage for the correspondent sub–part of the system since *Parameterised mRIs* are attached with design models.

Following, we detail previous work definitions:

6.1 *Build Initial Acquaintance Organisation* Work Definition

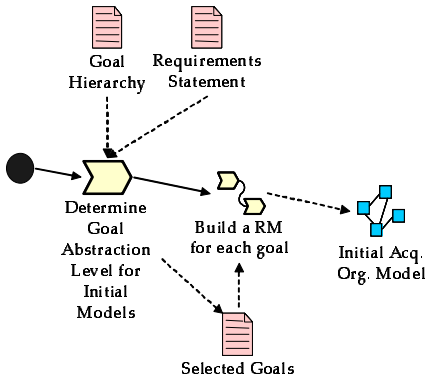


Fig. 10. Build Initial Acquaintance Organisation Work Definitions

This work definition is detailed in Figure 10. It purpose is to provide an initial abstract model of the system and sub-systems. The inputs are the requirements statements and a system–goals hierarchy which shows the goals of the systems and their relations. The output is an initial static acquaintance organisation model formed by a set of role models, one per system goal. Each of these role models models the relations between roles by means of a single mRI (which is the most abstract way of representing a system goal). Notice that we do not have to model how systems goals relate nor to specify the order in which goals have to be achieved.

The process consists on identifying which system goals in the hierarchy we are able to model almost directly using requirements information. These goals are usually the most general, that is to say, top goals in the hierarchy. Later, we provide a role model for each selected system goal (see *Build a Role Model for a Goal* Work Definitions for further details). Notice that starting the modelling with too lower goals may be difficult until we do not clearly understand the overall process (represented by top goals).

6.1.1 *Build a Role Model for a Goal (Initial-AO-WD)* Work Definitions This work definition is detailed in Figure 11. It consist on providing a role model which contains the the roles, interactions, etcetera, needed to fulfill a system goal. It inputs are a system goal and an *Static Acquaintance Organisation Model*, and it output is an static *Acquaintance Organisation Model* update with the new role model.

The process consist on identifying all the elements needed in a Role Model and it correspondent communication ontology.

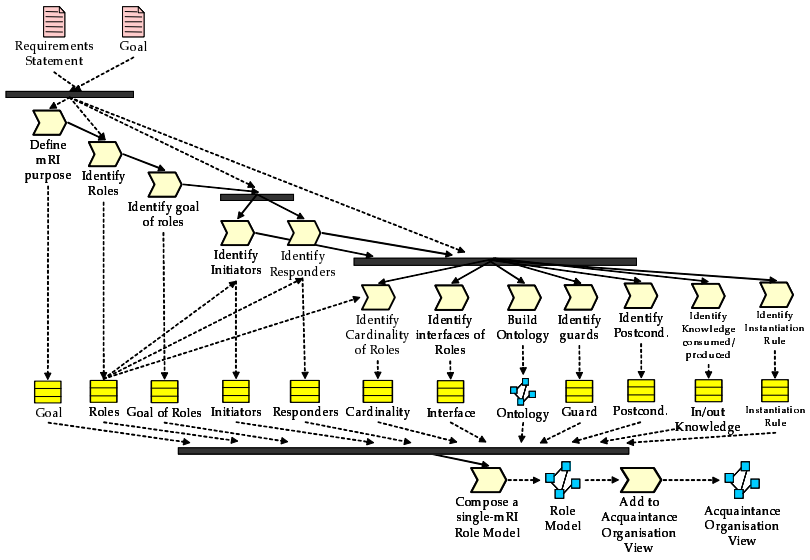


Fig. 11. Build a Role Model for a Goal Work Definitions

6.2 *Layer Completion* Work Definition (Layer-C-WD)

This work definition is detailed in Figure 12. It consists on producing a new Acquaintance Organisation Model as a result of the composition or the decomposition of a model developed in a previous iteration. The new model produced is used to fill a top or bottom layer. Each model describes a part of the system which represents the interactions and participants involved in the achievement of a certain system goal. We recommend describing models without taking into account relationships with other parts of the system since it limit our scope to a portion of the problem and eases the modelling process. Later, when we clearly understand each sub–part, the relationships between parts can be modelled composing their corresponding models. In order to decide between composition and decomposition we provide several criteria:

- if it exits system goal dependencies between several models and the level of detail of these models is adequate we recommend composition.
- if it exits system goal dependencies between several models and the level of detail of these models is not enough for composition, we recommend decomposition and add details. Thus, in the next iteration we try to compose them again with the refinement performed.
- if it does not exits system goal dependencies between several models and the level of detail of these models is not adequate, we recommend decomposition. Notice that models that present a level of abstraction adequate for the next stage are not coped with by this work definition (see Work Definition of Figure9).

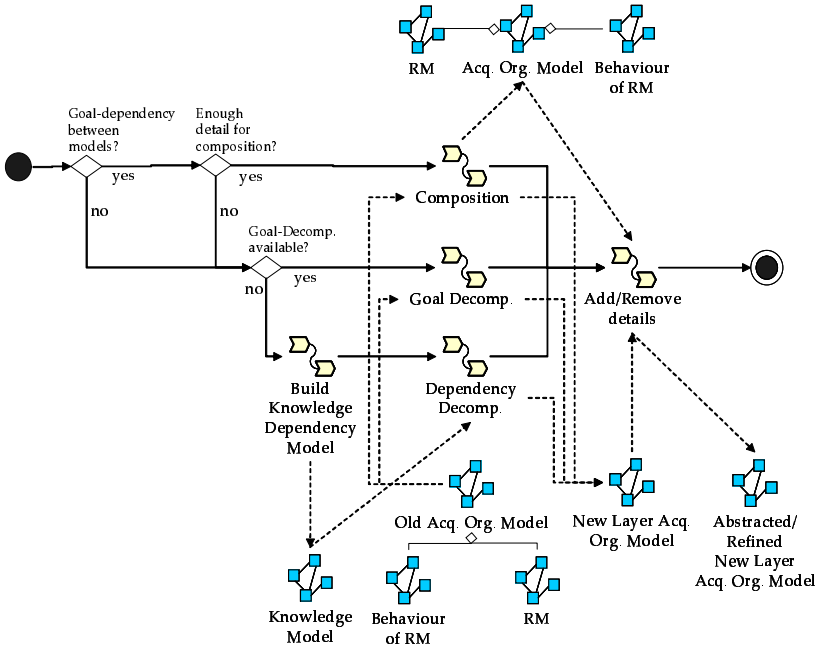


Fig. 12. *Layer Completion* Work Definitions

Regarding decomposition, notice that we provide two decomposition techniques: (i) goal decomposition and (ii) dependency decomposition. The former consist on decomposing a certain model basing on the information in the goal hierarchy model (see *Goal Decomposition* work definition for further details). The later is used when we do not have more information to perform a goal decomposition and it consist on analysing knowledge dependencies between roles in a Role Model (see *Dependency Decomposition* work definition for further details).

Finally, we add or remove details (*Add/Remove* work definition) depending on if we are developing a model for a top layer or a model for a bottom layer.

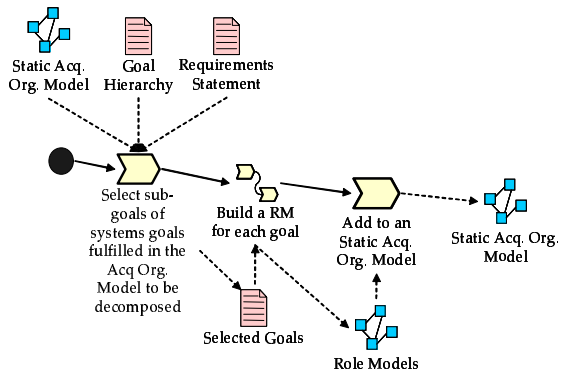


Fig. 13. *Goal Decomposition* Work Definition

6.2.1 Goal Decomposition (Goal–Decomp–WD) It consists on producing a model for each system sub–goal of the system goal of the model we are decomposing. These models can be composed later to produce an integrated refined view of the initial one.

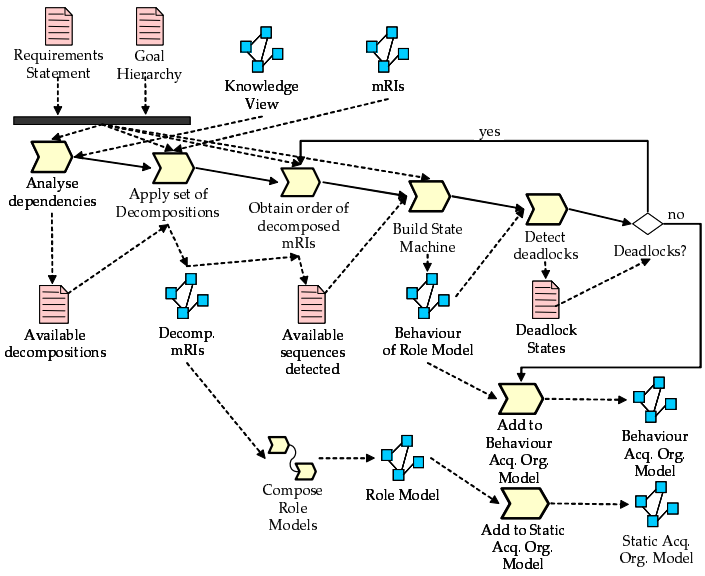


Fig. 14. "Dependency Decomposition" Work Definition

6.2.2 Dependency Decomposition (Depend–Decomp–WD) We can decompose problems by using two techniques: (i) goal decomposition (which has been described in previous stages) and, (ii) dependency analysis decomposition. When we cover all the goals it is possible that we do not reach the enough level of detail to proceed to design. In these situations, we need techniques to continue decomposing functionality into finer grain pieces. This decomposition can be achieved by analysing the dependencies between roles in an mRI grouping such sub–parts of roles that do not depend on others (see Figure 14). We provide three algorithms (from a coarse–grain decomposition to a fine-grain decomposition) which can divide mRIs semi–automatically (we presented the decomposition technique in [13], see it for further details). The main advantages of decomposition are that they usually decrease the number of participant of mRIs and the complexity of the problem to be solved. As a matter of fact, it eases their internal description and helps the transition to next stage. To obtain the *Knowledge Model* we must perform the *Build Knowledge Dependency Model* Work Definition:

This work definition is detailed in Figure 15. It shows us how to build the knowledge view for each mRI identified in the previous work definition.

Finally notice that although, a layer can represent a high level view of the system, this does not mean it is incomplete or inconsistent. Each layer must be consistent and auto–contained. By decomposition or discovering new details, the level of abstraction can be decreased giving rise to a new layer. Notice that each model in a layer may be refined by several models in the next layer. That it is to say, an mRI in a layer is refined by a complete role model in the next layer. This

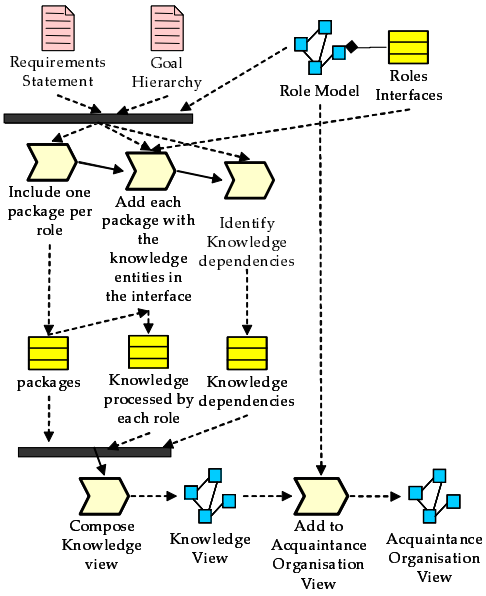


Fig. 15. *Build Knowledge Dependency Model* Work Definition

separation into layers provide a way to dealing with the problem incrementally and in an structured way providing a good understanding of the system and a good way of structuring the documentation which is done by using the mRI traceability view (see Section 5 and *Traceability Maintenance* work definition for further details).

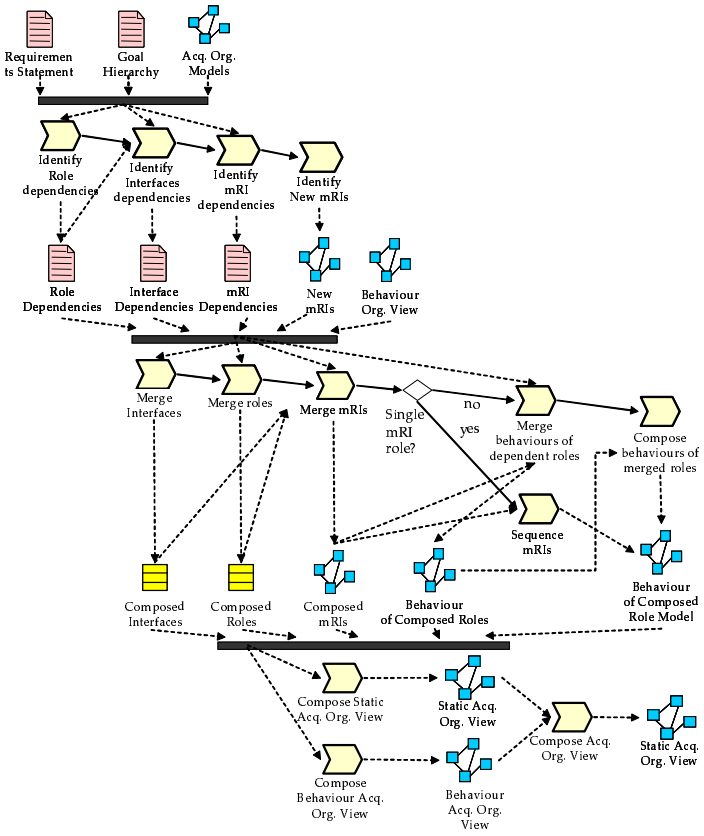


Fig. 16. *Role Model Composition* Work Definition

6.2.3 Composition (Comp–WD) This work definition is depicted in Figure 16. It consists on producing a composed model from several isolated ones. *Goal Decomposition* and *Build Initial Acquaintance Organisation Model* work definitions produce a set of isolated single–mRI Role Models. We can merge them into an integrated model using composition.

Furthermore, goals may depend on others. The most appropriate approach to such situations is to model each goal isolated to later, by means of the composition, merge the role models produced. In order to merge several models, we must identify if there exists dependencies between interfaces, roles and mRIs. Then, we merge such parts that depend. We also have to add such new features which appear in the composition process. That is to say, features that do not belong to the scope of any input model but to all of them.

Finally, notice that composition of models requires merging interfaces, roles, mRIs and behaviour of roles. Interfaces and roles can be merged by a simple union of their elements. The most difficult task is merging the behaviour of roles. We provide several algorithms to assist this task: extraction of the behaviour of

a role from the behaviour of a role model and another for the way back, and aggregation of several behaviours of roles (see Figure 16 for their applicability). We detail these algorithms in [14].

Notice that this operation can be also used to map several role models into a certain set of agents structured as the real word structural organisation in the *Structural Organisation Analysis* stage.

6.2.4 ***Add/Remove Details*** **Work Definition** It consists on adding details identified as a consequence of a deeper understanding of the problem. This is usually performed before decomposition, but it can be also used before a composition.

6.3 *Traceability Maintenance* Work Definition (Traceability-M-WD)

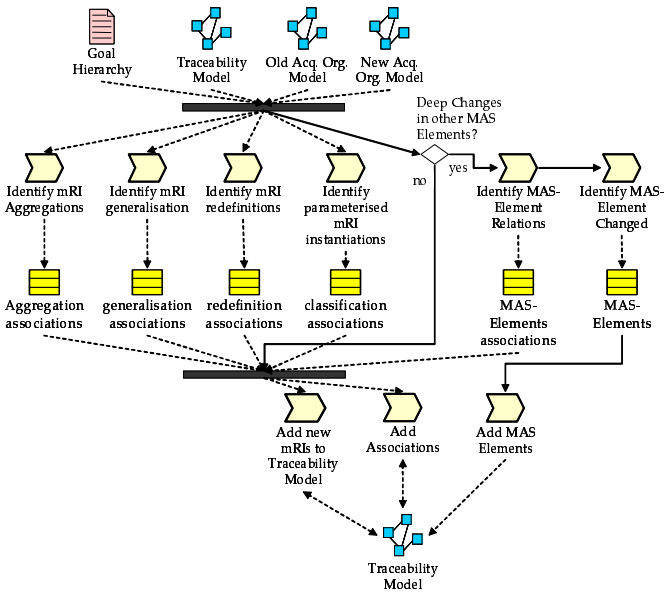


Fig. 17. *Traceability Maintenance* Work Definition

This work definition is depicted in Figure 17. It consists on updating the traceability model for every composition, decomposition or reuse we perform. We must identify all the relationships between mRIs in starting models and mRIs in resultant models. If deep changes are performed over the elements of starting models we must also document them.

6.4 *Reuse* Work Definition (Reuse-WD)

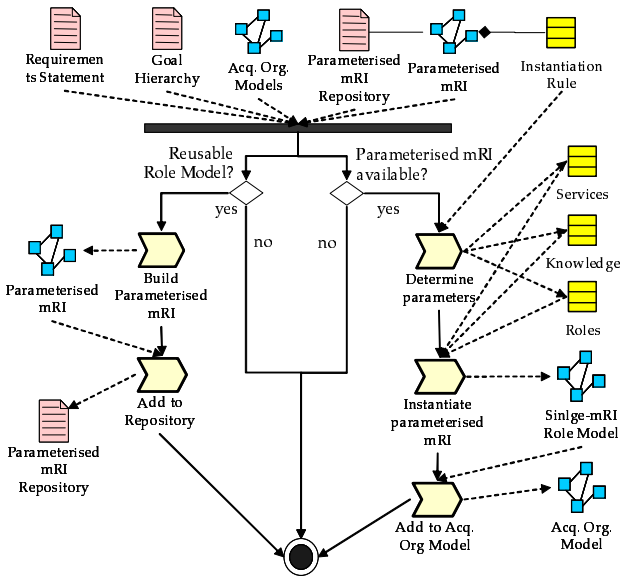


Fig. 18. *Reuse* Work Definition

This work definition is depicted in Figure 17. It consists on instantiating *Parameterised mRIs* from the *Parameterised mRI Repository*. To reuse an mRI we must establish the parameters it has taking into account the constraints imposed by its instantiation rule. We can also create new parameterised mRIs parameterising a role model.

6.5 Integration with other Fragments

In the Metamodel aspect, the main feature of our approach that differs from others is that we use multi-Role Interactions in all models. To integrate them with following fragments, we can directily use them (for example by AUML nested protocols) or describe them internally by, for example, AUML *interaction protocols* or message sequences.

In the process aspect, to integrate our fragment with following fragments we must perform a mapping of the acquaintance organisation we deliver onto an structural organisation obtained by other fragment. Notice that this mapping consists on superposing a set of role models onto a concrete structured group of agents. This mapping can be done by the Composition work definition since it allows to merge several role models superposing their roles and adding the activities needed to assign agents to composed roles. As a matter of fact, before Design, as it is described by most methodologies, we should include a fragment that performs this mapping.

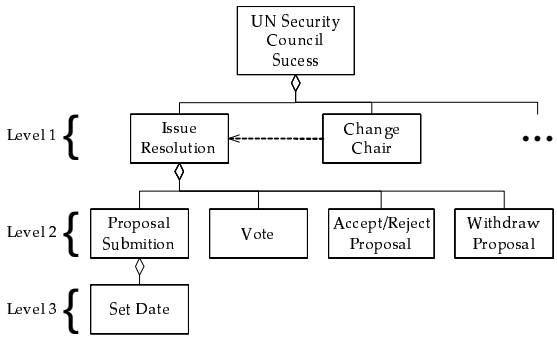


Fig. 19. Case Study Goal Hierarchy

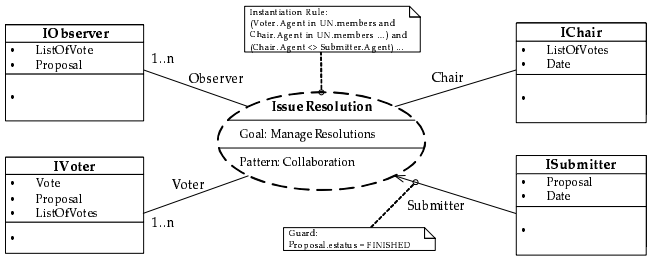


Fig. 20. Layer 1: Role Model *Issue Resolution*

7 MaCMAS/UML Case Study

The case study we use to illustrate our approach is the FIPA Modelling TC UN Security Council's Procedure to Issue Resolutions case study [2]. To pass a UN-SC resolution, the following procedure would be followed: 1) At least one member of UN-SC submits a proposal to the current *Chair*; 2) The *Chair* distributes the proposal to all members of UN-SC and set a date for a vote on the proposal; 3) At a given date that the Chair set, a vote from the members is made; 4) Each member of the security council can vote either FOR or AGAINST or SUSTAIN; 5) The proposal becomes a UN-SC resolution, if the majority of the members voted FOR, and no permanent member voted AGAINST; 6) The members vote one at a time; 7) The vote is open (in other words, when one votes, all the other members know the vote); 8) The proposing member(s) can withdraw the proposal before the vote starts and in that case no vote on the proposal will take place.

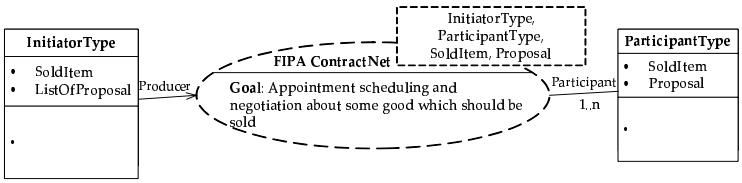


Fig. 21. Parameterised mRI of FIPA ContractNet Protocol

7.1 Layer 1

Figure 19 depicted the goal hierarchy of our case study. In a first approach of the model of the system we can use *Initial-AO–WD*. As a result, we obtain a set of role models for higher level system goals. For example, Figure 20 shows an abstract model obtained by *Initial-AO–WD* which represents the acquaintance organisation formed to fulfil the issue a resolution system goal using a four–party interaction. Several roles participate: *Chair*, *Voter*, *Observer*, and *Submitter*. At the end of CollaborationRoles we place the interface required by each role showing the external features that an agent playing it may expose to the organisation [10, pag. 131]. Notice that UML 2.0 interfaces may contain services and attributes, thus, we can represent the knowledge processed by each role and the services offered.

Although UML 2.0 collaborations do not define any attribute, we have added the goal of the collaboration using a textual description in order to establish a clear traceability with system goals in requirement statements. In Figure 20, this attribute can be observed inside the collaboration icon in a compartment with the name *Goal*. Furthermore, in order to represent the initiator/s of the mRI, we represent them with an arrow from the interface to the collaboration, in our example the role *Submitter*. We have to add also the cardinality of roles showing the number of agents required to play each of them, e.g. the role *observer* can be played by $1..N$ agents.

Each role that participates on an mRI can be decorated with a guard in order to indicate when it is interested on participating in it. Guards are graphically represented as textual notes linked with the association CollaborationRole. For example, the *Submitter* will only participate in the mRI *Issue Resolution* if and only if it has prepared a proposal. The guard could be: $proposal.estatus = FINISHED$. Guards promote the proactivity of agents because they are able to decide whether executing an mRI or not [7]. Furthermore, environment can be modelled by environmental roles represented as a dashed CollaborationRole and a stereotyped interface (application, resource, data–base, etc).Finally, as this interaction pattern could be reused at runtime and have to be mapped into the structural organisation of the UN, we add a textual note to the collaboration to show their instantiation rule indicating that all agents that play roles must be members of the UN.

[2] www.auml.org/auml/documents/UN-Case-Study-030322.doc

As the *Issue Resolution* mRI is similar to the FIPA ContractNet Protocol, we could define it using a parameterised mRI. Unfortunately, it does not fits with our purpose. In Figure 21, we show the parameterised mRI for the FIPA ContractNet Protocol where the type of the *Initiator* and *Producer* roles, and the knowledge that is exchanged are parameters. For example, if the issue resolution process could be fitted with this parameterised mRI, we could assign *Submitter* role to *Initiator* role parameter, *Voter* to *Producer*, *Proposal* to *soldItem* parameter and *Vote* to *Proposal* parameter. As this pattern is well known, it can be attached with a FIPA parameterised protocol description and a code framework that allows us to implement the issue resolution process almost directly.

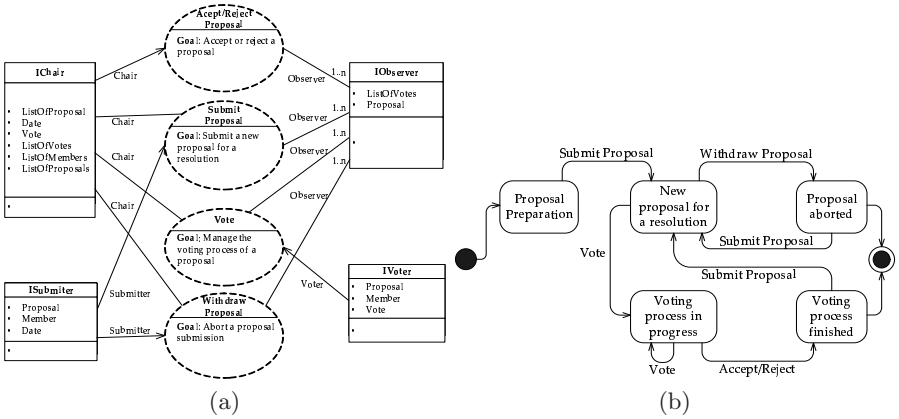


Fig. 22. Layer 2: Role Model *Issue Resolution* obtained by goal decomposition

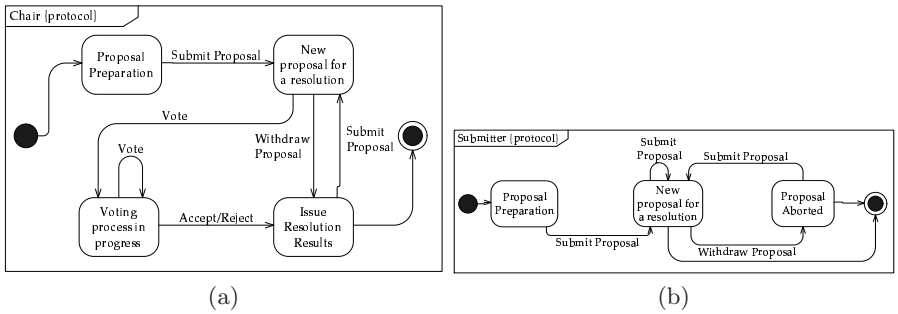


Fig. 23. Layer 2: Protocol State Machines of Roles *Submitter* and *Chair*

7.2 Layer 2

As the goal hierarchical diagram shows us a layered decomposition of the system goals, each mRI model can be placed in its correspondent layer. For example, as the *Vote* mRI is in the level 2 of the goal hierarchy, we can place it in this layer as well as *Proposal Submission*, *Withdraw Proposal*, etcétera.

mRIs identified from system goals are modelled isolated which limit our scope to certain parts of the problem. Them, each mRI in a layer whose goals represents a functional decomposition of a top–layer system goal can be composed by *Comp-WD* to produce a detailed description of the top–layer mRI. For example, we can model all goals in level 2 isolated to later compose them in the role model showed in Figure 22 which corresponds with the *issue resolution* system goal. Figure 22 a) shows the role model where we can see the acquaintance relations by means of mRIs, and Figure 22b) represents the whole role model behaviour in terms of mRIs. Notice that each transition represents an mRI execution which requires all the roles involved to execute it. For example, the transition labelled with the mRI *Accept/Reject* requires that role *Chair* and

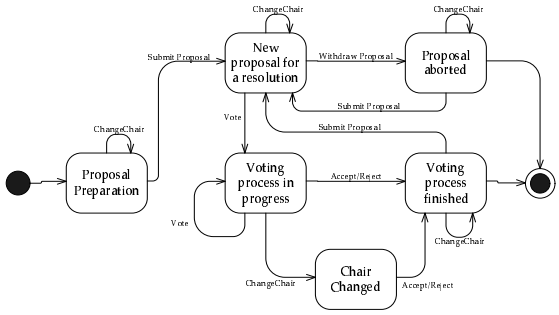


Fig. 24. Layer 2: State Machine of Composed Role Model *Change Chair + Issue Resolution*

Observer execute this mRI jointly. Notice that the guard of both roles for this mRI must hold to traverse the transition. In State Machines mRIs represents events that are produced/consumed by all the roles in it.

The other behaviour representation can be found in Figures 23 (a) and (b) where the behaviour of roles *Chair* and *Submitter* are represented isolated. Notice that in the role behaviour models, all roles protocols execute its transitions coordinately (see a preliminary version of our work in [14] for a detailed semantic of mRI execution). For example, traversing the transition *Submit Proposal* in the protocol of *Submitter*, Figure 22(b), implies to execute only this role part, but notice that as in the previous case, this transition can not be traversed unless the rest of roles of *Submit Proposal*, that is, the *Chair*, are in a state where this mRI is available and the guards of both hold (*Chair* must be in the state *Proposal Preparation* or *Issue Resolution Results*.

We have modelled the *Issue Resolution* goal isolated but it depends on the *Change Chair* role as it is shown in the goal hierarchy. Their composition can be also done by *Comp-WD* and it only implies the addition of a new interaction and some minors changes in the *Chair* interface. For summarising it results, we only show the resultant State Machine in Figure 24.

7.3 Final Layer

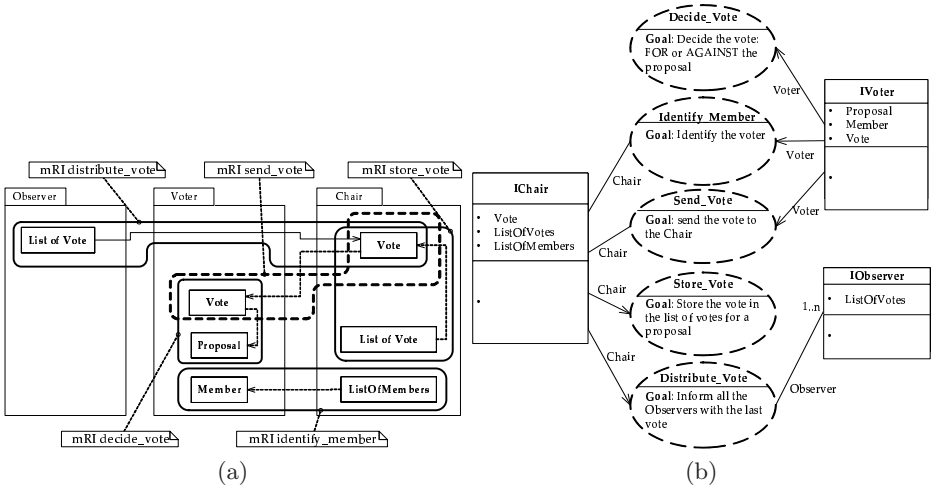


Fig. 25. Final Layer: Role Model *Vote* obtained by dependency decomposition

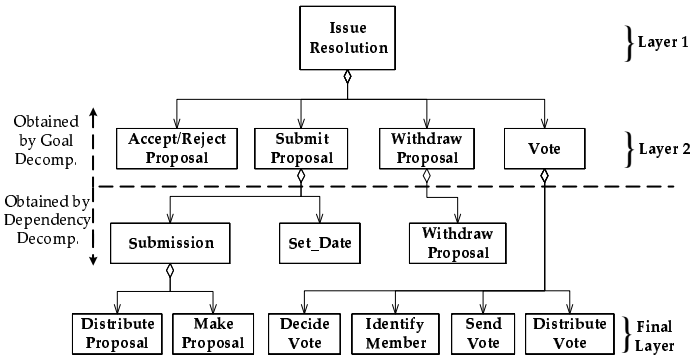


Fig. 26. Final Layer: Refinements traceability model

mRIs corresponding to finer goals identified in previous models are still to much complex to proceed to design. For example, the mRI *Vote* is still to much and we should decompose it further.

With this purpose, we apply the *Depend-Decomp-WD* to this mRI. In Figure 25(a), we show the knowledge view of mRI *Vote* where before applying *Depend-Decomp-WD* using the finer–grain refinement. In this figure, boxes group the knowledge that can be isolated thus producing automatically the role model of

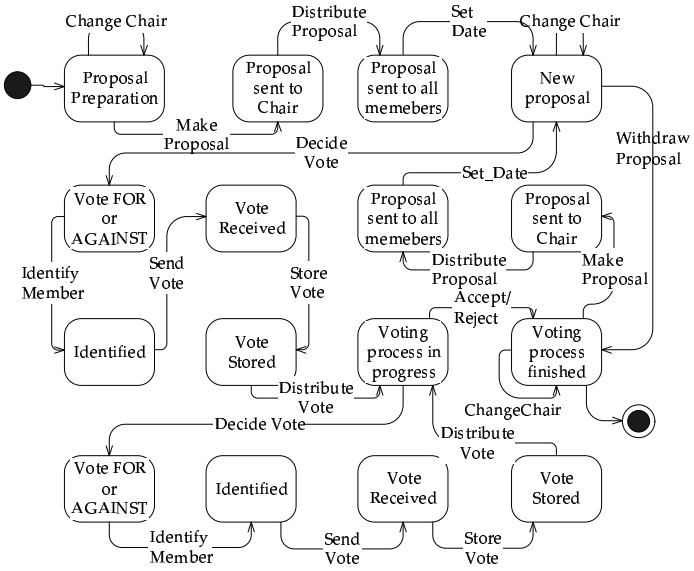


Fig. 27. Final Layer: Whole State Machine

Figure 25(b). Notice that as most of the obtained mRIs requires to share knowledge with the rest, applying the topological–sort algorithm we can infer the state machine of this role model automatically (notice that the mRI *identify_member* do not share knowledge thus it can not be automatically ordered).

Finally, the refined behaviour of *vote* can be substituted, in conjunction with the rest of refinement in Figure **??**a), in *issue resolution* role model behaviour to obtain the state machine of Figure **??**b). The new models obtained by decomposition generate a new layer not identified in requirements which help us to reach the design stage easily.

8 Summary

The main features of MaCMAS/UML are the followings:

Conquer Complexity: By means of abstraction, composition and decomposition principles, we maintain a set of abstraction layers which limit the analyst scope to certain portions of the problem and provide means for face complexity iteratively. Furthermore, we base on several semi–automatic algorithms to compose and decompose models which also conquer complexity automating some tasks of the modelling process. All of them by an UML 2.0 with minor extensions notation.

Traceability: Since we produce a set of models in different abstraction layers traceability is an important feature. We provide traceability from requirements to analysis (each goal has associated a set of models). In Figure 27, we show the final behaviour of role model of the case study. In Figure 26, we show the traceability diagram for goal *Issue Resolution* which shows us how models produced relates each other and with requirements.

Cover the gap between requirement and design: As models are intimately linked with requirement goals we reduce the distance between requirement and analysis. Furthermore, as we add new layers until the level of detail decrease enough to easily proceed to design, we cover the gap analysis–design. In Figure **??**(b), we show the final state machine of our case study in which all mRIs involves only two roles and perform simple calculus. This model can be the starting point of design and its mRIs can be easily described internally by AUML interaction protocols.

Reuse: As a complete role model can be parameterised by an mRI we can maintain a repository of reusable organisation descriptions. Furthermore, notice that problems can be described isolated from the other problems they depend to later compose them. This also improves reusability eliminating relations with other parts of the systems that belongs exclusively to the system we are modelling.

References

1. B. Bauer and J. Müller. Using uml in the context of agent-oriented software engineering: State of the art. In P. Giorgini, J. P. Müller, and J. Odell, editors, *IV International Workshop on Agent-Oriented Software Engineering (AOSE'03)*, volume 2935 of *LNCS*, pages 1–24. Springer–Verlag, 2003.
2. G. Caire, F. Leal, P. Chainho, R. Evans, F. Garijo, J. Gomez, J. Pavon, P. Kearney, J. Stark, and P. Massonet. Agent oriented analysis using MESSAGE/UML. In *Proceedings of Agent-Oriented Software Engineering (AOSE'01)*, pages 101–108, Montreal, 2001.
3. S. Cranefield, S. Haustein, and M. Purvis. Uml-based ontology modelling for software agents. In *Proceedings of the Workshop on Ontologies in Agent Systems, 5th International Conference on Autonomous Agents*, 2001.

4. A. Giret and V. Botti. Towards a recursive agent oriented methodology for large-scale mas. In P. Giorgini, J. P. Müller, and J. Odell, editors, *IV International Workshop on Agent-Oriented Software Engineering (AOSE'03)*, volume 2935 of *LNCS*, pages 25–35. Springer–Verlag, 2003.
5. N. Jennings. An agent-based approach for building complex software systems. *Communications of the ACM*, 44(4):35–41, 2001.
6. N. R. Jennings. Agent-Oriented Software Engineering. In F. J. Garijo and M. Boman, editors, *Proceedings of MAAMAW-99*, volume 1647, pages 1–7. Springer-Verlag: Heidelberg, Germany, 30– 2 1999.
7. J. Koning, M.Huget, J. Wei, and X. Wang. Extended modeling languages for interaction protocol design. In M. Wooldridge, P. Ciancarini, and G. Weiss, editors, *Proceedings of Second Internationa Workshop on Agent-Oriented Software Engineering (AOSE'02)*, LNCS, Montreal, Canada, May, 2001. Springer–Verlag.
8. J. Odell. Agents and complex systems. *Journal of Object Technology*, 1(2):35–45, July-August 2002.
9. J. Odell, H. Parunak, and M. Fleischer. The role of roles in designing effective agent organizations. In A. Garcia, C. Lucenaand, F. Zambonelliand, A. Omiciniand, and J. Castro, editors, *Software Engineering for Large-Scale Multi-Agent Systems*, number 2603 in LNCS, pages 27–28, Berlin, 2003. Springer–Verlag.
10. O. M. G. (OMG). Unified modeling language: Superstructure. version 2.0. Final adopted specification ptc/03–08–02, OMG, August 2003. www.omg.org.
11. H. Parunak, S. Brueckner, M. Fleischer, and J. Odell:. A design taxonomy of multi-agent interactions. In P. Giorgini, J. P. Müller, and J. Odell, editors, *IV International Workshop on Agent-Oriented Software Engineering (AOSE'03)*, volume 2935 of *LNCS*, pages 123–137. Springer–Verlag, 2003.
12. H. V. D. Parunak and J. Odell. Representing social structures in UML. In J. P. Müller, E. Andre, S. Sen, and C. Frasson, editors, *Proceedings of the Fifth International Conference on Autonomous Agents*, pages 100–101, Montreal, Canada, 2001. ACM Press.
13. J. Peña, R. Corchuelo, and J. L. Arjona. A top down approach for mas protocol descriptions. In *ACM Symposium on Applied Computing SAC'03*, pages 45–49, Melbourne, Florida, USA, 2003. ACM Press.
14. J. Peña, R. Corchuelo, and J. L. Arjona. Towards Interaction Protocol Operations for Large Multi-agent Systems. In M. Hinchey, J. Rash, W. Truszkowski, C. Rouff, and D. Gordon-Spears, editors, *Proceedings of FAABS 2002*, volume 2699 of *LNAI*, pages 79–91, MD, USA, 2003. Springer–Verlag.
15. D. Snowden and C. Kurtz. The new dynamics of strategy: Sense-making in a complex and complicated world. *IBM Systems Journal*, 42(3):35–45, 2003.
16. F. Zambonelli, N. Jennings, and M. Wooldridge. Developing multiagent systems: The gaia methodology. *ACM Trans. Softw. Eng. Methodol.*, 12(3):317–370, 2003.