DRAFT v.2.2

REQUEST FOR COMMENTS Guidelines, Techniques and Modelling Artifacts at the Analysis Stage of AOSE Methodologies to Deal With Complexity*

Joaquín Peña and Rafael Corchuelo

The Distributed Group Seville
University of Seville
ETSI Informática
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@lsi.us.es, web page: www.tdg-seville.info

Abstract Complexity is one of the main challenges of the Agent technology. In the Agent Oriented Software Engineering (AOSE) field many authors has developed techniques to palliate this problem. In this request for comments, we propose a a set of guidelines, techniques and modelling artifacts which we think should be used at the analysis stage to conquer complexity. Some of these features are present in some methodologies, has been only identified or even have not been identified.
With this request for comments, we intend to establish some consensus in the elements that AOSE community see as appropriate. With this purpose we have developed an online form to weigh each feature up available at http://www.tdg-seville.info/AOSE-RFC. The purpose of this document is to detail and motivate each of these elements in an abstract and methodology–independent way.

1 Introduction and Motivation

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. [9,11,20].

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:

---

* The work reported in this article was partially supported by the Spanish Ministry of Science and Technology under grant TIC2003- 02737-C02-01, and Castilla–La Mancha Local Government under grant PCB-02-001.

the 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.

In order to clearly distinguish between both views, the former is called *Acquaintance Organisation*, and the later is called *Structural Organisation* [2]. 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.

Complexity[1] is one of the main problems of AOSE. Many authors agree on that the main source of complexity of MASs is consequence their interacting nature of agents, e.g. [6,8]:

> *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. If acquaintance organisation is modelled independently from structural organisation the set of acquaintance relationships can be mapped over the structural organisation (see Section 6).

In [6] Jennings identify the three main principles to manage complexity: abstraction, decomposition and organisation/hierarchy[2] [6]:

Abstraction: It is based on defining simplified models of the systems 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.

Decomposition: is based on the principle "divide and conquer". It helps to limit the designer scope to a portion of the problem.

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 the high–level relationships between several units.

Unfortunately, current AOSE solutions do not cover all these perspectives but subgroups of them.

---

[1] *Complexity* is a vague term that must be further described. See Section 2, Applicability Context, for a detailed discussion.

[2] Notice that hereafter we call it *Composition* in order to differentiate it from the organisation term in AOSE

We think that the modelling process should base on previous principles to properly conquer complexity. As a result of focusing on interactions and principles bellow, we propose a set of features that AOSE methodologies must provide in the analysis stage to conquer complexity. These features are divided into:

Guidelines: Indicate the way in where modelling artifacts and techniques should be used, that is to say, best practices to deal with complexity.
Techniques: Procedures that allows to transform models.
Modelling artifacts: Icons used to graphically represent the system.

The set of features we present in the rest of this document intend to deal with complexity emphasising:

1. Modelling process is focused on interactions and thus on acquaintance organisation without taking into account structural constraints to later, map the former onto the later.
2. The three principles to deal with complexity
3. Guidelines, techniques, and modelling artifact to support a layered specification of the system by means of top-down and bottom–up approaches to obtain layers.
4. Modularity of description
5. Reuse
6. Traceability between requirement and analysis, and between analysis and design.

The final purpose of this document is to establish which of the features identified are relevant to deal with complexity and which are their relevance degree. Later, we are going to study current methodologies along with the results of this request for comments in order to establish quantitatively the degree of complexity coverage current methodologies provide.

This paper is structured as follows: Section 2 shows the features of systems where this work can be applied. Section 3 summarises the structure of the RFC. Section 4 shows features needed regarding the abstraction principle; Section 5 shows features needed regarding the decomposition principle; Section 6 shows features needed regarding the composition principle. And finally, Section 7 is a glossary of terms used in this document.

2 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 [19]. This taxonomy divide 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 the 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 fulfill 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 [12]). The guidelines, techniques and modelling artifacts we propose are also applicable to open system where we know interactions patterns at modelling time but not the concrete agents who participate on them.

3 Structure of the RFC

The request for comments is structured into three sections, one for each principle to deal with complexity. Each of them are also divided into two sub–sections covering the static and the dynamic aspects of modelling:

The acquaintance aspect: which models the relationships between agents in the system from the interaction point of view, e.g. a seller's bank agent is linked with a buyer's bank agent by means of a relationship that represents that they must interact to perform a money transfer. Notice that this differs from the structural organisation models where we represent teams, departments, relations indicating than an agent is subordinate of other, and so on. Hereafter, when we use the term organisation, we refer acquaintance organisation, and not structural organisation.

The behaviour aspect: which models the order of apparition of these relationships over time, e.g. in a online store, first the items to be purchased are chosen (order items relationship) to later purchase these items (order money transfer relationship), *etcetera*.

In Table 1, we show a summary of the features identified. Next sections detail each of the features that will be asked in the online–form. Each feature is marked as sentences in italic–bold font with an icon that show if it is a guideline (G), a technique (T), or a modelling artifact (M).

The online request for comments can be found at www.tdg-seville.info/AOSE-RFC. Each feature have only to be weighted up with two criteria:

- *The agreement level*: it ranges from 0 to 100, where 0 implies that you are totally disagree with the feature, and 100 implies that you are totally agree.
- *The ability to conquering complexity*: it ranges from 0 to 100, where 0 implies that the feature makes no sense to conquer complexity, and 100 implies that it is an essential element to conquer complexity.

4 Abstraction

- G ***A.0.1 Interactions must be abstracted since: agents interact by means of abstract knowledge level relations and they are the main source of complexity. Furthermore, the analysis stage must be focused on interactions***
  Agents interact with others by means of knowledge level relationships. Modelling an implementing such interactions must be done at a high level of abstraction focusing on knowledge exchange [10]. Hence, techniques to model and implements them are needed at the acquaintance and behaviour aspects.
- G ***A.0.2 Maintaining several abstraction layers allows us conquering complexity iteratively. This process should be done by decomposition and composition***

Abstraction	Structural Aspect	☐ A.1.1 Multiparty Interactions
		▒ A.1.2 Internal details of interaction optional
		☐ A.1.3 Abstraction of roles, services, and knowledge of interactions
		▒ A.1.4 Interactions first class modelling element
		☐ A.1.5 Interaction attributes: interaction goal, roles goals, and interaction patterns
		☐ A.1.6 View of knowledge consumed/produced by interactions
		☐ A.1.7 Ontology of an interaction
	Behavioural Aspect	☐ A.2.1 Behaviour descriptions based on multiparty interactions
		☐ A.2.2 (i) Role behaviour model and (ii) acquaintance organisation behaviour model
		▓ A.2.3 Technique to transform the whole behaviour model to a single role behaviour model and vice versa
Decomposition	Techniques to Decompose	▒ D.1.1 Two levels of decomposition: (i) interaction model decomposition, and (ii) interaction decomposition
		▓ D.1.2 Decomposition by Requirement Goals
		▒ D.1.3 Interactions must be linked with requirement goals to provide traceability between requirements and analysis
		▒ D.1.4 Hierarchical goal diagrams guide the decomposition
		▓ D.1.5 Decomposition by dependencies analysis
	Techniques to Support	☐ D.2.1 Roles
		☐ D.2.2 Environmental Roles and Passive Roles
		▓ D.2.3 Technique to extract acquaintance aspect of a role
		▓ D.2.4 Technique to extract behaviour aspect of a role
		▓ D.2.5 Technique to sequence decomposed interactions
		☐ D.2.6 Traceability models
		☐ D.2.7 “Refine” and “composition” associations for interactions
		☐ D.2.8 Roles, services and knowledge “refine” and “composed” associations
		▒ D.2.9 Decomposed models must be modelled without taking into account details on other problems
		☐ D.2.10 Parameterised Interactions
		☐ D.2.11 Instantiation rules of interaction models
		☐ D.2.12 Agents dynamic role playing diagrams
		▒ D.2.13 Instantiation rules allows to model open system
		▒ D.2.14 Distinguish between: analysis organisation and design organisation
Composition	Structural Aspect	▓ C.1.1 Technique to merge interactions and roles
		▒ C.1.2 To step to design, analysis organisation must be mapped onto the design organisation using composition
		▒ C.1.3 To identify interaction models instantiation rules we must compose them
	Behavioural Aspect	▒ C.2.1 Behaviour composition must be done over: (i) behaviour of roles, or (ii) over whole behaviour model
		▓ C.2.2 Techniques to compose the whole behaviour of several interaction models
		▓ C.2.3 Techniques to compose several role behaviour models
		▓ C.2.4 Techniques to compose behaviours: sequential composition, parallel composition and composition by interleaving
		▒ C.2.5 Behaviours composition must be based on multiparty interactions of a certain granularity when role interleaving is needed
		▓ C.2.6 We must provide techniques to isolate the behaviour of a role: (i) to compose it, (ii) to obtain it from a whole behaviour model obtained by whole behaviour composition

☐ Modelling Artifacts ▒ Guideline ▓ Technique

Figure 1. Summary of RFC

Furthermore, maintain several abstraction levels with consistent descriptions and techniques to abstract, refine and modularise them are crucial tools to deal with complexity. Top layers show us abstract models providing an overview of the whole system focusing on more important aspects. Bottom layers give us means for detailing top layers in order to have enough detail to reach a code models of the system. The completion of layers is usually done in an iterative way where abstract layers are refined to produce bottom layers and bottom layers are abstracted to produce top layers. Hence, techniques to perform a top–down and bottom–up approaches are needed to maintain these abstraction layers.

Abstraction mechanisms presented above provide means for maintaining top–layers. Furthermore, techniques to abstract or refine models must be provided which are presented in Sections 5 and 6.

Following, we motivate modelling artifacts, techniques and guidelines to apply abstractions to acquaintance and behavioural aspects.

4.1 Acquaintance Aspect

- M ***A.1.1 Multiparty Interactions:*** *Interactions must be multiparty since it allows us to model high–level social relationships*
  At the acquaintance aspect, the acquaintance organisation of a large MAS may contain a huge number of relationships between agents. Furthermore, these knowledge level relationships usually relates more than two agents. If we model relationships between agents by means of biparty links we are forced to decompose mentally n-party relations where $n > 2$ thus decreasing the level of abstraction from the beginning. For example, a relationship that relates three agents using multiparty interactions can be modelled as a single element, but if we limit it cardinality to biparty, we have to use at least two links decomposing mentally it and entering into the relationships details. This motivates the need for relationships between an arbitrary number of agents in order to provide an abstraction mechanism to deal with complexity.
- G ***A.1.2 Details on how interactions are carried out must be optional***
  Furthermore, these relationships are not accurately known at first modelling stages. For example, at a first approach to a purchase relationship we do not accurately known how is is going to be carried out. If we have to model how it is carried out, we need a deep knowledge of the agents involved what is not possible at analysis stage and counterproductive to maintain an abstract descriptions. This argue for descriptions techniques that do not force us to describe how they are carried.
- M ***A.1.3 Abstract roles, knowledge and services:*** *Abstraction mechanisms must be provided for roles, services and knowledge of interactions*
  Besides, these relationships are one of the main features of agents since a single agent make no sense isolated. Hence, such relationships must be seen

as first class modelling elements that, without detailing how they are carried out, must provide us some information describing their most important abstract features. Defining an interaction involve the definition of a set of agents represented as the role they play on the interaction[3], the services they must provide to the rest of participant to carried it out, and the knowledge that each participant manage. In order to model interaction abstractedly all of these elements must be described from a high level of abstraction:

Role abstractions It consists on a role played by an agent which represents an entire organisations, other system, a legacy systems, and so on. For example, a role banker may represent a complex organisation which when refined may involve several agents. Several role abstraction may be applied, see [11] for a summary of these techniques.

Services Abstractions It consists on abstracting the definition of services provided by agents roles to the group by using pre and post–conditions (Notice that services can be viewed as interactions with an unique performer). For example, banks agents may provide an *update_funds* service to manage the modification of funds which when refined may require a database access database with several queries and updates. It allows to not detail interactions internally.

Knowledge Abstractions: It consists of representing the knowledge owned by each agent in an abstract way at stages where details are not known. For example, banks' agents may own the knowledge about their account which we can represent abstractly as knowledge entity called *account*. *account* represents abstractly all the knowledge that agents must manage about accounts which refined my contain knowledge about the account number, the funds, the credit availability, etc.

– G ***A.1.4 Multiparty interactions must be first class modelling elements***
Interactions must be seen as first class modelling elements since they are the main source of complexity and one of the most important features of agents.

– M ***A.1.5 Interaction attributes: interaction goal, roles goals, and interaction pattern.*** *As first class modelling elements, interaction must own attributes such as the goal of the interaction, the goal of its roles, its participants and the interaction pattern*
As first class modelling elements, interactions must also own attributes: goal of the interaction which model a high level system goal. Participants must be also decorated with the goal they follow in the interaction. Thus, interactions relates system and agents goals (an agent has many goals as roles they play) clarifying the *Congruence* of the system. Interactions must be also attached with the interaction pattern using Co–X terms defined on [12] (cooperation or contention, coordination by conversation, construction, command, constraint,*etcetera*).

---

[3] we refer agents of an interaction as the role they play on it since it promotes decomposition. See Section 5 for a detailed discussion

- ***A.1.6 View of knowledge consumed/produced by interactions*** *Multiparty interactions must be decorated with the knowledge consumed/produced by each role.*

  Since agents interactions take place at the knowledge level, we should describe the knowledge and resources consumed/produced by each role in the interaction. Notice that it provides some information about the interactions internals, but without detailing how the process is carried out.
- ***A.1.7 Ontology of an interaction:*** *The model of knowledge consumed/produced in interactions must be sound, thus ontologies must be attached to interactions*

  These knowledge entities or resources can be modelled abstractedly until we have enough understanding of the system to refine them, however models must be complete at any abstraction layer. Representing the knowledge consumed/produced without any ontology that defines this knowledge is meaningless, thus techniques to do it are needed. The ontology of an interaction although describes the common knowledge that agents must manage to be able to engage in the problem resolution represented by it.

4.2 Behaviour Aspect

- ***A.2.1 Behaviour descriptions based on multiparty interactions:*** *Behaviour description must be able to express how multiparty interactions sequence*

  At the behaviour aspect, we also need mechanisms to represent the way in where abstract interactions (multiparty) evolve over time. That it is to say, their sequence of execution.

  Notice that using messages at this aspect, although they can be abstract, are limited to two participant. This hinder the relation between the acquaintance models of a certain problem and its behaviour model. It also forces analyst to decompose mentally multiparty relations at acquaintance models to a set of binary messages what decrease the level of abstraction and hinders the traceability between models in both aspects.
- ***A.2.2 (i) Role behaviour model and (ii) acquaintance organisation behaviour model:*** *Behaviour descriptions must be modelled from a single role point of view and from the whole organisation point of view*

  Mechanisms to maintain a complete description of the behaviour of a MAS or a sub–organisation and the portion of behaviour belonging to a single role may be also appropriate since it allows us to focus the modelling process on single roles or the MAS/sub–organisation itself.

  Organisation model helps us to understand the behaviour of a group of agents abstractedly (they are based on multiparty interactions). A role behaviour helps us to understand its behaviour in the acquaintance organisation it is engaged.

The role behaviour model can be used when assigning several roles to the same agent since it help us to group such roles which presents a similar behaviour in the same agent.

- T ***A.2.3 Technique to transform the whole behaviour model to a single role behaviour model and viceversa:*** *We must provide mechanisms to transform the whole behaviour model of an acquaintance organisation into a single role model and viceversa*
  Notice that when using multiparty descriptions both behaviour models are equivalent [14]: one is distributed over roles and the other is centralised as the MAS/organisation behaviour. Maintaining both descriptions consistent manually may be a difficult task. Thus, mechanisms to transform automatically one model into another and viceversa may be valuable to conquer complexity by automatisation of this transformation.

5 Decomposition

- G ***D.0.1 Decomposition deals with complexity dividing and conquering problems. It improves reuse and it also help us to obtain bottom layers (less abstract).***
  Abstraction mechanisms may result insufficient when we model a large MAS which faces a complex problem. We can use decomposition to divide complex problems into a set of simpler ones which can be managed easily. This separation requires for a set of techniques to: (i) maintain several separated models and (ii) to determine feasible separations.
  Both techniques provides means for maintaining several layers of abstraction where higher level layers represent abstractedly complex problems and bottom layers store detailed descriptions of sub–part of top layers models obtained by the decomposition techniques above.

Following, we motivate the need for separation and techniques to support and determine it. We also justify how decomposition allows to maintain a layered approach. We discuss first on techniques to determine decomposable parts since depending on the decomposition technique used different techniques must be used to support them.

5.1 Techniques to determine decomposable parts

- G ***D.1.1 There exists two levels of decomposition: (i) interaction model decomposition, and (ii) interaction decomposition. The former can be used to separate models, and the later can be used to refine models in a top–down approach***
  Decomposing a problem into a set of subproblems, may result a hard task if we do not provide techniques for identifying such divisions. Two levels of decomposition exists:

Interaction model decomposition: A decomposition where a complex problem is divided into several separate models without decomposing interactions. For example, a model where several roles interact by means of two interactions: *search_books* and *purchase_books* can be divided into two models: one with the *search_books* interaction and other for the *purchase_books* interaction. Notice that adding details to decomposed problems, where we limit the designer scope, may be more easy than in complete models.

Interaction decomposition: A decomposition where interactions are broken into finer grain interactions. For example, in the previous example we can divide the interaction *search_books* into several refined interactions. For example, *establish_preferences*, *selection_of_dealers*, *get_results* and *eliminate_duplicate_results*.

Without these mechanisms, refinement of multiparty interactions to biparty primitives such as those used at design and code level, e.g, messages, AUML interaction protocols, ACL primitives, and thus, the process to transit from analysis to implementation, may result on intuitive decisions without any guidance.

By decomposing a complex interaction we also divide and conquer complex problems. Decomposition decreases the number of participants of interactions patterns, which lead us to easier implementations (the protocol to coordinate n parties is more difficult than such for two parties) as it is shown in [1,5,13]. Furthermore, the complexity of the knowledge processed in each interaction decreases, thus easing their internal design. Furthermore, multiparty descriptions can be directly executed thus avoiding to describe the synchronization protocol or the communication protocol. There exists implementations languages that provide multiparty interactions primitives that requires the same attributes and elements described for abstracting interactions in in previous section [3,15,17,18,16]. Notice that these are object–oriented implementations which are not specially design for support agent features, but that could be useful to produce prototypes of the system.

The former technique helps us to perform a functional decomposition of the system where we maintain the level of abstraction, and the later helps us to refine models defining abstract interactions internally (by means of finer–grain interactions) thus obtaining a new layer with a refined model.

Following, we discuss on two techniques for determining feasible decompositions:

1. ***D.1.2 Decomposition by Requirement Goals:*** *We can decompose the system by means of system goals or use cases (functional decomposition). It improves system congruence*

   Since agents are limited to some environment and have limited abilities, complex problems are usually solved by a set of agents. For example, a banker agent is only concerned with its bank environment and it is only able to solve problems related with account movements or queries. If a complex goal has to be fulfilled we need several agents to achieve it and hence, at least one interaction between them.

As a matter of fact, a complex goal implies the formation of groups of agents to fulfill it. Identifying complex goals provides means for an initial division where acquaintance sub-organisations in the system are identified. This argue for goal–oriented requirements techniques which guides the initial decomposition of the problem [4,7]. This provides a functional decomposition of the system which may be also performed by analysing requirements document based on use cases.

This process also improves the system *Congruence* since system goals are taken into account and agents goals can be determined by a decomposition of a system goal into a set of sub–goals assigned to agents (notice that such decomposition may have been identified at requirement stage).

D.1.3 Interactions must be linked with requirement goals to provide traceability between requirements and analysis.

Using goal diagrams, we can determine interactions in the system. A goal results on an interaction between several agents when it is enough complex to require more than one agent to be achieved. This argue for linking interactions with goals providing a direct transition between requirements and analysis. Goals that are enough simple can be linked with agents.

D.1.4 Hierarchical goal diagrams guide the decomposition

When a requirement goal is refined by several goals it implies that several subgoals must be fulfilled to fulfill it (by “and” decomposition of goals). This shows us how to decompose an interaction which represent a high–level goal into a set of finer–grain interactions, one per sub–goals to be fulfilled. In “or” decomposition we must fulfill whichever of them.

It can be also used to perform a decomposition where interactions are not decomposed. If we have a model where several sub–goals are represented by means of interactions in a single model, and they are related by ”and” of several high–level goals, each of the high–level goals shows us a set of interactions that can be separated into a decomposed model.

For example, if we have two goals: *search_books* and *purchase_books*, each of them can be refined by and decomposition by several sub–goals. If we deal with a single model which represent the sub–goals of both by means of multiparty interactions, our model can be decomposed into two models, one for the interactions that represent the sub–goals of *search_books*, and another for the sub–goals of *purchase_books*.

2. ***D.1.5 Decomposition by dependencies analysis:*** *When goal decomposition has been exploited, system can be decomposed by analysing the dependencies between knowedge/services required/consumed by each agent* [13]

Furthermore, finer goals identified at requirement stage may result insufficient to reach a level of detail enough to step to design easily. Notice that in complex system, we should minimize the complexity of interactions at analysis as most as possible. If interactions are too much complex at design it will be harder to design them internally.

Thus, techniques to analyse and decompose finer requirement–goals (when they are still too much complex) into simpler unrelated enough ones are needed. This decomposition must group such agents which are highly related. This can be done analysing agents dependencies in a finer–grain interaction which model a sub–goal which is still too complex.

It consists on grouping agents that highly depend in a certain interaction. Since agents relate in order to exchange knowledge and/or problem-solved capabilities, the analysis of interdependencies can be based on analysing services and knowledge dependencies. Each group of dependent agents can be related by a new finer–grain interaction, thus reducing the complexity of models and providing means for identifying refined acquaintance sub–organisations.

For example, in a *purchase* interaction, which is not further detailed at requirement documents, several agents may be involved: a *seller's bank* agent, a *buyer's bank* agent and a *point of sales* agent . *Seller's bank* depends on the knowledge provided by the *point of sales*, *buyer's bank* also depends on the *point of sales*, and *buyer's bank* and *seller's bank* do not depend. Hence, we can decompose the *purchase* interaction into two interactions: *determine_and_pay_goods* between *buyer's bank* and *point of sales*, and *store_profits* between point of sales and *seller's bank*.

5.2 Techniques to support decomposition

– ***D.2.1 Roles:*** *We must use roles to be able to decompose the system at acquaintance and behaviour aspects. Roles are linked with one or more multiparty interactions and describe the knowledge and services offered on these interactions*

  A MAS can be separated into several separate models, each representing how a certain problem is solved, that it is to say, how systems goals are fulfilled. As a result, an agent may be involved in several models. In order to decompose, the activity of an agent may be seen from several points of views, one for each problem resolution it participates. These views are represented by means of the role that the agent plays in each problem.

  An agent has certain behaviour and acquaintance features. All of the features can be grouped by the roles played by it. Notice that a role can be linked with one or more multiparty interactions and must define the knowledge and services that the agent offers to the rest of roles in the interactions it participate.

  Using interaction models (a collection of roles which solve a certain complex problem) every agent participating in the problem resolution is represented as the role it plays. A group of roles that get together in order to solve a problem (perform a system goal) represents an acquaintance sub–organisation. Notice that an organisation derived from the relations of agents in a real organisations does not to be equivalent to organisations formed as a result of acquaintance. For example, the group of teachers of a certain subject are grouped because they teach the same subject, but it does not implies any acquaintance relation between them. See D.2.14 for further details.

- ***D.2.2 Environmental Roles and Passive Roles:*** *Environment must be also modelled by means of roles using "environmental roles" allowing see environment from different isolated points of view. Passive objects can be also modelled as "passive roles" with same advantages*

  All the elements presents in models have not to be agents. There may exists entities with a low degree of proactivity or even totally passive. These entities can be seen as environment resources or passive objects. As such environmental entities may be also used to solve several problems we should also provide different views of them. For example, in a conference management system a paper database can be used for several purposes: in the reviewing problem we are only interested on papers topics, and in the notification process we are interested on authors address and in papers score. Thus, this database can be represented as two different views, one for each sub–problem.

  This separation can be also done by using environmental or passive roles. This also help us to perform an uniform model where a few modelling artifacts has to be managed.

- ***D.2.3 Technique to extract acquaintance aspect of a role:*** *We must provide techniques to extract acquaintance aspect from a given complex model*

  Since the process of engineer a software application is done in an iterative way, we have to deal with descriptions where a set of roles may be involved in the resolution of a certain problem which can be further decomposed. Thus, techniques that allows us to isolate the acquaintance and behaviour aspect of a role regarding a certain problem from a composed model are also quite appropriate.

  At the acquaintance aspect, we need techniques to extract such acquaintance relations (interactions) which are used to solve the sub–problem we are isolating. In addition, if a role is linked with several interactions, we need to separate the part of the interface used by the interactions isolated.

- ***D.2.4 Technique to extract behaviour aspect of a role:*** *We must provide techniques to extract the behaviour aspect of roles from a complex model given a certain decomposition*

  At the behaviour aspect, we need techniques to extract the behaviour of a role regarding the sub–problem we are isolating. That it is to say, extract the order of execution of such interactions in where the role is involved in the problem we are isolating.

- T ***D.2.5 Technique to sequence decomposed interactions:*** *We must provide techniques to sequence a set of decomposed interactions automatically if possible*

  By requirement–goal decomposition we break an abstract interaction into several finer grain interaction. We need techniques to determine how decomposed interactions sequence. In this case, requirement document may help us in this tasks.

  By dependency decomposition, since we know how knowledge and/or services relates, in most cases we can establish automatically how interactions sequence. When an interaction work with the result of other interactions (exists dependencies between their knowledge/services) we can sequence them by analysing their dependency graphs.

- M ***D.2.6 Traceability models:*** *We must provide traceability models in order to justify design decisions and in order to relate layers for both techniques*

  In a top–down approach for the analysis stage, we must transit from requirements to design using decomposition. We can use traceability diagrams in order to clarify this transition. They allow us to justify design decisions more clearly since we can decide between several available decompositions basing the decision on non–functional requirements or the decomposition techniques above and document them in traceability diagrams.

- M ***D.2.7 "Aggregation" associations for interactions:*** *Multiparty interactions can be decomposed, thus we must relate bottom layers interactions with top-layer interactions, that it is to say, top–layers interactions relate with bottom–layers interactions with "aggregation" association*

  If we decompose interactions into several finer grain interactions we obtain a new layer. Relating high-level interactions with interactions resultant of a decomposition provides us traceability from requirements to design. Abstract interactions must be related with the set of interactions that refine it. The association we propose is standard "aggregation".

  For example, we can decompose the interaction *search_books* we can obtain the finer grain interactions :*establish_preferences*, *selection_of_dealers*, *get_results* and *eliminate_duplicate_results*. All of these interactions must be aggregated to form the interaction *search_books*.

- M ***D.2.8 Roles, services and knowledge "aggregation" association:*** *As a result of decomposing an interaction, it roles, services and knowledge must be also decomposed. Hence we must relate these modelling artifacts by means of a "aggregation" relation*

  Notice that when an interaction is decomposed it also implies to decompose it roles, thus roles must be also related by means of a "aggregation" association. Services and knowledge can be also decomposed in finer–grain elements. They should also be related by this association.

- G ***D.2.9 To improve reusability and decrease complexity of decomposed models they must be modelled without taking into account details on other problems***

  It is important to point out that some features are lost when problems are studied isolated. Some interactions may be lost. For example, if a MAS to search and purchase items is decomposed into a search model and a purchase model, the way in how the search is restricted to such dealers that admit the credit card of the user do not belong to the scope of any problem, but both.

  Notice that problems can be modelled keeping in mind constraints with other problems or interactions. However, if we want to reuse a certain acquaintance model it should be as general as possible. If we decompose models isolating them from other models constraint, we produce more general models. This promotes reuse since descriptions do no model dependencies with other problems in the system and can be reused in the same system or others adding specific dependencies. Furthermore, ignoring interdependencies with other models in the system easy the modelling of the problem, thus, decreasing complexity.

- M ***D.2.10 Parameterised Interactions:*** *Agents and knowledge of interactions can be set as parameters to reuse it. Instantiation rules and Dynamic role playing are also crucial elements which must be attached to parameterised interactions to properly reuse them*

  We can parameterised agents and knowledge of interactions to describe a parameterised pattern of interaction which is suitable to be reused.

  As we show bellow, some important features can be lost when we decompose a model such as: (i) instantiation rules (constraint on the identity of agents which can play roles in the parameterised interaction) and constraints on the order in which agents play roles. Both features must be also attached to parameterised pattern of interactions.

- M ***D.2.11 Instantiation rules of interaction models:*** *Interaction models must be decorated with instantiation rules: constraints on the identity of agents who play roles*

  It must be emphasized that constraint on the identity of agents who play each role in a interaction model are also lost in decomposition. For example, using a single model is easy to show that a paper's author can not review its own paper, but when the reviewing process is modelled separated from the submission process this constraint is lost. Instantiation rules help us to model these constraints on how a certain interaction model can be instantiated over a certain group of agents.

  This constraints can be identified and modelled by composition when assigning several interaction models to a set of agents (see Section 6 for further details). Notice that constraints on the identity of agents and their dynamics (see next feature) may apear in the context of a single interaction model or in the context of several interaction models. When several interaction models are involved, composition principle may be used to provide an overall model of both problem where these constraint can be meaningfully described (see Section 6).

- M ***D.2.12 Agents dynamic role playing diagrams:*** *We must provide dynamic role playing diagrams in where we show how roles are played by an agent.*

  Dynamics of role playing are also lost in decomposition. For example, considering an agent who plays roles *book_shooper* and *book_reader*, we can determine that it should play the role *book_shooper* before the role *book_reader*, or even it can play both roles concurrently or interleaved. We need models to show how roles are played. This is further detailed in Section 6.

- G ***D.2.13 Instantiation rules allows to model open system where interaction pattern are known at analysis but not concrete agents who participate on them***

  Instantiation rules for interaction models can be used as we do in Object Oriented paradigm when instantiating an object through a parameterized constructor. This shows a correlation of Classes to Interaction Models and Object to Agent Models. Therefore, this parametrization of interaction models easy the reuse of them and even the instantiation at runtime. It allows to model open systems where interaction models are instantiated dynamically at runtime as we do when creating a new object at runtime in the OO paradigm.

  Besides, constraint on the order of instantiation may appear. For example, the submission process in a conference must be instantiated before the reviewing process. This argue for techniques to show the dynamics of role playing and interaction models instantiation.

- G ***D.2.14 We must distinguish between to architectural levels: analysis organisation and design organisation. Analysis models must be based on roles while design models must be based on agents***

  Separating agents and roles allows us to be able to distinguish between two architectural levels:

  Analysis organisation it is described by means of interaction models. It specify the acquaintance relationships in the system clarifying it functionality by means of interactions.

  Design organisation it represents a concrete instantiation of a set of interaction models over a set of agents (this process is guided by instantiation rules and dynamic role playing models). It specifies the organisation structure based on the real organisation where the system has to be deployed. That it is to say, the design organisation is the mapping of the acquaintance relationships identified at analysis onto a concrete organisational structure.

  Since the main source of complexity are interactions, at analysis we should not bother about organisation structure. This easy the process of understanding the complex behaviour of a MAS. Furthermore, we should not take into account organisation structure at analysis since it may imply to work with implementation issues.

Notice that if we start analysis with a certain organisational structure (by means of agents) based on the real organisation where the MAS is going to be deployed, the initial subdivision in interactions and roles may not be optimal. The main reason is that real organisation are not always modelled from the interaction point of view. For example, the group of teachers of a certain subject are grouped because they teach the same subject, but it does not implies any acquaintance relation between them. Thus, in these situations where the real organisation is not focused on interactions we should start from scratch.

Another risk of starting analysis with the real organisation in mind is that we mimic its mistakes. Thus, when real organisations impose constraints on the system architecture, abstract organisations may be modelled by means of interaction models at analysis to later map them onto concrete agents structured as the real organisation by the composition principle.

6 Composition

- G ***C.0.1 To get a big picture of the system or a sub–part of it we must provide techniques to compose models. That it is to say, composition helps us to maintain top layers. It is also a crucial tool since we can identify instantiation rules, interactions, roles, knowledge, services,*** **etcetera** ***not in the scope of isolated models. Finally, it can be used to map the analysis organisation into the design organisation***

  A complete decomposition of a complex problem into a set of orthogonal sub–problems is not usually possible. When several problems have been isolated in order to study them separately, we are ignoring the interdependencies between them which may contain crucial features of the system. That it is to say, the whole is greater than the sum of its parts. As a matter of fact, new interactions not foreseen when studying problems separately or clearly out of the scope of all the merged problems may appear. New roles, knowledge and services may also appear to support these interactions.

  Furthermore, a set of interaction models which decompose a complex problem do not offers the big picture of it but a tour by the system specification. But designers/implementor must see the most relevant of each subpart of the system. As a matter of fact, we can compose detailed models to abstract them and represent most relevant features in a single more abstract model. It represents the tools needed to perform a bottom–up approach.

  Composing several interaction models (there not exists constraints on the number of interaction models we can merge) implies: (i) to identify new elements, (ii) to merge some of their parts, and (iii), to keep intact others:

  Roles In the composite model two roles of different interaction models may result in a single composite role. For example, composing the paper submission model and the paper reviewing process models the role paper's receiver in the former must be composed with the role paper's dispatcher role in the later. Notice that new roles can be also identified in the composition process.

Interactions Several interactions from different interaction models may result in a composite interaction in the composite interaction model. For example, if we want to get a big picture of a MAS where two interaction models are presented to describe how items are searched and to describe how they are purchased, we can build a composite model where all interactions are merged in a general one for abstraction purposes. Notice that new interactions can be also identified in the composition process.

As a matter of fact, both have to be analysed from the acquaintance and behaviour points of view:

6.1 Acquaintance Aspect

- T ***C.1.1 We need techniques to merge interactions and roles (interface of the roles, links with interactions, goals,* etcetera**
  As roles and interactions can be merged in composite interaction models, we must provide techniques to perform this composition at the acquaintance aspect. For example, when we merge several interactions we have to determine the new initiator, the participants (may be the sum of all of them or not) and so on. When merging several roles, their interfaces must be also merged and their links with interactions.
- G ***C.1.2 To step to design, analysis organisation must be mapped onto the design organisation using composition***
  Interaction models created at analysis must be mapped onto the real world organisation structure to create the initial design organisation. Commonly, as an agent may play several roles, analysis sub–organisations are superposed at design. For example, in the a conference management system, the roles in the organisation for reviewing and the organisation for assigning papers must be mapped onto a set of not disjunct agents. An agent may play the reviewer role, the author role and the assigner role which result in a superposition of several analysis interaction models.
  Hence, in order to step to design the composition operation allows us to create an initial design organisation.
- G ***C.1.3 To identify interaction models instantiation rules we must compose them***
  Furthermore, building a new interaction model from several isolated ones allows us to identify new constraints on the agents who may play each of them. Thus, composition of interaction models is a crucial tool to identify instantiation rules that traverse the frontier of one interaction model as we show in previous section: a paper's reviewer must be different from the paper's author but this instantiation rule does not belong to the reviewing process nor the the submission process, but to both.

6.2 Behavior Aspect

- G ***C.2.1 Behaviour composition must be done over: (i) behaviour of roles, or (ii) over whole behaviour model. If most roles are composed, composition of whole behaviour is recommended. If few roles are composed and most remain unchanged, role behaviour composition is recommended***

  When the behaviour of several interaction models has to be merged we have two alternatives:

  Whole view composition: When most behaviour of roles has to be composed or their behaviour is affected by new interactions we recommend to merge the whole behaviour model. If most roles in the system change, it is easier to understand these changes in a centralised description that in a distributed description over all the roles. For example, if we have an interaction model to airline booking and another to calculate expenses, both models has to be merged to provide information on the prices. Them, we can compose the whole behaviour model of each interaction model to obtain the composite whole behaviour model (see C.2.6).

  Later, if we are interested on the behaviour of one of the composed roles, we can use techniques in point C.2.6 to obtain it.

  Role composition When most roles remain unchanged, it is easier to compose only affected roles since they represent a partial model of the system where we manage only such interactions where affected roles are involved (not all). For example, if we have to compose the *search_book* interaction model with the *purchase_books* interaction model, we have only to compose the role searcher in the former with the role shopper in the later. This will be easier than building a new whole behaviour model since these roles are involved in a sub–group of the interactions that we should manage when dealing with whole behaviour composition.

  Later we can transform the affected roles behaviour along with the roles unchanged to obtain the whole behaviour model of the composite model by techniques in point C.2.6.

- T ***C.2.2 Techniques to compose the whole behaviour of several interaction models:*** *Several interaction models behaviours must be merged to obtain the behaviour of the composite model*

  Techniques to compose the whole behaviour of several interaction models have to be provided in order to determine the behaviour of the composite model. It can be used to obtain the big picture of several problems modelled isolated or to map several interaction models to a certain design structural organisation.

- T ***C.2.3 Techniques to compose several role behaviour models:*** *Several roles that are seen as the same role in a composite interaction model or design organisations must be merged*

  When several roles are composed, their behaviour description (the sequences of interactions they execute) must be also merged. As a matter of fact, techniques to compose the behaviour of several roles that are seen as the same role in a composite interaction model are quite valuable.

Furthermore in design organisations several roles are mapped to the same agent, thus we need a way of merging the behaviour of several roles into a single agent behaviour.

- T ***C.2.4 Techniques to compose behaviours: sequential composition, parallel composition and composition by interleaving***

  We need techniques to compose behaviour descriptions (whole behaviour models or role behaviour models). Several techniques can be identified:

  Sequential The behaviour of each model is executed atomically in sequence which others.

  Parallel The behaviourial of each model is executed in parallel. It is the less common since models that can be executed in parallel are independent and thus, we do not usually compose them.

  Interleaving It is the most common since the composition of several models that are highly related implies to highly interleave them. Notice that the proper interaction granularity must be used for this kind of composition.

- G ***C.2.5 Behaviours composition must be based on multiparty interactions of a certain granularity when role interleaving is needed***

  Notice that if we have to interleave several behaviour descriptions (whole behaviours or role behaviours), we have to do it at the proper interaction granularity. For example, in a computer science department a role professor has to be interleaved with a role head of department. In this situation high level interactions could not be useful. For example, if we model the behaviour of each role by a loop executing a single abstract interaction, *manage_department* for *head* and *teach* for *professor*, we are only able to produce a behaviour description where both roles are alternated. However, if we model both behaviour of roles with finer–grain interactions, we are able to perform a higher interleaved behaviour.

  As a matter of fact, when specifying behaviours, we have to determine the granularity of interactions to be used in the model.

- T ***C.2.6 We must provide techniques to isolate the behaviour of a role: (i) to compose it, (ii) to obtain it from a whole behaviour model obtained by whole behaviour composition***

  As we state in the abstraction section, we must maintain two equivalent behaviour descriptions: a whole description and another composed of a set of single role behaviours.

  If we compose behaviours at the role level, we have to work with the behaviour of affected roles. Thus techniques to obtain them from a whole behaviour description are needed.

  If we perform a composition based on the whole behaviour description of each interaction model, we need techniques to obtain the behaviour of a role before composition. This help us to understand how composition has affected each role behaviour.

7 Glossary

Notice that this is a preliminary glossary. We are currently studying FIPA Methodology TC glossary to re-use its definitions when appropriate.

Acquaintance organisation Organisation from the interaction point of view

Complexity Property of MASs intimately related with the number of interactions in the system

Interaction Model A set of roles related by means of multiparty interactions that jointly fulfill a system goal by contention or cooperation

Interaction granularity Abstraction level of an interaction. Finer grain interactions represent simple interactions, while coarse–grain interactions represent abstractly complex acquaintance relationships

Modelling Artifact Graphical representation of a certain artifact of the system in a model

Modelling Technique Systematic procedure that transform or calculate a piece of a model or a complete new model.

Multiparty interaction Relationship between an arbitrary number of roles established to fulfill some system goal defined abstractly

Role Behaviour Model A model that represents how the set of interactions in where the role is involved are sequenced.

Role Goal A goal that can be fulfilled by a single agent

Role A role is partial view of an agents which represent its features regarding a certain interaction/interactions. A role defines also the interface that the agent offer to the rest of participant by means of services and knowledge

Structural organisation it represents structural relations between agents grouping them in departments, teams, relating them by subordination relationships, and so on

System Goal A goal that requires several agents (more than two) to be achieved

Whole Behaviour Model A model that represents how the set of interactions in an interaction model sequence

References

1. R. Bagrodia. Synchronization of asynchronous processes in CSP. *Transactions on Programming Languages and Systems*, 11(4):585–597, October 1989.
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. R. Corchuelo, J.A. Pérez, and M. Toro. A multiparty coordination aspect language. *ACM Sigplan*, 2000. (To appear in this Journal).
4. A. Dardenne, A. van Lamsweerde, and S.Fickas. Goal-directed requirements acquisition. *Science of Computer Programming*, 20:3–50, 1993.
5. N. Francez and I. R. Forman. *Interacting Processes*. Addison–Wesley, 1996.
6. N. Jennings. An agent-based approach for building complex software systems. *Communications of the ACM*, 44(4):35–41, 2001.

7. E. Kendall, U. Palanivelan, and S. Kalikivayi. Capturing and structuring goals: Analysis patterns. In *Proceedings of the* 3^{rd} *European Conference on Pattern Languages of Programming and Computing*, Germany, July 1998.
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 organisations. In A. Garcia and C. Lucenaand F. Zambonelliand A. Omiciniand J. Castro, editors, *Software Engineering for Large-Scale Multi-Agent Systems*, number 2603 in LNCS, pages 27–28, Berlin, 2003. Springer–Verlag.
10. A. Omicini and A. Ricci. Integrating organisation within a mas coordination infrastructure. In Andrea Omicini, Paolo Petta, and Jeremy Pitt, editors, *Fourth International Workshop Engineering Societies in the Agents World*, number to be publisshed in LNCS, UK, 2003. Springer–Verlag.
11. H. Van Dyke Parunak and James Odell. Representing social structures in UML. In Jörg P. Müller, Elisabeth Andre, Sandip Sen, and Claude Frasson, editors, *Proceedings of the Fifth International Conference on Autonomous Agents*, pages 100–101, Montreal, Canada, 2001. ACM Press.
12. H. V.n D. Parunak, S. Brueckner, M. Fleischer, and J. Odell. A design taxonomy of multi-agent interactions. In Paolo Giorgini, Jörg P. Müller, and James Odell, editors, *IV International Workshop on Agent-Oriented Software Engineering (AOSE'03)*, volume 2935 of *LNCS*, pages 123–137. Springer–Verlag, 2003.
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 the Second International Workshop on Formal Approaches to Agent-Based Systems (FAABS 2002)*, volume 2699 of *LNAI*, pages 79–91, NASA-Goddard Space Flight Center, Greenbelt, MD, USA, 2002. Springer–Verlag.
15. J. A. Pérez, R. Corchuelo, D. Ruiz, and M. Toro. An order–based, distributed algorithm for implementing multiparty interactions. In *Coordination Models and Languages. Proceedings of the 5th International Conference COORDINATION 2002.*, Lecture Notes in Computer Science, pages 250–257, York, United Kingdom, 2002. Springer.
16. J A. Pérez, R. Corchuelo, and M. Toro. An order-based algorithm for implementing multiparty synchronisation. *Concurrency and Computation: Practice & Experience*, to be published, 2004.
17. JA Pérez, R Corchuelo, D Ruiz, and M Toro. A framework for aspect–oriented multiparty coordination. In *DAIS*, volume 198 of *IFIP Conference Proceedings*, pages 161–173. Kluwer, 2001.
18. J.A. Pérez, R. Corchuelo, D. Ruiz, and M. Toro. An enablement detection algorithm for open multiparty interactions. In *Applied Computing 2002. Proceedings of the 2002 ACM Symposium on Applied Computing*, pages 378–384, Madrid, Spain, March 2002. ACM Press.
19. 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.
20. F. Zambonelli, N. Jennings, and M. Wooldridge. Developing multiagent systems: the GAIA methodology. *ACM Transactions on Software Engineering and Methodology*, to be published 2003/2004.