Agent Oriented Analysis using MESSAGE/UML

Giovanni Caire[1], Wim Coulier[2], Francisco Garijo[3], Jorge Gomez[3], Juan Pavon[3], Francisco Leal[4], Paulo Chainho[4], Paul Kearney[5], Jamie Stark[5], Richard Evans[6], Philippe Massonet[7]

[1]Telecom Italia LAB, Via Reiss Romoli 274, 10148 Turin – Italy
giovanni.caire@tilab.com
[2]Belgacom, E. Jacqmainlaan 177, 1210 Brussels, Belgim
wim.coulier@belgacom.be
[3]Telefónica I+D, Emilio Vargas, 28043 Madrid, Spain
fgarijo@tid.es
[4]PT Inovação, Largo de Mompilher, 22 – 3º, 4050-392 Porto, Portugal
fleal@ptinovacao.pt
[5]BtexaCT, Adastral Park, Martlesham Heath, Ipswich IP53RE, UK
paul.3.kearney@bt.com
[6]Broadcom Eireann Research Ltd, Kestrel House, Clanwilliam Place, Dublin 2, Ireland
re@broadcom.ie
[7]CEDITI, Av. Georges Lemaître,21, 6041 Charleroi, Belgium
phm@info.ucl.ac.be

Abstract. This paper presents the MESSAGE/UML agent oriented software engineering methodology and illustrates it on an analysis case study. The methodology covers MAS analysis and design and is intended for use in mainstream software engineering departments. MESSAGE integrates into a coherent AOSE methodology some basic agent related concepts such as organisation, role, goal and task, that have so far been studied in isolation. The MESSAGE notation extends the UML with agent knowledge level concepts, and diagrams with notations for viewing them. The proposed diagrams extend UML class and activity diagrams.

1. INTRODUCTION

1.1 Agent Oriented Software Engineering

The agent-oriented (AO) approach promises the ability to construct flexible systems with complex and sophisticated behaviour by combining highly modular components. The intelligence of these components – the agents – and their capacity for social interaction results in a multi-agent system (MAS) with capabilities beyond those of a simple 'sum' of the agents. The availability of agent-oriented development toolkits has allowed the technology to be assessed for industrial use. Many case studies have been carried out, yielding promising results that have aroused industrial interest in the technology.

Most recent software engineering methodologies are designed for an object-oriented approach. Engineering of commercial MAS requires the availability of agent oriented software engineering (AOSE) methodologies. Most MAS systems will be implemented with object and component based technology in the near future unless a widely accepted agent programming language emerges. In this case, viewed at a detailed level, an agent is a relatively complex object or component. However, this is like considering that a house is a pile of bricks, but it is more convenient to view a house in terms of higher level concepts such as living room, kitchen and bedroom. When an agent is viewed at a more abstract level, structures come into focus that are not found in conventional objects or components. Agent-orientation is thus a paradigm for analysis, design and system organisation. An agent-oriented modelling language must provide primitives for describing these higher-level structures, the inspiration for which derives from cognitive psychology and social modelling via artificial intelligence.

MESSAGE[1] [5] (Methodology for Engineering Systems of Software Agents) is an AOSE methodology which builds upon current software engineering best practices covering analysis and design of MAS which is appropriate for use in mainstream software engineering departments. It has well defined concepts and a notation that is based on UML whenever appropriate.

1.2 Comparison to Other Approaches

Work toward an AOSE methodology can be divided into two broad categories. The first category aims to apply existing software engineering methodologies to AOSE. AgentUML (AUML) [9] for example defines extensions to UML with notations suited for agent concepts. AUML has extended UML's interaction diagrams to handle agent interaction protocols. Although this notation is useful and has been adopted within MESSAGE, it does not have the concept of agent at its centre, i.e. specifying an object's behaviour in terms of interaction protocols does not make it an agent.

The second category of work aims at developing a methodology from agent theory, mainly covering analysis and design. Typically these methodologies define a number of models for both analysis and design [8] such as Gaia [6] and MAS-CommonKads [7]. The Gaia methodology has two analysis models and three design models. While the analysis models are based on well-defined concepts, these only represent a subset of the concepts required for agent oriented analysis. The design models are not clearly explained and the authors envisage OO methods being used for detailed design. Mas-Common-Kads has six models for analysis, and three for design. While these models are comprehensive, the method lacks a unifying semantic framework and notation. In addition to this work, goal analysis techniques have been shown to be very useful [4, 10]. The techniques range from informal to formal analysis and cover functional and non-functional goal analysis. MESSAGE combines the best features of the above approaches.

---

[1] MESSAGE was a two year collaborative project funded by EURESCOM. EURESCOM is a research organization owned by European telecommunications companies, http://www.eurescom.de/.

1.3 Outline and Contributions

The MESSAGE/UML methodology covers MAS analysis and design and is designed for use in mainstream software engineering departments.

This article focuses on analysis of MAS using MESSAGE/UML. Section 2 describes the principal "knowledge level" agent-oriented MESSAGE concepts and describes different views on the analysis model. Section 3 describes the MESSAGE analysis process. Section 4 describes an analysis case study using the MESSAGE/UML notation. The following diagram types are introduced: organisation, goal, task, delegation, workflow, interaction and domain. All are extensions of UML class diagrams, except for the task diagram, which extends the UML activity diagram. The use of schemas to textually describe the concepts is also illustrated.

The contributions of MESSAGE are the agent knowledge level concepts, and the diagrams for viewing these concepts in the analysis model that have been added to UML. MESSAGE integrates into a coherent AOSE methodology that can be used by mainstream software engineering departments some basic agent related concepts such as organisation [11, 15], role [12], goal [4] and task [13], that have so far been studied in isolation. The case-study focuses on illustrating these new agent related concepts and the new diagrams to visualise them. A complete case study can use existing UML notation in addition to the new notation.

2. MESSAGE DESCRIPTION

2.1 Extending UML for Agent Modelling

UML is a convenient starting point for an agent-oriented modelling language for the following reasons:

- UML is widely accepted as a *de facto* standard for object-oriented modelling, many software engineers are trained in its use, and commercial software tools are available to support it (some of which are extendable).
- The object- and agent-oriented paradigms are highly compatible. Agent-oriented concepts can readily be defined in terms of object-oriented ones.
- UML is based on a meta-model (UML uses the MOF meta-modelling language [3]), which makes it extendable[1].

The MESSAGE modelling language is related to UML as follows:

1. It shares a common metamodelling language (meta-metamodel) with UML and MOF
2. It extends the UML metamodel with 'knowledge level' agent-oriented concepts.

A more complete description of the relationship between the MESSAGE metamodel and the UML metamodel is given in [5].

2.2 Main MESSAGE Concepts

2.2.1 Foundations

MESSAGE takes UML as a starting point and adds entity and relationship concepts required for agent-oriented modelling. Agent-oriented modelling borrows from the study of human organisations and societies in describing the way in which agents in a Multi-Agent System work together to achieve a collective purpose, and from artificial intelligence (AI) and cognitive psychology to describe the agents themselves. These additional concepts can be defined in terms of object-oriented ones, but deal with ideas and structures at a higher conceptual level. In AI this higher level is often referred to as "the knowledge level", contrasting knowledge with data. Essentially, MESSAGE uses standard UML as its "data level" modelling language, but provides additional "knowledge level" concepts. These additional concepts are defined in the MESSAGE metamodel [5]. The metamodel also gives a declarative interpretation to some UML concepts used to describe behaviour. The most significant of these is "State" which is described hereafter.

The MESSAGE interpretation of State can be described as follows. A UML model is a collection of objects. A full description of this model at a point in time consists of a description of the value of every attribute of every object. Let us call such description a micro-state. It is rarely practical or useful to work directly with micro-states, however. A State is characterised by a partial description of the model, i.e. a constraint restricting the micro-state of model to being one of a set of possible micro-states. The simplest form of constraint would be to give the value of one attribute of one object in the model. Because States are sets (of micro-states), the language of Boolean algebra can be used to describe their relationships (set union, intersection and containment are equivalent to logical or, and implication).

Note that this is entirely consistent with the UML State concept. From the UML 1.3 specification [1]:

> *A state is an abstract metaclass that models a situation during which some (usually implicit) invariant condition holds. The invariant may represent a static situation such as an object waiting for some external event to occur. However, it can also model dynamic conditions such as the process of performing some activity (i.e., the model element under consideration enters the state when the activity commences and leaves it as soon as the activity is completed).*

The rest of this section describes the knowledge level concepts that feature most prominently in the MESSAGE methodology as it stands at the moment, particularly those that appear explicitly in diagrams.

2.2.2 Knowledge-level concepts

Most of the MESSAGE knowledge level entity concepts fall into the main categories: *ConcreteEntity*, *Activity*, and *MentalStateEntity*. The main types of ConcreteEntity are:

Agent: An Agent is an atomic autonomous entity that is capable of performing some (potentially) useful function. The functional capability is captured as the agent's

services. A *service* is the knowledge level analogue of an object's *operation*. The quality of autonomy means that an agent's actions are not solely dictated by external events or interactions, but also by its own motivation. We capture this motivation in an attribute named *purpose*. The purpose will, for example, influence whether an agent agrees to a request to perform a service and also the way it provides the service. SoftwareAgent and HumanAgent are specialisations of Agent.

Organisation: An Organisation is a group of Agents working together to a common purpose. It is a virtual entity in the sense that the system has no individual computational entity corresponding to an organisation; its services are provided and purpose achieved collectively by its constituent agents. It has structure expressed through *power relationships* (e.g. superior-subordinate relationships) between constituents, and behaviour/co-ordination mechanisms expressed through Interactions between constituents.

Role: The distinction between Role and Agent is analogous to that between Interface and (object) Class: a Role describes the external characteristics of an Agent in a particular context. An Agent may be capable of playing several roles, and multiple Agents may be able to play the same Role. Roles can also be used as indirect references to Agents. This is useful in defining re-usable patterns.

Resource: Resource is used to represent non-autonomous entities such as databases or external programs used by Agents. Standard object-oriented concepts are adequate for modelling Resources.

The main types of Activity are:

Task: A Task is a knowledge-level unit of activity with a single prime performer. A task has a set of pairs of Situations describing pre- and post-conditions. If the Task is performed when a pre-condition is valid, then one can expect the associated post-condition to hold when the Task is completed. Composite Tasks can be expressed in terms of causally linked sub-tasks (which may have different performers from the parent Task). Tasks are StateMachines, so that e.g. UML activity diagrams can be used to show temporal dependencies of sub-tasks.

Interaction and **InteractionProtocol**: The MESSAGE concept of Interaction borrows heavily from the Gaia methodology [6]. An Interaction by definition has more than one participant, and a purpose which the participants collectively must aim to achieve. The purpose typically is to reach a consistent view of some aspect of the problem domain, to agree terms of a service or to exchange to results of one or more services. An InteractionProtocol defines a pattern of Message exchange associated with an Interaction.

The internal architecture of an agent typically is based on one of several models derived from cognitive psychology. MESSAGE is intended to be applicable to a variety of agent cognitive architectures. However, without some basic abstract reference model it is difficult to say anything meaningful. We suppose that the architecture separates an inference mechanism from a knowledge base and a working memory. The knowledge base contains fixed or slowly changing domain or problem-solving knowledge in a declarative form. The working memory contains more transient sensed or derived information. We view this working memory as an abstract database holding instances of MentalStateEntities, and its contents define the Agent's mental state. For present purposes we focus on one type of MentalStateEntity: Goal.

Goal: A Goal associates an Agent with a Situation. If a Goal instance is present in the Agent's working memory, then the Agent intends to bring about the Situation referenced by the Goal. Some Goals are intrinsic to the agent's identity, and are derived from its purpose. These persist throughout the life of the Agent. Others are transient tactical Goals. It is often useful to express the purpose in terms of a utility function that associates 'goodness values' with Situations. The target situation of the Goal is then the one that is estimated to maximise utility (determined dynamically). Note that the agent's knowledge base needs to include 'rules' governing assertion and deletion of (tactical) Goals. One fairly standard rule would be to assert a Goal to provide a given service whenever the Agent agrees with another Agent to do so.

Two other simple but important concepts used in MESSAGE are: **InformationEntity** (an object encapsulating a chunk of information) and **Message**. The agent-oriented concept of Message differs from the object-orient one in a number of respects. In UML, a Message is a causal link in a chain of behaviour, indicating that an Action performed by one object triggers an Action by another object. In MESSAGE, a Message is an object communicated between Agents. Transmission of a Message takes finite time and requires an Action to be performed by the Sender and also the receiver. The attributes of a Message specify the sender, receiver, a speech act (categorising the Message in terms of the intent of the sender) and the content (an InformationEntity).

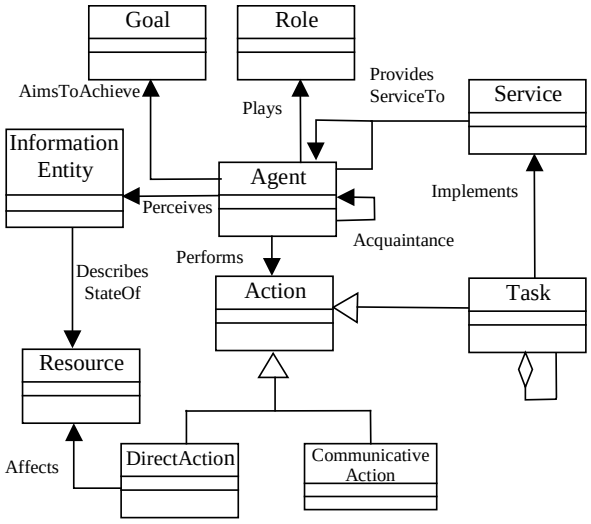

Fig. 1. Agent centric MESSAGE concepts

Figure 1 gives an informal agent-centric overview of how these concepts are inter-related, showing their relationship to the agent concept. A complete description of the MESSAGE metamodel can be found in [5].

2.3 Analysis Model Views

An analysis model is a complex network of inter-related classes and instances derived from concepts defined in the MESSAGE/UML metamodel. MESSAGE defines a number of views that focus on overlapping sub-sets of entity and relationship concepts.

Organisation view (OV) – This shows ConcreteEntities (Agents, Organisations, Roles, Resources) in the system and its environment and coarse-grained relationships between them (aggregation, power, and acquaintance relationships). An acquaintance relationship indicates the existence of at least one Interaction involving the entities concerned.

Goal/Task view (GTV) – This shows Goals, Tasks, Situations and the dependencies among them. Goals and Tasks both have attributes of type Situation, so that they can be linked by logical dependencies to form graphs that show e.g. decomposition of high-level Goals into sub-goals, and how Tasks can be performed to achieve Goals. Graphs showing temporal dependencies can also be drawn, and we have found UML Activity Diagram notation useful here.

Agent/Role view (AV) – This focuses on the individual Agents and Roles. For each agent/role it uses schemata supported by diagrams to its characteristics such as what Goals it is responsible for, what events it needs to sense, what resources it controls, what Tasks it knows how to perform, 'behaviour rules', etc.

Interaction view (IV) – For each interaction among agents/roles, shows the initiator, the collaborators, the motivator (generally a goal the initiator is responsible for), the relevant information supplied/achieved by each participant, the events that trigger the interaction, other relevant effects of the interaction (e.g. an agent becomes responsible for a new goal). Larger chains of interaction across the system (e.g. corresponding to uses cases) can also be considered.

Domain view (DV) – Shows the domain specific concepts and relations that are relevant for the system under development (e.g. for a system dealing with making travel arrangements, this view will show concepts like trip, flight, ticket, hotel….).

Provisional ideas on notation, diagrams and schemas to visualize the views are illustrated in the case study section below.

3. ANALYSIS PROCESS

The purpose of Analysis is to produce a model (or collection of models) of the system to be developed and its environment, that is agreed between the analyst and the customer (and other stakeholders). It aids communication between the development team and the customer, and provides a basis from which design can proceed with confidence. The analysis models are produced by stepwise refinement.

Refinement Approach: The top level of decomposition is referred to as level 0. This initial level is concerned with defining the system to be developed with respect to its stakeholders and environment. The system is viewed as a set of organisations that interact with resources, actors, or other organisations. Actors may be human users

or other existing agents. Subsequent stages of refinement result in the creation of models at level 1, level 2 and so on.

At level 0 the modelling process starts building the Organisation and the Goal/Task views. These views then act as inputs to creating the Agent/Role and the Domain Views. Finally the Interaction view is built using input from the other models. The level 0 model gives an overall view of the system, its environment, and its global functionality. The granularity of level 0 focuses on the identification of entities, and their relationships according to the metamodel. More details about the internal structure and the behaviour of these entities are progressively added in the next levels.

In level 1 the structure and the behaviour of entities such as organisation, agents, tasks, goals domain entities are defined Additional levels might be defined for analysing specific aspects of the system dealing with functional requirements and non functional requirements such as performance, distribution, fault tolerance, security. There must be consistency between subsequent levels. In the MESSAGE project only level 0 and level 1 have been considered.

Analysis Refinement strategies: Several strategies are possible for refining level 0 models. Organisation-centered approaches focus on analysing overall properties such as system structure, the services offered, global tasks and goals, main roles, resources. The agents needed for achieving the goals appear naturally during the refinement process. Then co-operation, possible conflicts and conflict resolution may be analysed.

Agent centred approaches focus on the identification of agents needed for providing the system functionality. The most suitable organisation is identified according to system requirements. Interaction oriented approaches suggest progressive refinement of interaction scenarios which characterise the internal and external behaviour of the organisation and agents. These scenarios are the source for characterising task, goal, messages, protocols and domain entities.

Goal/task decomposition approaches are based on functional decomposition. System roles, goals and tasks are systematically analyzed in order to determine the resolution conditions, problem-solving methods, decomposition and failure treatment. Task preconditions, task structures, task output and task post-condition may determine what Domain Entities are needed. Goals and tasks must be performed by agents playing certain roles. Consequently looking at the overall structure of goal and tasks in the Goal/task view decisions can be made on the most appropriate agents and organisation structure for achieving those goals/tasks.

The experience in MESSAGE shows that the different views of the system leave the analyst free to choose the most appropriate strategy. In practice a combination of refinement strategies with frequent loop-backs among them are used. The analysis process might start with the OV, then switch to the AV and continue with the IV. The results of the analysis of specific interaction scenarios may lead to reconsider part of OV, and starting again refining and adapting OV constituents.

4. MESSAGE/UML CASE STUDY

This section illustrates the MESSAGE/UML concepts and views on a case study. MESSAGE diagrams are introduced with proposed notations. The analysis process is illustrated by describing level 0 and then refining it into level 1.

4.1 Case Study Description

The system under development is a knowledge management system to be used by a team of engineers of a telecom operator company (TOC) that perform equipment installation and maintenance operations on a given territory.

Context: Each engineer in the team gets the list of jobs assigned to him from a co-ordination centre and performs them sequentially moving on the territory in his van. At the end of each job he fills in a proper paper form where he reports the type of problem, if and how the problem was solved. These forms are then sent back to the co-ordination centre where the relevant information is stored in a database. Moreover the TOC owns a database storing all the technical documentation about the equipment deployed in the fields.

Requirements: The TOC wants now to improve the efficiency of the whole process by giving each engineer a proper wireless terminal and developing a system (distributed both on these terminals and on the terrestrial network) that

- Automatically notifies engineers about the jobs they are assigned,
- Automatically and/or on request retrieves the relevant documentation for the job to be carried out,
- Automatically and/or on request identifies other engineers in the team who can provide help in the job to be performed (e.g. because they have proper skills or because they recently solved similar problems) so that it is possible for an engineer to directly receive assistance from another qualified engineer,
- Allows engineers to report about performed jobs filling an electronic form so that the relevant information are directly inserted into the report database.

Appropriateness of an Agent Approach to the case-study: Since the documentation relevant to a job must be proactively provided to the engineer who is going to perform that job, the system to be developed requires its components to show a high degree of autonomy. Moreover it is almost impossible to exactly foresee all possible faults that can happen in the equipment to be maintained and therefore goal oriented behaviour will be needed. Finally finding an engineer with proper skills to provide assistance in a certain job may require some form of negotiation and distributed co-ordination.

4.2 Level 0 Analysis

4.2.1 Organisation view

The analysis starts at level 0 viewing the system to be developed as a black box and focusing on its relationships to the entities in its environment (users, stakeholders, resources, …).

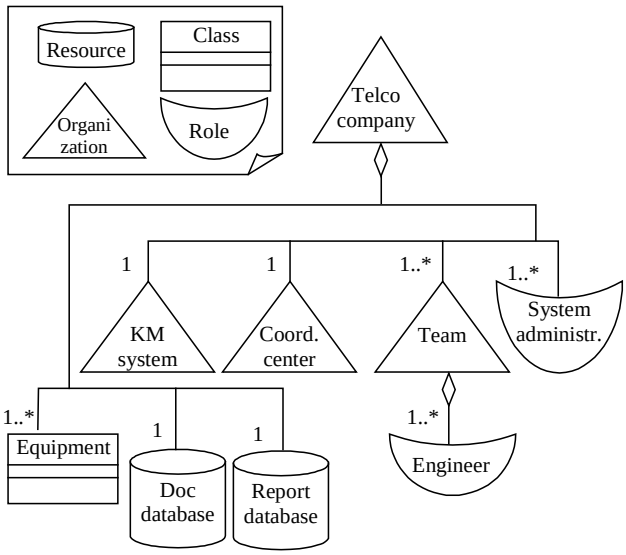

Fig. 2. Level 0 Organisation Diagram (Structural relationships)

Two diagrams from the level 0 organisation view are reported as examples showing the main (from the system point of view) structural and acquaintance relationship in the TOC.

Figure 2 describes structural relationships in a level 0 organisation diagram. The diagram shows that the Knowledge Management (KM) system is owned by the TOC. An Engineer is part of a team and there are several teams in the TOC. It should be noticed that this organisation diagram is a UML class diagram where proper icons have been associated to different stereotypes. At level 0 the system under development, i.e. the KM system, is seen itself as an organisation that will be analysed at level 1.

Figure 3 shows the acquaintance relationships in the level 0 organisation diagram. The KM system interacts with two roles, the System Administrator and the Engineer and with two external systems (resources), the Technical Documentation DB to retrieve documentation and the Report DB to insert the job reports filled by the engineers. Moreover it interacts with the Coordination centre to get the list of jobs to perform. An Engineer also interacts with other Engineers to get direct help. It has to

be noticed that an engineer does not interact directly with the Documentation DB and the Report DB. All these interactions are carried out through the KM system.

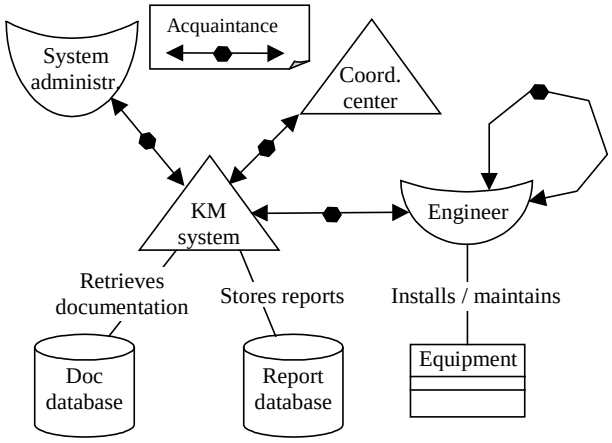

Fig. 3. Level 0 Organisation Diagram (Acquaintance relationships)

4.2.2 Goal/Task view

As for the Goal view the main goal of the system (i.e. providing assistance to the engineers) is and/or decomposed according to the Goal/Task implication diagram in Figure 4.

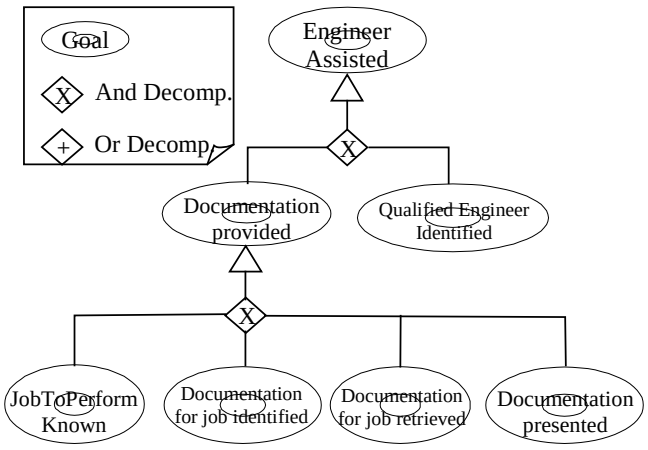

Fig. 4. Level 0 Goal/Task Implication Diagram

The diagram in figure 4 shows that the main goal of the system (EngineerAssisted) is satisfied when the relevant documentation for the current job is provided and the

name of a qualified engineer to possibly request direct help to is identified. The DocumentationProvided goal on its turn is satisfied when the job to be performed is known, the documentation required to perform the job has been identified/retrieved, and that documentation is presented to the assisted engineer. The decomposition of the QualifiedEngineerIdentified goal is not shown. Alternative decompositions can be modelled with or-decomposition notation not illustrated here.

Alternatively, or in conjunction with goal/task implication diagram it is useful to analyse how a given service is realised by a partially ordered set of tasks. The example in figure 5 shows the workflow of tasks implementing the Identify-Qualified-Engineer service by means of a workflow diagram (i.e. a UML Activity Diagram where tasks are shown instead of activities). The diagram also shows the classes that are input/output of tasks using object flows and the roles that perform the tasks. This diagram is similar to the agent head automata, which is an extended state automata, proposed in [14].

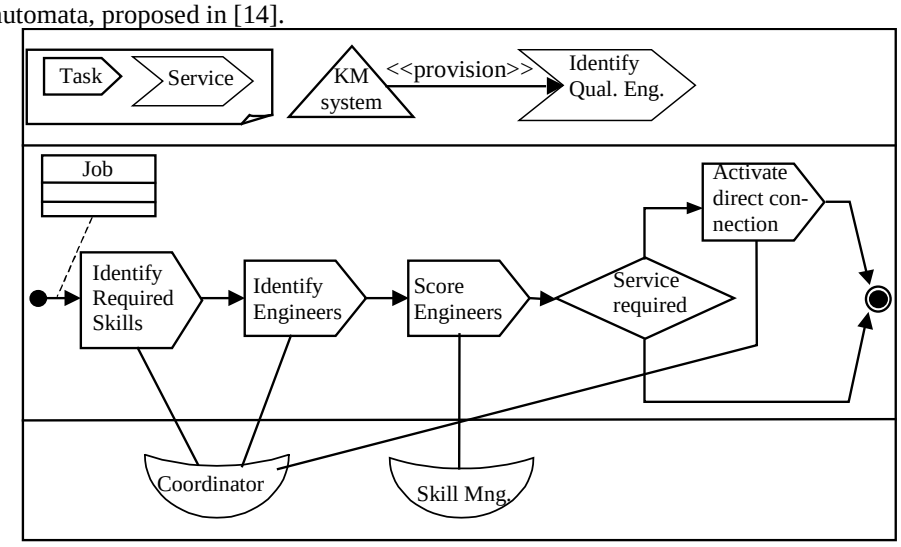

Fig. 5. Level 0 Workflow Diagram

4.3 Level 1 Analysis

4.3.1 Organisation view

Moving from level 0 to level 1, analysis focuses on the system itself identifying at a glance the main pieces of functionality required (seen as roles and/or types of agents). The approach followed in this simple case study is to consider only roles initially and to define what agents will populate the system and what roles each agent will play at the beginning of the design process. However the developer is free to start identifying agents during analysis.

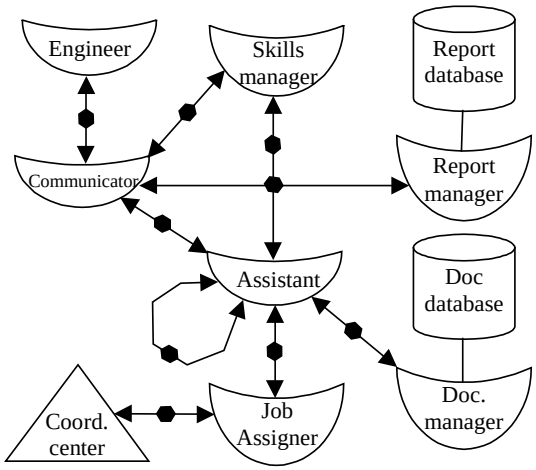

Fig. 6. Level 1 Organisation Diagram (Acquaintance relationships)

Figure 6 shows the level 1 acquaintance relationships in an organisation diagram. The skills manager maintains knowledge of engineer's skills on the basis of the jobs he carries out. The interaction between Assistants requires a contract-net to identify another engineer who has the right skills to provide assistance for a given job.

4.3.2 Agent/Role view

Delegation, Workflow structure diagrams, and textual Agent/Role schemas are useful to describe the view.

A delegation structure diagram shows how the sub-goals obtained decomposing a goal of an organisation are assigned to the agents/roles included in the organisation. Clearly this diagram is strictly related to (and must be consistent with) both the goal decomposition diagram showing the decomposition of the organisation goal and the organisation diagram showing the agents/roles inside the organisation.

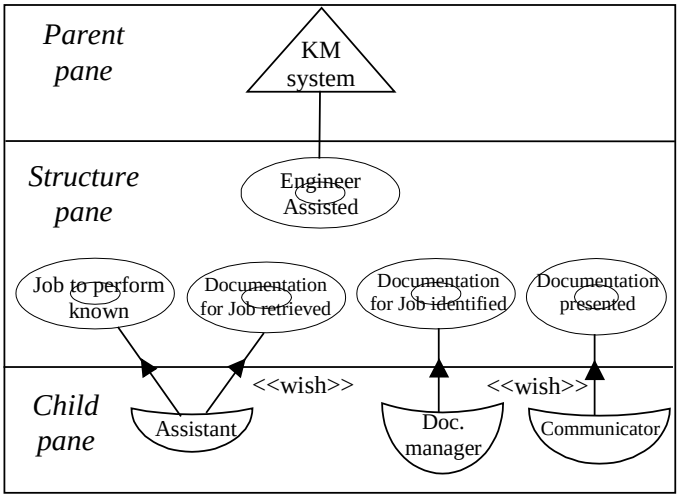

Fig. 7. Level 1 Delegation Structure Diagram

Figure 7 shows a Delegation structure diagram. Only the root and the leaves of the decomposition of the parent organisation goal are shown.

Similarly a workflow structure diagram shows the roles in an organisation that must perform the tasks necessary to implement a given service provided by the organisation.

For each agent/role there is one Agent/Role schema that describes its characteristics. At the analysis level this information is typically quite informal and therefore free text is preferred to a graphical notation. The schema below describes the Assistant role.

Table 1. Role schema

Role Schema	Assistant
Goals	JobToPerformKnown, DocumentationForJobRetrieved
Capability	Some learning capability is required to keep the profile of the engineer updated on the basis of the completed job.
Knowledge, Beliefs	A profile of the skills of the engineer to be used to evaluate if and how the engineer can provide help to a colleague requesting assistance A profile
Agent requirements	This role will be played by the agent that actually assists the Engineer.

4.3.3 Interaction view

This view highlights which, why and when agents/roles need to communicate leaving all the details about how the communication takes place to the design process.

The interaction view is typically refined through several iterations as long as new interactions are discovered. It can be conveniently expressed by means of a number of interaction diagrams. These diagrams are interaction centric (i.e. there is one of such diagram for each interaction) and show the initiator, the responders, the motivator (often a goal of the initiator) of an interaction plus other optional information such as the trigger condition and the information achieved and supplied by each participant.

The following picture shows as an example the interaction diagram describing the Documentation Request interaction between the Assistant and the Documentation Manager roles. Figure 8 shows an Interaction diagram.

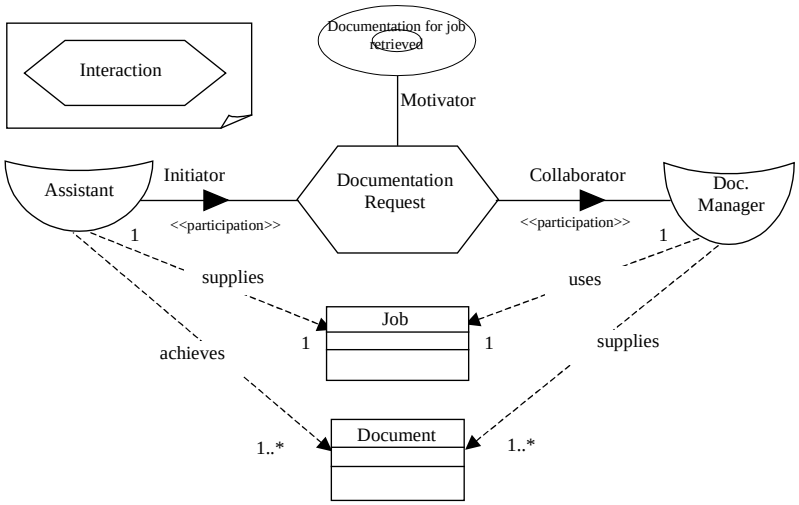

Fig. 8. Level 1 Interaction Diagram

The details of the interaction protocol and the messages that are exchanged between roles can be represented using AUML sequence diagram [2].

4.3.4 Domain view

The domain view can be conveniently represented by means of typical UML class diagrams where classes represent domain specific concepts and named association represent domain specific relations. It is typically built in parallel to the other views by adding new concepts and relations as long as they are needed in the other views. Figure 9 shows a provides a very simplified example related to the considered case study.

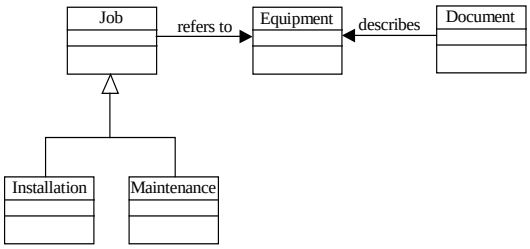

Fig. 9. Domain Information Diagram

5. CONCLUSIONS

This paper has presented the MESSAGE/UML AOSE methodology and illustrated it on an analysis case study. MESSAGE extends UML by contributing agent knowledge level concepts, and diagrams with notations for viewing them. The diagrams extend UML class and activity diagrams. The methodology covers MAS analysis and design and is designed for use in mainstream software engineering departments.

Section 2 described the principal "knowledge level" agent-oriented MESSAGE concepts and described how a MESSAGE specification is organised in terms of an analysis model and views. The following overlapping views have been defined on the analysis model: Organisation, Goal/Task, Agent/Role, Interaction and Domain. Section 3 described the MESSAGE refinement based analysis process. Section 4 described an analysis case study using the MESSAGE/UML notation. The following diagrams were illustrated: organisation, goal/task implication, workflow, delegation, interaction and domain information. The use of schemas to textually describe the concepts was also illustrated. A more complete analysis model completes the MESSAGE diagrams with existing UML notation and AUML sequence diagrams to describe role/agent interactions.

ACKNOWLEDGMENTS

The authors would like to thank EURESCOM for the project support, all P907 project contributors, and the AOSE reviewers for their useful comments.

REFERENCES

1. OMG Unified Modeling Language Specification Version 1.3. Object Management Group, Inc., http://www.rational.com/uml/resources/documentation/index.jtmpl, June 1999.

2. Bauer, B. et al. Response to the OMG Analysis and Design Task Force UML 2.0 Request for Information: Extending UML for the specification of Agent Interaction Protocols. ftp://ftp.omg.org/pub/docs/ad/99-12-03.pdf .OMG, December 1999.
3. OMG Meta Object Facility (MOF) Specification. ftp://ftp.omg.org/pub/docs/ad/99-09-04.pdf., September 1999.
4. Dardenne, A., van Lamsweerde, A. and Fickas, S. Goal-Directed Requirements Acquisition. Science of Computer Programming Vol. 20, North Holland, 1993, 3-50.
5. MESSAGE website, http://www.eurescom.de/Public/Projects/p900-series/P907/P907.htm
6. Wooldridge, M., Jennings, N.R., Kinny D. "The Gaia Methodology for Agent-Oriented Analysis and Design". Kluwer Academic Press, 2000.
7. Iglesias, C., Garijo M., Gonzalez, J. and Velasco, J.R. Analysis and Design of multiagent systems using MAS-CommonKADS. Intelligent Agents IV: Agent Theories, Architectures and Languages, 1997, Singh, M. P., Rao, A. and Wooldridge, M.J., eds., Lecture Notes in Computer Science 1365.
8. Iglesias, C., Garrijo, M., Gonzalez, J. A survey of agent-oriented methodologies. Agent Theories, Architectures and Languages, 1998.
9. Odell, J., Van Dyke Parunak, H., Bauer, B. Extending UML for Agents. Proc. Of the Agent-Oriented Information Systems Workshop at the 17 th National Conference on Artificial Intelligence, Wagner, G., Lesperance, Y., and Yu, E. eds. 2000.
10. Mylopoulos, J., Chung, L., Liao, S., Huaiqing Wang, Yu, E. Exploring alternatives during requirements analysis. IEEE Software, Vol. 18, N. 1, 2001, 92 –96.
11. Zambonelli, F., Jennings, N.R., Wooldridge M. Organisational Abstractions for the Analysis and Design of Multi-agent Systems. In P. Ciancarini, M.J. Wooldridge, Agent-Oriented Software Engineering, vol. 1957 LNCS, 235-251. Springer-Verlag: Berlin, Germany 2001.
12. Kendall, E.A. Agent Software Engineering with Role Modelling. In P. Ciancarini, M.J. Wooldridge, Agent-Oriented Software Engineering, vol. 1957 LNCS, 163-169. Springer-Verlag: Berlin, Germany 2000.
13. Omicini, A. SODA: Societies and Infrastructures in the Analysis and Design of Agent-Based Systems. In P. Ciancarini, M.J. Wooldridge, Agent-Oriented Software Engineering, vol. 1957 LNCS, 185-193. Springer-Verlag: Berlin, Germany 2000.
14. Bauer, B. UML Class diagrams Revisited in the Context of Agent-Based Systems. In this volume.
15. Van Dyke Parunak, H., Odell, J. Representing Social Structures in UML. In this volume.