

Utilizing Dynamic Roles for Agents

Conor Brendan Ward, University of Technology, Sydney
Brian Henderson-Sellers, University of Technology, Sydney

Abstract

The development of Agent Oriented Software Engineering (AOSE) and the use of roles within AOSE have been suggested as an important enabling feature in the future development of robust software systems. This paper seeks to identify and develop the definition of roles within an existing agent-oriented modelling language, namely FAML. The paper discusses the importance of role reuse, the process of role adoption and the advantages of role adaptation, and then proposes a framework for role reuse and adaptation within the FAML framework.

1 INTRODUCTION

A general consensus in the agent-oriented research community is emerging that roles appear to be an important feature of Multi Agent Systems [1-4]. An agent role is a characterisation of a set of normative or typical behaviours that an agent, within a particular system, may perform [5]. These provide a subset of an agent's responsibilities and are dynamic i.e. any particular role can be temporary and/or an agent may be playing more than one role at any given moment in time. Odell *et al.* [6] draw strong parallels to thespian roles i.e. taking the idea of a role as being like the script used by an actor, informing their behaviour and providing a set of guidelines of acceptable behaviour, but in which each player applies their own interpretation. All of the players act in concert to achieve an overall effect or Goal, but each of these players has a different part to play [6,7]. This leads to the notion of roles being assigned to agents operating within organizational groups (Figure 1).

It should be noted that one feature that a role does not have is a state. A state is a concrete characteristic or attribute, whereas a role is an abstract characteristic of the agent that is playing the role. A role informs an agent what it should do; it does not complete the action for the agent. A real life example of this is a manager role; an agent needs to play the role of a manager, the role doesn't exist without the agent. This being the case, it must also be said that a role can add to or modify the state of an agent; for example, the agent's state will be modified when taking on the role saying, "I am now playing the role of a manager".

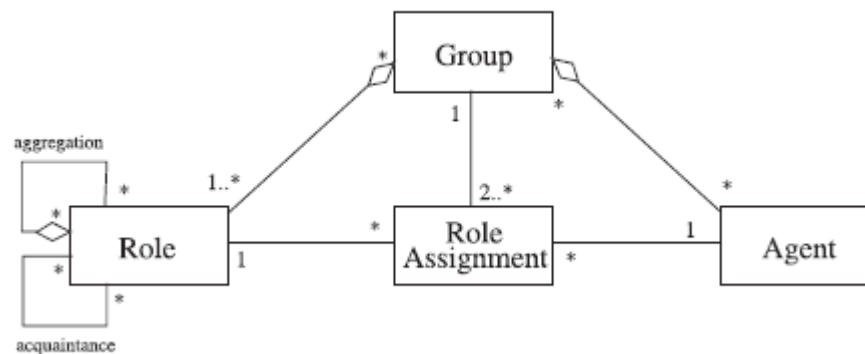


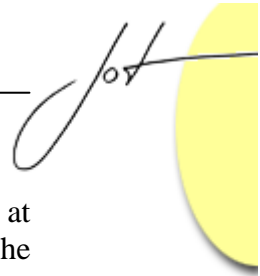
Figure 1. Agent-group-role class model with role assignment (after [6] With kind permission of Springer Science+Business Media)

Furthermore, roles are especially important when trying to characterise the interaction between multiple agents. The definition of a role can describe behaviours that are dependent on the cooperation of other roles. This ability to describe interactions provides the system designers with a new method of formalising these types of interactions that a community of agents can perform without being overly prescriptive. A role definition imposed, at design time, by software developers (exogenous) is contrasted with self-emergence of roles (endogenous) in [6], recommending that the former are more likely to gain acceptance by end-users since they perceive that a reliance on emergent behaviour is too risky.

Restricting our discussion here to the exogenous definition of roles, and noting their dynamic characteristics (see also [2], who raise concerns about support for dynamicity in AUML: [8]), leads to our proposal that role definitions can be specified at design time (and later adopted dynamically at runtime by the various agents in the MAS) as if they were method fragments e.g. [9-16]. This means that it is reasonable to include in our proposal the design of a repository-like facility that provides agents within a community with a set of roles that can be adopted within that community. The idea of a framework, or repository of roles (a.k.a. “Role Library:”), is a concept that has been implied by some of the literature [17], but no explicit references to this type of construct have been found within the AOSE community.

The Role Library will inform agents within a community, or an agent that joins the community at a later point in time, about which behaviours are permitted in the community, as well as how they are allowed to modify these behaviours. The Role Library also establishes a list of acceptable roles as well as the characteristics of such roles. This permits the use of roles in the community that have not yet been envisaged at the time that the system was designed. This ability to use previously unknown roles should facilitate their evolution, thus increasing the adaptability and robustness of the Multi Agent System.

The only mention of this type of facility that has been found in the literature previously [17,18] is the work on the Behavioural Role Agent Interactions (BRAIN) Framework (see also [19]) that discusses the possibility of forming an XML-based



framework to help manage agent interactions. The BRAIN Framework is aimed at creating a common XML-based interface for agents in Open Distributed Systems. The purpose of this framework is to enhance the ability of agents to interact with other agents that have not previously been party to an interaction by providing a common interface for that interaction [18]. In contrast, we seek here to achieve the ability to build a feature into the agents' community to facilitate the advertisement and exchange of previously defined roles. This concept is much like that used within the arena of web services. The aim of the reuse framework that will be investigated here will be to increase the ability of an agent to find and adopt previously defined roles. This type of facility will increase the flexibility and robustness of a Multi Agent System, by allowing agents greater access to roles allowed within a system or agent community. Finally, we seek a formal underpinning in terms of a metamodel and investigate some of the possible links to agent modelling languages.

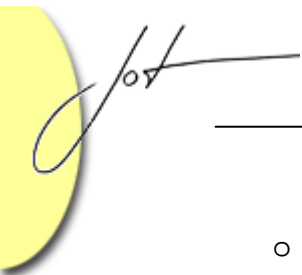
In the remainder of this paper, Section 2 summarizes the current metamodel of the FAME¹ Agent Modelling Language, FAML [20], focussing upon the extent of its current support for roles. In Section 3, we extend the FAML role model in terms of additional administrative features: responsibilities and necessities. Section 4 describes the Role Reuse Framework as a repository for dynamic role descriptors, while Section 5 describes the adoption process possibilities for these roles. Following a discussion (Section 6), we highlight limitations of this work, together with further ensuing research that becomes possible (Section 7) before concluding in Section 8.

2 THE FAML METAMODEL FOR ROLES

The existing FAME metamodel describes a role as “a specification of a goal directed behavioural pattern expected from some Agents in a given system”. The Role class within the metamodel (Figure 2) is described in terms of its Attributes and Associations, namely:

- Attributes
 - Name string Name of the Role.
 - Responsibilities string Responsibilities associated with this Role.
- Associations:
 - BelongsTo Belongs to a particular system.
 - CanUse Can use a number of message schemata.
 - CollaboratesIn Can be collaborator in a number of tasks.
 - Implies Implies a Role specific Agent definition.
 - IsDestinationIn May be destination in a Role Relationship.

¹ FAME (Framework for Agent-oriented Method Engineering) is the project name under which FAML has been developed.



- IsPlayedBy May be played by a number of Agents.
- IsResponsibleFor Is responsible for a number of tasks.
- IsSourceIn Source May be source in a Role Relationship.
- IsSpecialisedInto May be divided into sub type Roles.
- SpecialisesFrom May specialise a particular super type Role.
- Uses May use a number of ontologies.

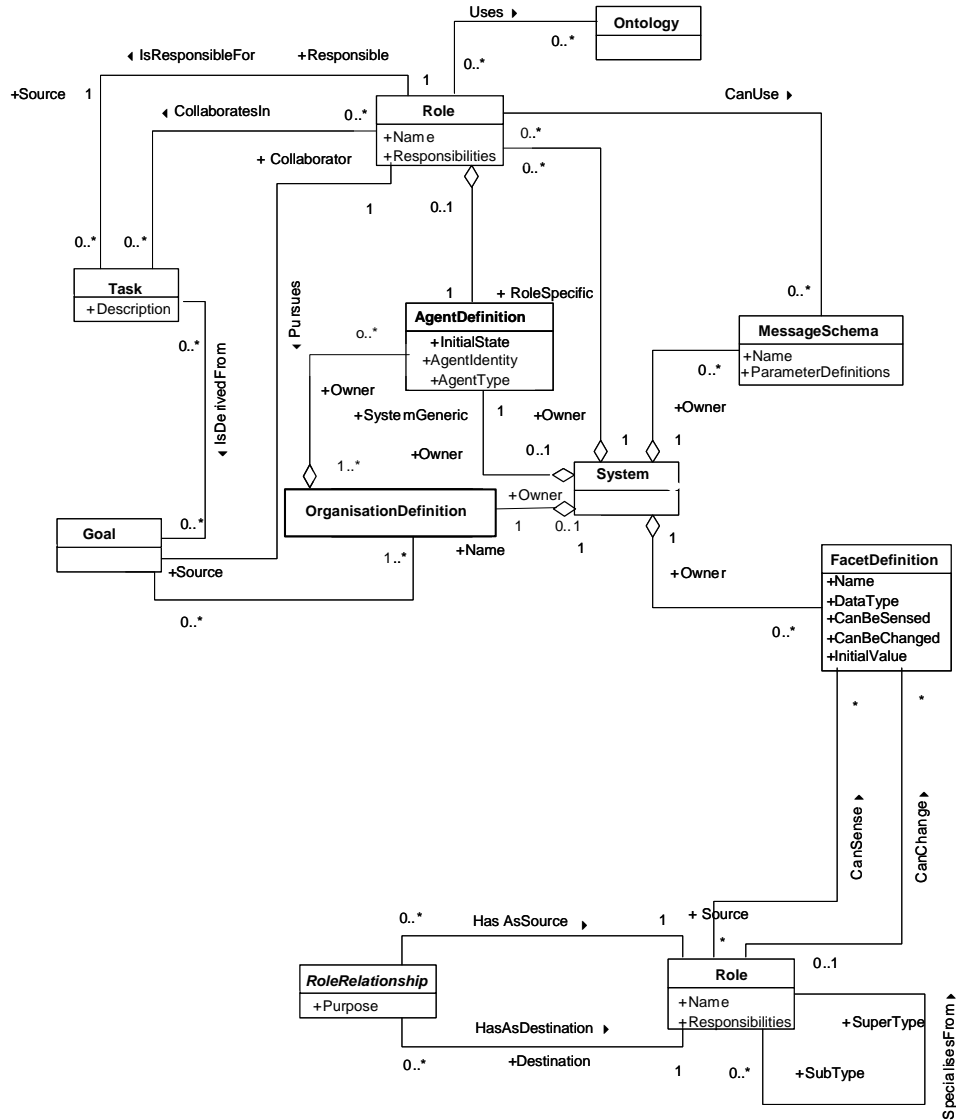
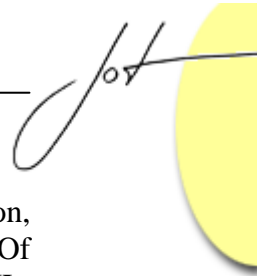


Figure 2. Some of the most relevant design time, system-level classes in FAML. (The Role class is repeated merely to avoid awkward layout issues such as line crossings)

The aggregation relationship of Figure 1 is not currently supported. The Aggregation / Composition is only accounted for implicitly by the use of the Role Relationship feature. It would be advantageous if the ability to construct Roles, by the addition of other Roles,



was to be enhanced by the provision of a Role Aggregation relationship. For this reason, this paper suggests the addition to the FAML metamodel of a recursive Composed-Of association on Role to model role aggregation. Finally, sub-roles are modelled in FAML, as are role relationships (acquaintance in Figure 1). FAML has not yet considered the ability of a number of Agents to hold a Role of the same Role Type, such that a Group Role may be implemented. This would allow the repeated use of the same structures, and possibly the use of a special “member of Group or Community” Role. The benefit of this Group Role (Figure 1) would be to maintain a consistent way of implementing social structures in the agent communities, and is based on the agents of each of these groups sharing a particular Goal. This approach would also allow for the use of a particular type of recursion in the decomposition of these shared goals amongst the members of the groups of agents.

At design time, roles are stressed over agents, the AgentDefinition class being a repository of initialization data for the agent. The Role has a number of Responsibilities, which will be discussed in detail in this paper – Figures 1 and 2 are both at a high level of abstraction and provide no such detail.



Figure 3. The Agent-Role relationship that is at the heart of the agent-level classes in FAML, as described more fully in [20]

At runtime, FAML Agents are directly linked to Roles defined in the design. The many (*) multiplicity in Figure 3 expresses succinctly the dynamicity of roles as discussed by e.g. [21,22].

In this paper, we examine, at a fine granularity, the details of a role specification (Role in Figure 2), specifically expanding upon its Responsibilities attribute.

3 EXTENDING THE DEFINITION FOR ROLE

The proposed model of an Agent Role has been developed through a careful examination and analysis, including textual and situational analysis, of the many different roles or role-like behaviours described in the literature e.g. [3,21,23-33]. It is interesting to note that some authors e.g. [34] look at both the implementation (endeavour domain) and metamodel level characteristics (i.e. different level of abstraction) of roles in the same context, although this runs the risk of introducing inconsistencies in the model. In this paper, we have therefore concentrated initially on the dynamic characteristics of roles.

After the initial model was developed, it was informally tested against a number of scenarios to ensure that there were no gaps or inconsistencies in the model. These tests were carried out by selecting typical example applications given in the research literature and examining them to see if the model of a role developed here provided a satisfactory

explanation of the behaviour exhibited in those examples. The key criterion for this evaluation was to minimise the conflicts between these previous studies. The model was evaluated for its ability to be implemented and scaled for large scale systems.

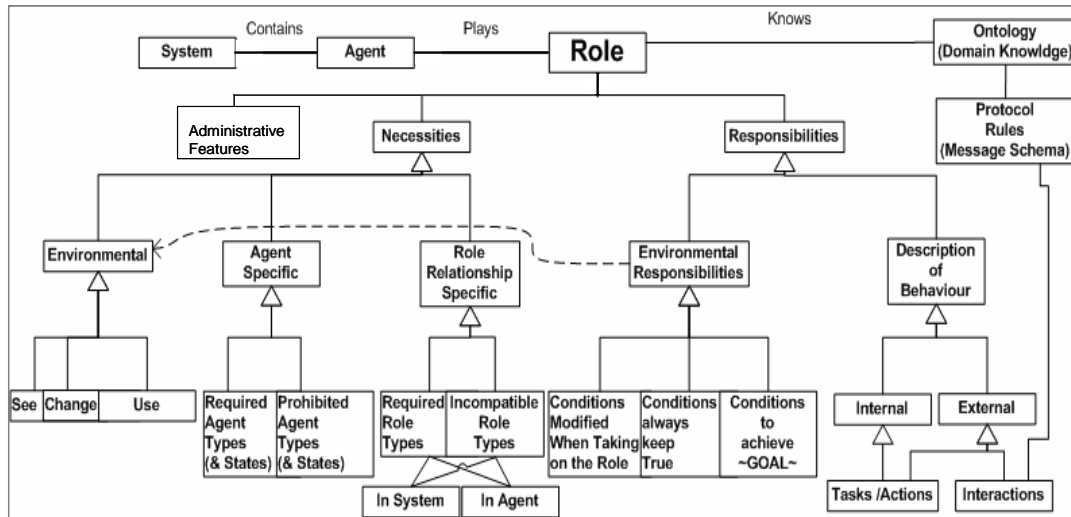


Figure 4 Extended metamodel for Role

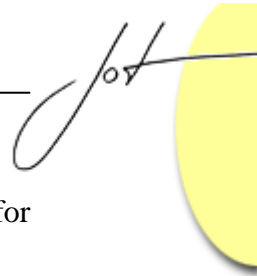
The output of the process described above is detailed in this section and is summarized graphically in the metamodel of Figure 4, which shows the main groups of characteristics:

- Administrative Features
 - Features that help the designer identify the Role and decipher the Agent holding the Role and its membership of an Agent community.
- Responsibilities.
 - Features that describe what other system entities expect of an Agent playing the Role.
- Necessities.
 - Features that describe the things that an Agent playing the Role will need in order to achieve the desired outcome.

Each of these groups will be explored in more detail in the following sections.

Role Model: Administrative Features.

In order that the system is able to use the role, it is important that it must be able to identify that role from the many others being played by an agent (or indeed other agents). The simplest form of these administrative features is the Role Name, which would be a short textual descriptor of the Role. This initially appears to be the only requirement but,



as the Role Model was developed further, it was apparent that this was insufficient, for the reasons discussed below.

It is widely accepted in the literature [5] that it is possible for an agent to play multiple roles. It is therefore also implied that an agent may wish to play multiple versions of the same role; thus a role identifier based only on a role name would be insufficient. A more complete role identifier would consist of the five following parts:

- Role Name: a succinct textual description of the Role, this will be the same of all of the agents that play the same role
- Role Description: A detailed description of the purpose of the Role, which will document in detail the reason and motivations behind the creation of the Role as well as any unusual features of the Role.
- The Name of the Agent that is currently playing or holding the Role:
- The Name of the Community of which the Agent holding the Role is a member
- A Role Index: a unique (numeric) identifier, (sequentially) assigned whenever a role is adopted.

This last piece of information is necessary so that the model can deal with the rare situation where an agent will want to hold two copies of the same role. These features will be assigned to the role when an agent within the system adopts the role.

Role Model: Responsibilities.

A role describes the normative or typical behaviour that is expected to be performed by an agent within a particular agent community. Thus the role must have a list of actions that it can perform. This is the behaviour that the role prompts the agent to perform (Figure 3). Roles are usually used to describe behaviours where the agent interacts with other agents but, in order that the agent uses the role to its fullest potential, it is useful if the role also describes the changes to the internal state of the agent brought about by playing the role.

This characterisation of the normative behaviour of agents playing a role has been denoted by many different names in the literature. Some writers, for instance [35], call it a *Protocol*. This can be confusing since the term “protocol” is also used to describe the guidelines under which role-to-role interactions takes place, rather than internal and external behaviours of the agent, as discussed here. The description of what the role tells the agent to do will be referred to as a “*Description of Behaviour*”. This solves any issues with misunderstanding of the concepts encapsulated by a culturally-specific description of this feature (e.g. script, act, repertoire).

One of the most important responsibilities of the role is to describe the interactions that the agent can perform whilst it is enacting the role. The role may also refer to sets of guidelines under which these interactions take place. These guidelines are variously referred to as Protocols, Languages, Conventions, Message Schemata or, more formally, as Speech Acts [36,37]. This paper will refer to them as “*Communication Guidelines*”.

These “Communication Guidelines” can be seen as part of the “*Description of Behaviour*”, although they also combine elements from outside the role. These external elements are an acknowledgement that the way the role behaves is constrained by the rules of the environment and community in which the agent and role exists (this relates to Group of Figure 1 and the classes System and Organization Definition in Figure 2).

The role may also specify which of the agent’s services, available to other agents within the community, are enabled or disabled while it holds this role. This feature is also included in the “Description of Behaviour”.

Since the agent has a goal, which is one of the requirements that make it an agent, the role will hopefully lead the agent closer to that goal. However, this progress towards the agent’s goal may not necessarily be direct nor obvious. This implies a distinction between the goal of the agent and the goal of the role. The role must have some sort of sub-goal of the agent’s goal, but the role sees the sub-goal as its own goal. In this paper, we will refer to this agent goal as a *goal* and a role goal as a *target* to avoid confusion between the two. It might be said that the target is a waypoint on the way to the final destination of the agent goal (see also [38]).

The goal and the target must agree; at a minimum they mustn’t be contradictory or incongruent, since there may be circumstances where it might be easier for an agent to step diagonally sideways before stepping forward, called *tacking*, toward its goal.

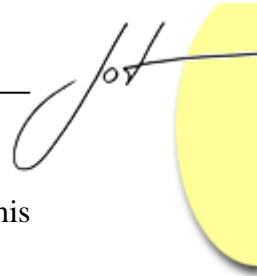
The role has to conform to some set of conditions that define the state in which the agent-role pair can start, operate and terminate, without invalidating its frame of reference. This set of conditions is commonly referred to as *constraints*. There are three sets of constraints: pre-conditions, invariants and post-conditions. For each of these constraints there are two aspects: those internal to the agent (the agent state) and those external to the agent (the state of the environment).

One set of pre-conditions may be that there are certain roles already held by the agent or held by other agents in the community. Pre-conditions may also prescribe which resources must be available to the agent on the agent’s community.

Conditions that must hold for the whole time that the agent is holding or playing the role are called invariants. If the invariants are contradicted, the role become invalid and is dropped by the agent playing it.

The post-condition is the description of the state of the agent and the environment that must be set immediately after the agent has dropped the role. This combination of states represents the target of the role. The ultimate justification for the adoption of the role is the achievement of these target conditions.

- The responsibility features of the Role Model, which are to be included in the new Role Artefact, are thus proposed under two headings: first, “Description of Behaviour”, which describes the set of typical or normative behaviours that the role prompts the agent to perform. This feature is made up of:
 - “Communication Guidelines”: indicates the possible communications between the agent playing this role and other agents playing this or other roles.



-
- The Role Type will affect which of the agent's services are available. This characteristic is described by the features:
 - Enabled Services
 - Disabled Services
 - Role Target: A description of the ambitions or aspirations of the role

The "Communication Guidelines" relate to the Is-Destination-In and Is-Source-In features describe in the existing FAML metamodel. The communication guidelines also encapsulate the Can-Use characteristic, which links the Agent Role to a number of Message-Schemata or communication protocols. The Message Schema differs from the Communication Guideline in that it specifies which agents can take part in a message rather than the sequence of these messages.

It can be said that the Communication Guidelines use the Message Schemata but they also include the order and sequence of these messages. There does not appear to be any such relationship in the existing metamodel that describes this longer term communicative relationship between two or more role types. This type of descriptive characteristic is, however, implied in the existing metamodel by the aggregation of the Role-Relationship and the Message-Schema. The approach that we have taken in this discussion is to suggest that it may be an advantage to specify this more explicitly.

The modification of the availability of services must be defined with care to ensure that the role does not become overly agent-specific. This occurs as a result of the fact that the range of services provided will differ between different agent types. The solution to this issue is to allow agents to take on the role only if they provide all the active services, regardless of whether they provide the deactivated services.

It is useful to separate out the role target from the other behaviours that describe the steps taken to reach the target. This saves interpreting the actions and trying to determine why they have been taken. The target encapsulates both the actions taken and the motivation for taking those actions. It may be suitable for this feature to be associated with a Role Intention artefact that can be characterised as a specialisation of the Agent Intention classifier that is currently included in the FAML metamodel.

Secondly, when defining these descriptions of behaviour, it is important to identify both the characteristics of the behaviour internal and external to the agent that is playing the role. This leads to a set of conditions that define the state in which the role can exist. These are named the Environmental Responsibilities:

- Start in: The agent state / environment modification that the role is responsible for upon adoption.
- Operate in: The agent / environment state that the role is responsible for maintaining while it is active.
- Terminate at: The agent state / environment modification that the role is responsible for upon cessation of the role.

Environmental Responsibilities are sometimes confused with constraints (Pre-conditions, Invariants and Post-Conditions) but there is a clear distinction between the two. These

features relate to the conditions that the role is responsible for maintaining and not the conditions placed upon the role, which are dealt with in the Necessities group of features. For each of these Environmental Responsibilities, there are two aspects: those internal to the agent (the agent state) and those external to the agent (the state of the environment). The Environmental Responsibilities of the Role entity type are associated with *Facets* of the Environment that are already included in the FAME Metamodel.

Role Model: Necessities.

The Role Necessities are the abilities, permissions, resources and services that the Role needs in order that the Agent can adopt it although access to these resources and services does not necessarily guarantee the success of the agent achieving the target of the role. There are three general groups of necessities that have been identified in the modelling process:

- Environmental
- Agent-Specific
- Role-Specific

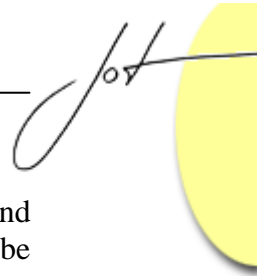
Environmental Necessities relate to the ability of the agent adopting the role to observe, manipulate and consume (a.k.a. see/change/use) resources or services in the environment of the system.

A particular role should have a list of the agent types that are required to exist within the system before an agent can adopt this specific role. This is further refined by specifying what range of states (or existing roles) these agents must be in (or playing) for the role to be taken on. The role will also contain a similar list of prohibited agent/states.

These necessities should always be viewed in the light of the fact that adopting a role may affect the agent's position within the community hierarchy (e.g. by giving the agent access to more or less resources or services). A typical example of this is the difference between Manager and Worker Roles, which have different access and privileges in the agent community. This means caution must be exercised before placing restrictions on role adoption within the system.

Finally, there are Role-Specific Necessities. These features relate to the availability of other roles within the agent and the system. The definition of the role will contain a list of roles that are required to be held by the same agent or to be present in the environment (held by another agent) together with a list of prohibited roles for the agent and within the agent community.

There are some scenarios where one role within a community must exist in order that another role may be adopted. A typical example would be the need for a worker role within the system before a manager role can be adopted. In the same way, the adoption of some roles may also be prohibited because of the existence of other inconsistent roles in the system. For instance, it may be that if a monarch role exists in the system then a president role cannot be adopted. However, once the monarch role has been dropped, the president role may be adopted.



The *Necessary* features of the Role Model are the characteristics of the system and agent that a role requires in order that it can be adopted. These features that are to be included in the new role artefact include (Figure 4):

Environmental Necessities include:

- The ability to see resources or services of the environment
- The ability to change resources or services of the environment
- The ability to use resources or services in the environment of the system.

These features are associated with the Facet Artefacts of the FAML metamodel. Facet is a generic descriptor or representation of *a property of the system environment with which the Agent can interact*.

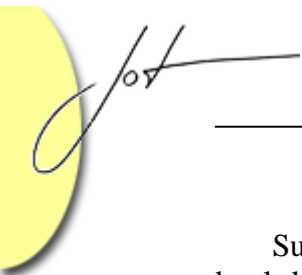
- Required Agent Types: an agent or agent state required in order that the role can be acquired by an agent.
- Prohibited Agent Types: prohibited agent or agent state which preclude the agent from taking on the role
- Required Roles: in the same agent and / in the environment,
- Prohibited Roles in the agent and / in the environment.

The four previous Role Model features are closely related to the Role Relationship already present in the FAML metamodel but they extend the currently available facilities by also specifying what the state of the agent must be (or not be) before the role is adopted.

4 REPOSITORY FOR ROLES (ROLE REUSE FRAMEWORK)

The process by which the agent adopts a role can be very complex, since instilling this type of behaviour requires some sort of reasoning engine. The development of such a reasoning engine requires a detailed knowledge of Artificial Intelligence and Machine Learning. The details of the construction of such behaviour are too complex to explore here. For roles to be truly dynamic, they need ready access to role specifications, the ability to find and adopt roles rapidly and, in some cases, to customize roles autonomously. A key element in this is the provision of a storage mechanism – called here a repository or a “framework”.

The framework we are proposing should, firstly, help to make the actual detailed mechanism for choosing the roles a lot simpler due to the modularisation of the process and, secondly, make the algorithm used immaterial to the outcome, i.e. that the outcome will not depend on the algorithm used. This will make it increasingly possible for heterogeneous agents to work in a predictable manner, no matter what method they use to choose and adopt a role. This ability to disregard the implementation of the Role will become increasingly important as reuse develops, especially when coupled with the various types of adoption discussed below.



Such a Role Reuse Framework would work much like a directory in a building or a bookshelf in a library (thus giving rise to an alternative appellation of “Role Library”). Any agent entering the community should be able to *look up* which roles are available within a system. The agent would then be able to select a role, based on its internal reasoning capabilities, that matches both the agent’s goals and the goals of the community or system.

The types of relationships that are held between different agents in a community and the bonds on trust / closeness / respect [31] that these agents form are important issues for the assignment of roles to agents in a community and require separate exploration.

There are a number of criteria that the Role Reuse Framework should try to maintain to order to achieve the aims stated above. Firstly, the Reuse Framework should be accessible to all of the agents that are members of the community. Without making the framework visible, agents in the system would not be able to make an appropriate and fully informed choice of which roles best suit their needs.

While it might at first seem appropriate to apply some restriction on access to the roles by using a level of authority or hierarchy to the availability of the roles by restricting the view of the role availability within the system, this type of restriction will be enforced by the details specified in the characteristics of the role description placed by the role on the agent adopting it, i.e. the list of prohibited agent types that has been noted as an internal characteristic of the role. Allowing all of the agents in the system to view the available roles enhances an agent’s ability to interact effectively within the system.

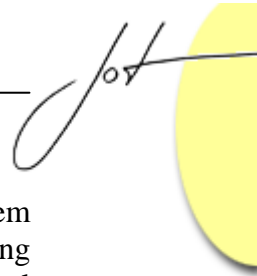
The second criterion to which the Role Reuse Framework needs to adhere is that the framework should supply the agent adopting the role with all the information needed so that the agent can adopt and play that role. This means that the description of the role that is stored in the framework should contain all the information on the role characteristics discussed in the previously developed Agent Role Model (Figure 4).

Thirdly, the Role Reuse Framework should provide some additional information that will be required to ensure that adoption and adaptation occur smoothly. Such additional characteristics of the Role Template will inform the agent what they are allowed to do with this role in terms of how strictly they must follow the specified description of the role. These modifiers should be applied to all of the Role Model characteristics.

Allowing this type of modification of roles may initially seem to leave the system open to perversion and corruption of the roles by the community’s agents. However, without the ability to modify the roles available to the community, there is no possibility for the system roles to adapt and improve in the future. This is a trade off-that must be undertaken with caution.

If the derived role proves to be a useful and distinct development of the role already stored in the framework, then it can also be added into the framework. The newly derived role will then be available for other agents to use in the future and be developed further. These modification indicators are described below.

The last feature of the Role Templates stored in the Role Reuse Framework that needs to be discussed is that of the abstract nature of some of these Role Templates. In



may be that, in some circumstances, the system designer would like to build the system with maximum flexibility. In this case, the system designer will want to avoid being overly prescriptive. If a system is overly prescriptive, this will invalidate the general principle of agent-oriented systems as being goal-oriented. To solve this issue, the designer will need to provide some roles that are very broad, specifying in the widest possible terms what it possible in the agent community, with the right authority/privilege. These roles will be designated as Abstract Role Templates. An indicator is needed within the Agent Role Template to tell the agents in the system that this role is a generic *Abstract* Role Template.

5 ROLE ADOPTION

In order that the agents using the Role Templates know what sort of adoption is acceptable, the Role Library will have to provide an indication of acceptable use. This is the one major feature that differentiates the Role Templates from the description of Role Types discussed earlier. The Role Template and the Role Types are in other ways very similar. Consequently, two features have to be added to the existing Role Type definition in order that it can become a Role Template. The first feature is “*Type-of-Adoption*”. The Type-Of-Adoption feature can take one of four values:

- Completely Specified Adoption (Ordinary adoption / Instantiation).
- Adoption where the Implementation can be changed.
- Adoption where the both Implementation and Interface can be changed.
- Adoption from Abstract Role Templates.

The traditional mode of operation widely discussed in the literature [5] corresponds to the Completely Specified or Ordinary Adoption and Instantiation. In this situation, the agent that is adopting the role does not seek to make any modifications or additions to the Role Template stored in the Role Reuse Framework. It is envisaged that the large majority of role adoptions will be of this basic type. If this were not the case, the management of the Role Reuse Framework would place an undue burden on the system.

It is also likely that an agent community will undergo long periods of stability, followed by short and hectic periods of change and upheaval in the list of Role Templates stored in the Role Library. These perturbations of the community are likely to be brought about by major changes in the environment, changes in the demands placed upon the community by users, by the availability and cost of resources used by the agents of the system or by new advances in the implementation algorithms reaching the agent community. For this first type of adoption the Type-Of-Adoption feature of a Role Template will be assigned the value of *Ordinary*.

Agents that enter the community with new knowledge about a better way to complete their assigned task will be able to share these advances with the rest of the community. By using the second type of adoption, the agents will be allowed to change the implementation of a previously defined Role Template. This type of adoption will not

change the interactions between the different agents or the agents and the users of the system. The change will only affect the way in which the actions demanded or required of an agent playing a particular role are processed. This second type of adoption presents system developers and maintainers with a new and unique way to maintain and upgrade their system. By deliberately sending an agent into the community with certain characteristics, they can modify the way that the agent community handles the demands placed upon the system by its users. These actions will be able to be achieved by modifying only a small amount of the implementation. The actions can also be completed without taking the system offline. This may be a major advantage where system reliability or up-time is an important measure.

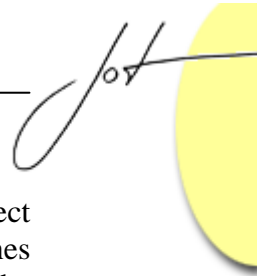
For the second type of adoption, the Type-Of-Adoption feature of a Role Template will be assigned the value of `Implementation`. Any Role Template that is assigned the Type-Of-Adoption value `Implementation` will also be able to Adopted as if it were an Ordinary Type-of-Adoption.

The third type of adoption is where an agent is able to change both the implementation and the interface of the role as defined in the Role Template stored in the Role Reuse Framework. This type of adoption is the most distantly removed from the typical adoption proposed in many other papers. This is also the type of adoption that is most likely to be open to abuse by malicious agents. Therefore, caution must be applied when setting the limits regarding to which Role Template this type of adoption can be applied. However, this type of adoption will also allow system designers and maintainers to increase the advantages available with the implementation-only modification, by allowing them to provide additional functionality from the system, by the addition of new Role Templates based upon roles already in the system.

One of the other advantages of this type of adoption is the ability of the agent community to perform optimisation of tasks that share some common link. The agents within the community should be able to be grouped together in similar roles in order for them to be carried out more efficiently, or to decompose the roles, so that they can be more efficiently distributed amongst the members of the community. This is especially advantageous where the agent community suffers from a restricted supply of resources in their environment. This is also an advantage where the agents that make up the community are limited in their ability to process requests. In this case, the sharing and delegation of tasks become very important.

For the third type of adoption, the Type-Of-Adoption feature of a Role Template will be assigned the value of `Interface`. Any Role Template that is assigned the Type-Of-Adoption value `Interface` will also be able to be adopted as if it were an `Implementation` or Ordinary Type-of-Adoption.

The final category is adoption (and specialisation) from an Abstract Role Templates. This is a peculiar form of adoption specifically included to give system designers the facility to define generic modes of behaviour. The Abstract Role Templates defined do not specify Role Targets or implementations, they only specify the range of behaviours that are acceptable within the agent community. No roles can be directly instantiated from this type of Abstract Role Template, but they supply an important set of guidelines



as to how all other roles within the system should behave. Agents with the correct authority can take the abstract template and devise a new role set within the guidelines provided by the Abstract Role Template. This type of Role Template encourages the system designers not to over-specify the systems they are designing, allowing them to develop with the greatest freedom possible, generating a system with the ability to be applied in or to deal robustly with changes to a wide range of circumstances.

It is anticipated that the system modeller may implement either domain-specific or generic business rules that all the agents in the system should follow. This will leave the system free to operate within these broad restraints, finding the optimum solutions to problems within this set of guidelines.

For the fourth type of adoption, the Type-Of-Adoption feature of a Role Template will be assigned the value of `Abstract`. Although no direct instantiations of Role Templates with this value of Type-Of-Adoption can be made, roles are derived by fusing this Abstract Role Template and are assigned the Type-Of-Adoption value of `Interface`, `Implementation` or `Ordinary`.

6 DISCUSSION

It is hoped that the dynamic model of an agent role provided in this paper will be the basis for a working model for the use of roles in Multi Agent Systems at the implementation level, drawing together, as it does, the disparate ideas currently discussed in the research literature.

One of the major objectives in creating this Role Model is to change the way that system designers think about the Multi Agent Systems they are designing. The outcome of this shift in understanding is that the designers realize that, while the agents in these systems are the major concept and are representative of entities carrying out activities, it would be advantageous to place the description of these activities, or at least the possible intermediate steps needed to achieve these activities, into the roles available within the system. This will allow the knowledge of how these steps should be performed to be shared amongst all the agents in a community, not just known by a particular agent.

The resultant Role Model and constructs described in this paper should also raise the awareness of system designers and modellers that the static structures that are typically developed during the analysis and design phases of software development do not necessarily represent all of the dynamic behaviours that may be present in a running system.

The addition of the new Role Entity to the FAML metamodel should serve to strengthen the FAML metamodel. In particular, it has been shown here that the framework has proved to be very flexible, requiring only minor modifications in order to accept the Role Model. The FAML framework has also been shown to provide good coverage of the software development lifecycle, in that the model developed here is able to be applied at both the design-time and run-time phases of the lifecycle.

The development of a facility, like the Role Library, to share and reuse roles at run-time opens up a whole new range of possibilities with regards to the ability of Multi Agent Systems to adapt to new and changing situations. This framework for reuse also provide researchers who are interested in different ways of modifying agent behaviour with a setting or infrastructure in which to conduct their experiments.

7 LIMITATIONS AND FURTHER WORK

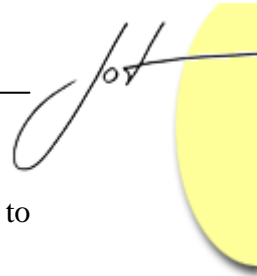
This paper has not discussed the mechanics of how a role is selected and adopted by an agent. What has been provided is a framework in which these activities can take place. This mechanism will need to be inspired by the area of Machine Learning and is deemed outside the scope of this paper. If an agent is required to select a role autonomously, as would be suggested by the fact that all intelligent agents should be autonomous, then the agents will have to be embodied with some sort of reasoning engine that will allow them to determine which role is most suitable for their purposes, i.e. allowing them to progress the furthest toward their goal. To compound this inherent difficulty, all this has to be done in an environment that is constantly changing.

It will be interesting to see how the idea of tacking towards a larger goal using intermediate goals progresses in the future. This aspect of the relationship between the agent and its goal and the role target will be an important influence on how and why an agent adopts a particular role. It is also important to consider how we measure the progress towards the agent's goal or, indeed, the goals of the community or system as a whole. Without being able to determine if the system or community goals have been met (or optimised as best as is possible for this point in time), it will never be able to be determined as to whether the way the system is implemented is a success.

This paper has not covered any issues surrounding the conversion of the modelled system (roles and agent) into an implementation language or tool. Although this discussion could inform the development of tools or techniques, it is also likely that, after trying to complete this stage, it will be able to be said with some weight which approach to designing and building Multi-Agent Systems is more effective, and therefore more likely to be acceptable to industrial practitioners.

Neither has this discussion dealt explicitly with agent mobility, although this type of agent behaviour is suggested or implied by the fact that an agent can join and leave a system. It might also be the case that the only way an agent can join or leave a system or community would be to be created or destroyed, but this would go against the idea of a persistent agent.

We have not dealt with who is responsible for the ownership or maintenance of the Role Reuse Framework. This is a crucial issue, as there has to be some filter as to which newly modified roles are accepted back into the framework. It is likely that this issue will be closely linked to the concepts of trust in the agent community. That is, that agents that have established a rapport within the community through long and diligent service will be allowed to modify or supervise the modification of the Role Reuse Framework.



Some of the applications of this type of Multi Agent System, which utilises roles to implement dynamic behaviour, include:

- Auction applications: where the agents in a system are grouped into several communities, which may represent auction rooms that are dealing in a particular item or performing a specific type of auction. In this case, the two major roles are the buyers and sellers within the auction room.
- Resource allocation: In a system where a large number of parties would like to access a limited resource pool, such as an Aircraft Control Systems, agents can be used to represent the parties. The roles in this type of system correspond to the parties that desire to hold or use a resource, and the parties that control access to the resource or possess a surplus amount of the resource. The system self-mediate access to these sparse resources.
- Telecommunication infrastructure: Telecommunications systems have been an early test-bed for Multi Agent Systems, whether it be routing of Internet requests or phone calls. The characteristic of telecommunications networks, that they require the optimum up-time and are often prone to individual failures that can trigger cascade failures, make them an ideal environment for Multi Agent Systems. The agents are embodied in the nodes or links that make up the network, with the roles related to the transmission and coordination of switching. The roles that these agents play depend on the availability of the resources around them.
- Other examples include:
 - Project Task Delegation Systems
 - Meeting Negotiation Systems
 - Collaborative Filtering Tools
 - Unmanned Vehicles

Multi Agent Systems offer innovative solutions to these and many other problems that have been difficult to solve with traditional systems and software engineering methods. One of the key features that unifies all these problems is that the parties or entities within these systems must behave differently depending on their environment. This environment is often too complex to be fully mapped into deterministic systems; the solution must be able to cope with a wide range of unforeseen circumstances.

8 CONCLUSIONS

In this paper, it has been shown that, although this area of research currently contains many competing ideas and influences, it is possible to draw these disparate influences together into a single cohesive model. The concept of an agent role in Multi Agent Systems has been shown to be an important construct at many different stages of the software development lifecycle, not solely at the design, modelling or implementation stage.

By creating a model of this agent role, this discussion has extracted the key characteristics of the agent role. By fully describing the Agent Role Model, it has been able to be shown that many of the features that have previously been considered to be attributes of the Agents within a system are actually part of the roles.

The abstraction of this Role Model enables researchers and practitioners to use the concept of an agent role in a wider context. This activity also allows the FAML Framework to be tested and shown to be a useful setting to explore the ideas in an agent-oriented system.

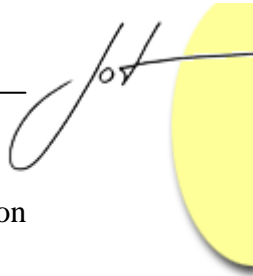
Finally, the paper has proposed that, in order to maximise the potential of the previously defined agent roles, it will be necessary to provide agent communities with a structure for storing Role Templates. An added advantage of this approach is that this facility will also allow the adaptation of the roles used in the system as circumstances change. This will not only increase the robustness and adaptability of Multi Agent Systems but will also allow them to remain useful for a longer period of time.

ACKNOWLEDGEMENTS

This is contribution number 08/06 of the Centre for Object Technology Applications and Research.

REFERENCES

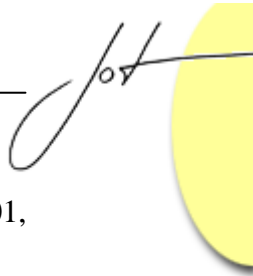
- [1] J. Odell, H. Van Dyke Parunak, S. Brueckner, J.A. Sauter, Changing Roles: Dynamic Role Assignment, *Journal of Object Technology*, 2 (5) (2003) 77-86.
- [2] J.L. Koning, I. Romero-Hernandez, Generating Machine Processable Representations of Textual Representations of AUML, in F. Giunchiglia, J. Odell, G. Weiß (Eds.), *Agent-Oriented Software Engineering III.- Third International Workshop, AOSE 2002, Bologna, Italy, July 15 2002, Revised Papers and Invited Contributions*, LNCS2585, Springer-Verlag, Berlin, 2003, pp. 126-137.
- [3] G. Cabri, L. Ferrari, L. Leonardi, Rethinking Agent Roles: Extending the Role Definition in the BRAIN Framework, *Procs. 2004 IEEE International Conference on Systems, Man and Cybernetics, Volume 6*, IEEE Computer Society, Los Alamitos, USA, 2004, pp. 5455-5460.
- [4] K. Chan, L. Sterling, K. Karunasekera, Agent-Oriented Software Analysis, *Procs. 15th Australian Software Engineering Conference (ASWEC 2004)*, 13-16 April 2004, Melbourne, Australia. IEEE Computer Society, 2004, pp. 20-27.
- [5] J. Odell, H. Van Dyke Parunak, S. Brueckner, M. Fleischer, Temporal Aspects of Dynamic Role Assignment, in: P. Giorgini, J. Müller, J. Odell (Eds.), *Procs.*



Agent-Oriented Software Engineering (AOSE) IV, Lecture Notes on Computer Science volume 2935, Springer, Berlin, 2004, pp. 201-213.

- [6] J.J. Odell, H. Van Dyke Parunak, M. Fleischer, Modeling Agent Organizations Using Roles, *Software and Systems Modeling*, 2 (2003) 76-81.
- [7] J. Odell, H. Van Dyke Parunak, M. Fleischer, The Role of Roles in designing effective Agent Organisations, in: A. Garcia, C. Lucena, F. Zambonelli, A. Omicini, J. Castro, J. (Eds.), *Software Engineering for Large-Scale Multi Agent Systems*, Lecture Notes on Computer Science volume 2603, Springer, Berlin, 2003, pp 27-38.
- [8] B. Bauer, J.P. Müller, J. Odell, Agent UML: A Formalism for Specifying MultiAgent Software Systems, in: P. Ciancarini, M. Wooldridge (Eds.), *Agent-Oriented Software Engineering, First International Workshop, AOSE 2000*, Limerick, Ireland, June 10, 2000, Revised Papers. *Lecture Notes in Computer Science 1957*, Springer, Berlin, 2001, pp. 91-103.
- [9] A.F. Harmsen, S. Brinkkemper, H. Oei, 1994, Situational method engineering for information systems projects, in: T.W. Olle, A.A. Verrijn-Stuart (Eds.), *Methods and Associated Tools for the Information Systems Life Cycle. Procs. IFIP WG8.1 Working Conference CRIS/94*, North Holland, Amsterdam, 1994, pp. 169-194.
- [10] S. Brinkkemper, Method engineering: engineering of information systems development methods and tools, *Inf. Software Technol.*, 38 (4) (1996) 275-280.
- [11] H.T. Punter, K. Lemmen, K., The MEMA Model: Towards a New Approach for Method Engineering, *Inf. Software Technol.*, 38 (4) (1996) 295-305.
- [12] A.F. Harmsen, *Situational Method Engineering*, Moret Ernst & Young, 1997.
- [13] A.H.M. ter Hofstede, T.F. Verhoef, On the feasibility of situational method engineering, *Information Systems*, 22 (6/7) (1997) 401-422.
- [14] S. Brinkkemper, M. Saeki, F. Harmsen, 1998, Assembly techniques for method engineering, in B. Pernici, C. Thanos (Eds.), *Advanced Information Systems Engineering. 10th International Conference, CAiSE'98*, Pisa, Italy, June 8-12, 1998, *Proceedings, LNCS1413*, Springer Verlag, 1998, pp. 381-400.
- [15] D.G. Firesmith, B. Henderson-Sellers, *The OPEN Process Framework. An Introduction*, Addison-Wesley, UK, 2002
- [16] B. Henderson-Sellers, Method Engineering for OO System Development, *Comm. ACM*, 46 (10) (2003) 73-78.
- [17] G. Cabri, L. Leonardi, F. Zambonelli, Implementing Role-based Interactions of Internet Agents, *Proceedings of the 2003 Symposium on Applications for the Internet (SAINT'03)*, IEEE Computer Society, Washington DC, USA, 2003, p. 380.

- [18] G. Cabri, L. Ferrari, L. Leonardi, F. Zambonelli, Role-based Approaches for Agent Development, Third International Joint Conference on Autonomous Agents and MultiAgent Systems - Volume 3 (AAMAS'04), IEEE Computer Society, Los Alamitos, USA, 2004, pp. 1504-1505.
- [19] L. Ferrari, G. Cabri, L. Leonardi, BRAIN: A Framework for Flexible Role-based Interactions in Multiagent Systems, in: R. Meersman, Z. Tari, D.C. Schmidt (Eds.), On The Move to Meaningful Internet Systems 2003: CoopIS, DOA, and ODBASE - OTM Confederated International Conferences, CoopIS, DOA, and ODBASE 2003, Catania, Sicily, Italy, November 3-7, 2003, LNCS2888 Springer, Berlin, 2003, pp. 145-161
- [20] G. Beydoun, C. Gonzalez-Perez, B. Henderson-Sellers, G.C. Low, Developing and Evaluating a Generic Metamodel for MAS Work Products, in: A. Garcia, R. Choren, C. Lucena, P. Giorgini, T. Holvoet, A. Romanovsky (Eds.), Software Engineering for Multi-Agent Systems IV: Research Issues and Practical Applications, Vol. LNCS 3914. Springer-Verlag, Berlin, 2006, pp. 126-142.
- [21] E.A. Kendall, Role Model Designs and Implementations with Aspect-oriented Programming, Proceedings of the 14th ACM SIGPLAN Conference on Object-oriented Programming, Systems, Languages, and Applications, Denver, Colorado, United States, ACM Press, New York, NY, USA, 1999, pp. 353-369.
- [22] R. Depke, R. Heckel, J.M. Kuster, Roles in Agent-Oriented Modeling, International Journal of Software Engineering and Knowledge Engineering, 11 (3) (2001) 281-302.
- [23] L. Cavedon, L. Sonenberg, On Social Commitment, Roles and Preferred Goals, Procs. Third International Conference on Multi Agent Systems, IEEE Computer Society, Washington DC, 1998, pp. 80-87.
- [24] E.A. Kendall, Agent Software Engineering with Role Modelling, in: P. Ciancarini and M. Wooldridge (Eds.), Agent-Oriented Software Engineering. First International Workshop AOSE 2000, Limerick, Ireland, :LNCS1957, Springer, Berlin, 2001, pp. 163-169 75.pdf>
- [25] M. Wooldridge, N.R. Jennings, D. Kinny, The GAIA methodology for Agent-oriented analysis and design", Autonomous Agents and Multi Agent Systems, 3 (3) (2000) 285-312.
- [26] H. Lu, Z. Lu, Y. Li, Trust!- A Distributed Multi Agent System for Community Formation and Information Recommendation, 2001 IEEE International Conference on Systems, Man, and Cybernetics, Volume 3, IEEE Computer Society, Los Alamitos, USA, 2001, pp. 1734-1739.
- [27] G. Caire, W. Coulier, F. Garijo, J. Gomez, J. Pavon, F. Leal, P. Chainho, P. Kearney, J. Stark, R. Evans, P. Massonet, Agent Oriented Analysis using MESSAGE/UML, Proc. Second International Workshop on Agent-Oriented



-
- Software Engineering (AOSE-2001), Montreal, Canada, May 2001, LNCS2222, Springer, Berlin, pp. 119-135.
- [28] C. Bernon, M.-P. Gleizes, G. Picard, P. Glize, The ADELFE Methodology For an Intranet System Design, in: P. Giorgini, Y. Lesperance, G. Wagner, E. Yu (Eds.), Procs. Fourth International Bi-Conference Workshop on Agent-Oriented Information Systems (AOIS-2002), Toronto (Ontario, Canada) at CAiSE'02, 27-28 May, 2002, CEUR-WS, Volume 57, <http://CEUR-WS.org/Vol-57/>, 15pp.
- [29] Q. Yan, X. Mao, H. Zhu, Z.C. Qi, Modelling Multi Agent Systems with Soft Genes, Roles, and Agents, in: P. Giorgini, J.P. Müller, J. Odell (Eds.), Agent-Oriented Software Engineering IV, 4th International Workshop, AOSE 2003, Melbourne, Australia, July 15, 2003, Lecture Notes in Computer Science 2935, Springer, Berlin, 2003, pp. 231-245.
- [30] K. Chan, L. Sterling, Specifying Roles within Agent-Oriented Software Engineering, Procs. 10th Asia-Pacific Software Engineering Conference (APSEC'03), IEEE Computer Society, Los Alamitos, CA, USA, 2003, p. 390-395.
- [31] Z. Kobti, R. Reynolds, T.A. Kohler, The Effect of Kinship Cooperation Learning Strategy and Culture on the Resilience of Social Systems in the Village Multi-Agent Simulation, Proceedings of the 2004 IEEE Congress on Evolutionary Computation, Portland, June 2004, IEEE Press, 2004, pp. 1743-1750,
- [32] L. Cernuzzi, T. Juan, L. Sterling, F. Zambonelli, 2004, The Gaia Methodology: Basic Concepts and Extensions, in: F. Bergenti, M.-P. Gleizes, F. Zambonelli (Eds.), Methodologies and Software Engineering for Agent Systems, Kluwer Academic Publishers, 2004, 69-88.
- [33] L. Padgham, M. Winikoff, The Prometheus Methodology <http://www.cs.rmit.edu.au/Agents/prometheus/>, 2004.
- [34] R. Depke, R. Heckel, J.M. Kuster, Improving the Agent-oriented Modeling Process by Roles, Proceedings of the Fifth International Conference on Autonomous Agents, ACM Press, New York, USA, 2001, pp. 640-647.
- [35] T.L. Juan, L.S. Sterling, Achieving Dynamic Interfaces with Agent Concepts, Proceedings of the Third International Joint Conference on Autonomous Agents & Multi Agent Systems (Vol. 2), 2004, pp.690-697.
- [36] J.L. Austin, How to Do Things with Words, Harvard University Press, Cambridge, MA, USA, 1962.
- [37] J.R. Searle, Speech Acts, Cambridge University Press, Cambridge, UK. 1969.
- [38] B. Henderson-Sellers, N. Tran, J. Debenham, An Etymological and Metamodel-based Evaluation of the Terms “Goals and Tasks” in Agent-oriented Methodologies, J. Object Technology, 4 (2) (2005) 131-150

About the authors



Conor Ward received a B.Sc.in Computer Science and Physics in 1999 and a BE in Electrical and Information Systems Engineering in 2001 from the University of Sydney.

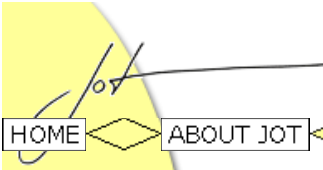
In 2006 he received a M.Sc. in Software Engineering from the University of Technology, Sydney, with a focus on software processes and agent oriented methodologies.

Conor has previously worked as a software integrator and maintenance engineering consultant with ABB Australia and as a Senior Software Engineer with Avolution Pty Ltd. Since 2007 he has been a Medical Software Engineer with Micropace Pty Ltd, developing embedded software for cardiac diagnostic equipment.



Brian Henderson-Sellers is Director of the Centre for Object Technology Applications and Research and Professor of Information Systems at University of Technology, Sydney (UTS). He is author of ten books on object technology and is well known for his work in OO methodologies (MOSES, COMMA and OPEN) and in OO metrics. He was recently awarded a DSc degree by the University of London for his

work in object-oriented methodology. E-Mail: brian@it.uts.edu.au



Utilizing Dynamic Roles for Agents

REFEREED ARTICLE

Conor Brendan Ward, University of Technology, Sydney
Brian Henderson-Sellers, University of Technology, Sydney



SEARCH

GO!

Abstract

The development of Agent Oriented Software Engineering (AOSE) and the use of roles within AOSE have been suggested as an important enabling feature in the future development of robust software systems. This paper seeks to identify and develop the definition of roles within an existing agent-oriented modelling language, namely FAML. The paper discusses the importance of role reuse, the process of role adoption and the advantages of role adaptation, and then proposes a framework for role reuse and adaptation within the FAML framework.

1 INTRODUCTION

A general consensus in the agent-oriented research community is emerging that roles appear to be an important feature of Multi Agent Systems [1-4]. An agent role is a characterisation of a set of normative or typical behaviours that an agent, within a particular system, may perform [5]. These provide a subset of an agent's responsibilities and are dynamic i.e. any particular role can be temporary and/or an agent may be playing more than one role at any given moment in time. Odell *et al.* [6] draw strong parallels to thespian roles i.e. taking the idea of a role as being like the script used by an actor, informing their behaviour and providing a set of guidelines of acceptable behaviour, but in which each player applies their own interpretation. All of the players act in concert to achieve an overall effect or Goal, but each of these players has a different part to play [6,7]. This leads to the notion of roles being assigned to agents operating within organizational groups (Figure 1).

It should be noted that one feature that a role does not have is a state. A state is a concrete characteristic or attribute, whereas a role is an abstract characteristic of the agent that is playing the role. A role informs an agent what it should do; it does not complete the action for the agent. A real life example of this is a manager role; an agent needs to play the role of a manager, the role doesn't exist without the agent. This being the case, it must also be said that a role can add to or modify the state of an agent; for example, the agent's state will be modified when taking on the role saying, "I am now playing the role of a manager".

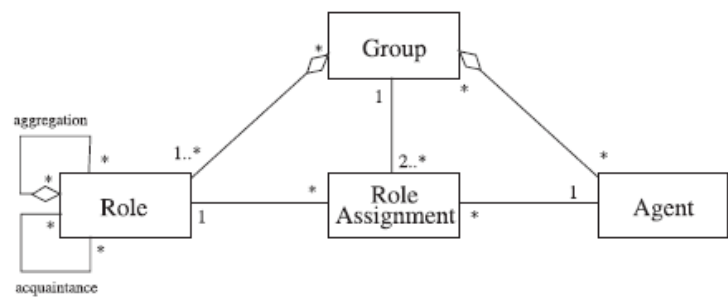


Figure 1. Agent-group-role class model with role assignment (after [6] With kind permission of Springer Science+Business Media)

Furthermore, roles are especially important when trying to characterise the interaction between multiple agents. The definition of a role can describe behaviours that are

[Subscribe to JOT's newsletter](#)
[O-O NEWS & EVENTS](#)