COPYRIGHT NOTICE

Designing for planned emergence in multi-agent systems

Kathleen Keogh[1,2] and Liz Sonenberg[2]

[1] School of Engineering and Information Technology
Federation University Australia
P.O. Box 663 Ballarat. VIC 3353 Australia.
[2] Department of Computing and Information Systems
The University of Melbourne, Australia

Abstract. We present an approach for designing organization-oriented multi-agent systems (MASs) to allow improvisation at run time when agents are not available to exactly match the original organizational design structure. Working with system components from an existing MAS organizational meta-model, OJAzzIC, the approach sets out five stages for the design process. We illustrate the design approach with an incident response scenario implemented in the Blocks World for Teams (BW4T) environment, and show how agents at runtime can improvise - for example they can adopt tasks even if those tasks do not precisely match a predefined role.

Keywords: Multi-Agent Systems, Coordination, Adaptation, Organizations

1 Introduction

People coordinating in dynamic environments can do so based on predefined roles, but also can operate with a degree of flexibility that allows individual improvisation to achieve shared tasks. Indeed, meso-level control has been shown to improve coordination and provide structure and collective responsibility to otherwise ad hoc teams of people [24]. Meso-level mediation and control has also been argued to ensure that micro-level, operational decision making does not interfere with or cause undesirable macro outcomes [20]. Similar multi-level approaches have been used in the design of multi-agent systems (MASs) for some time [6].

When designing and implementing a MAS, generally the process includes adopting a conceptual framework, developing a platform independent design, detailed design then implementation [23]. A more generic software engineering approach involves following a process of adapting and reusing existing meta-models to create an organizational model for agents (e.g. [1, 22]). Our focus is on organizational meta-models and approaches that provide organizational structures and frameworks that can be instantiated with some flexibility - to govern agents' behaviour but still allow improvisation – i.e. a form of planned emergence [20].

An organization-oriented MAS is one that is not considered primarily in terms of individual agent mental states, but involves organizational concepts such as roles, groups, tasks and interaction protocols, thus the focus is on what relates the structure of an organization to the externally observable behavior of its agents. The structure needs to be general enough to allow for context based adaptation at run time but specific enough to constrain agent's behaviour where necessary. An organizational meta-model defines a representation of the MAS organization, with the choice of meta-model driven by the domain requirements. Organization-aware agents then can prioritise goal selection based on organizational information as well as individual goals [5, 8]. Improvisation can be thought of as allowing agents flexibility to ignore or adapt role descriptions based on which agents are available. The conceptual framework requires us to adopt models for goals, roles, organizations and the domain. Sterling and Taveter refer to this as the conceptual viewpoint [23].

In this paper we address requirements drawn from complex, dynamic domains such as emergency management, where flexibility and improvisation is required. Characteristics of such settings include interdependencies between tasks, distributed coordination between members and adaptive, emergent behaviour. Appropriate knowledge sharing between agents is important, as is behaving with awareness of collective objectives, so that organizational goals can be as important as individual goals. We have previously addressed these requirements in the specification of an organizational MAS meta-model OJAzzIC [16, 17]. The meta-model specifies necessary components and relationships. In this paper we outline a process for the design of such organization-oriented MASs. The need for improvisation requires specific features and the contribution of this paper is to highlight the issues to be considered at design time regarding the meta-model and the way it is used to specify an organization that supported run-time improvisation.

The OJAzzIC meta-model provides a conceptual framework that builds on features from OperA+ [14], OMACS [7] and SharedPlans [11], and has been designed for situations when agents cannot rely on pre-scripted plans or pre-defined roles for coordinated behaviour, but must dynamically coordinate knowledge and plans. To describe the systems design approach, we adapt O-MaSE [9], an organisation-based multi-agent software engineering methodology. At design-time, the system requirements are described using goals and tasks, agents are defined in terms of capabilities and potential roles that could be enacted at run-time. Organizations are defined based on domain related roles and responsibilities. Agents are aware of the organizational structures and at run-time engage in organizational reasoning to prioritise goal selection based on organizational policies as well as individual goals.

In the execution model behind OJAzzIC, organizational reasoning at run-time includes an agent committing to social policies to ensure that appropriate knowledge sharing and coordinated behaviour occurs within the organization. The social policies operate as a meso-level, place explicit obligations on the agent within an organization regarding coordination of knowledge and plans, and also

allow the creation of adhocracies to facilitate coordination between emerging collectives of cooperating agents [17]. Our implemented scenario, developed in the Blocks World for Teams (BW4T) environment [15], demonstrates features in OJAzzIC that facilitate improvisation at run time. In specifying a series of issues to be addressed during the design phase, we highlight the need to identify complexities of the requirements in a domain and consider these at design time where possible.

In the next section, we provide a brief overview of the OJAzzIC meta-model then introduce our design considerations. In Section 4 we elaborate on each of the stages in the design process using an example system we built for an incident response scenario. In Section 5, we reflect on our findings, follow this in Section 6 with pointers to related work, then offer some concluding observations in Section 7.

2 OJAzzIC overview

OJAzzIC [16, 17] provides a meta-model based on a layered specification. High level modeling completed at design time provides flexibility and allows for improvisation at run time. The improvised behaviour is similar to that observed in a jazz musician who follows a high level score then improvises to add detail during a performance. This flexibility supports planned emergence, when agents dynamically combine to form a complex system [20]. In OJAzzIC, meso-level policies can be defined at design time and instantiated at run time to facilitate coordination by creating an ad hoc organization of agents (i.e. an *adhocracy*) [17]. An adhocracy is temporarily formed to achieve coordination between agents with a shared objective. Each organization provides a context for coordination. While an organization exists, all members know who else is involved so that appropriate knowledge can be shared and so that individuals can mutually adjust their own plans to fit in with others. Social policies in OJAzzIC explicitly define behaviour for role adoption, selection of goal objectives and communication obligations [17]. To provide flexibility, in addition to agents enacting a role, agents may adopt tasks or be allocated based on capabilities [7, 16]. Figures 1 and 2 show the design time models in OJAzzIC indicating how capabilities relate to goals and roles. More details can be found in [16, 17]. A distinctive feature in OJAzzIC that addresses the requirement of planned emergence is one that allows for responsibilities in a role to be split and shared by multiple individual coordinating agents, without a centralized coordinator. In OJAzzIC, as shown in Figures 1 and 2, agents may play roles and thus possess the appropriate role based capabilities, but agents may possess capabilities apart from role allocations so, if permitted, agents can improvise at a micro level (operational level) and 'fill in' where there is a need even if they do not match the required role description exactly.

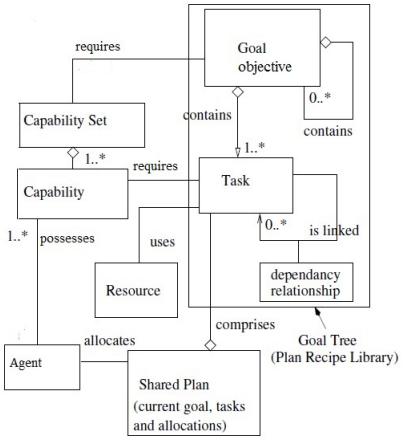

Fig. 1. Goal Task Model in OJAzzIC related to capabilities

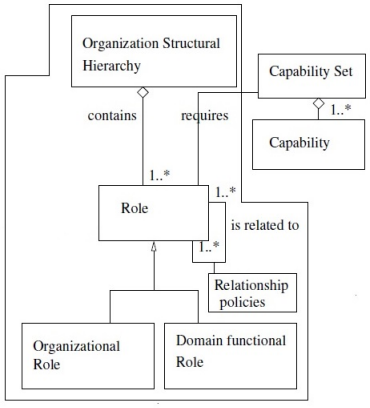

Fig. 2. Role Model in OJAzzIC

3 Design considerations

3.1 Scenario

To highlight design considerations and illustrate our requirements and proposed design approach, we use an incident response scenario used previously [13]. The scenario involves multiple agencies involved in rescuing injured individuals from a disaster area. Two agencies are involved: a medical agency (Medics) and a law enforcement agency (Officers). Medics are responsible for the rescue of injured parties and delivery to an ambulance; the objectives of the Officers include clearing away fights that break out between Bystander agents and clearing Bystanders as delegated by Medic agents. Bystander agents are from two opposing football teams and fights may break out that need to be resolved by separating fans into different areas.

The system is implemented using BW4T [15] and agents using the GOAL programming language [12]. Locations in the disaster scene are represented by rooms in a blocks world environment. Each room may contain injured individuals. Only one agent is permitted in a room at a time, so a Bystander agent in a room must be cleared before a Medic agent can enter the room to rescue an injured party. The domain is complex and dynamic enough to require considerable flexibility and coordination in agent behaviour. The problem involves first searching for injured participants, then coordinating the rescue. If there is no agent available to adopt a role, multiple agents may be able to coordinate in order to achieve the associated objective.

3.2 Design questions

Considering the desire for flexibility to improvise at run time, a number of issues must be considered at design time regarding agent knowledge and awareness:

What type of adaptation is required in the system? Organizational adaptation can involve structural adaptation of an existing organization as well as reallocation of roles used within the organization. It may also involve a revision of tasks chosen to achieve an objective. As a dynamic domain situation involves the possibility that agents may leave and enter the organization, the re-allocation of agents to tasks is also important. If new organizations can be instantiated at run-time, then at design time, if this can be anticipated, organizational policies and triggers for creation can be determined. Organizational policies can also be specified to guide dynamic coordination of knowledge and goal prioritisation.

How complete and adaptable are roles specified during design? The system requirements may be represented as a set of goals and a decomposition of goals and related tasks to be completed. In many MAS organizational approaches, the next step is to identify a set of organization/s and roles responsible for such tasks. It is common for roles to be directly associated with objectives or goals within an organizational MAS (e.g. [3, 14, 19]), so agents are associated with a role to determine the activities the agent may adopt. We seek to enable agents to dynamically adopt responsibility for tasks outside role-specific definitions where appropriate to achieve system goals. We also seek to enable agents to form adhocracies at run time in order to facilitate an awareness and context for coordinated behaviour [16]. These requirements lead us to adopt the notion of representing agents as individuals with particular capabilities and relationships separate from role specific definitions. Agents may adopt or be assigned predefined roles, however roles can be split and agents may also be matched to potential tasks using individual capabilities.

How much autonomy should be given to agents in terms of choosing tasks outside of a role specification? For example, the task of clearing away bystanders might be fulfilled by any agent type within the vicinity. However, rescuing an injured agent and moving them to the ambulance might only be adopted by an agent enacting the Medic role. If there is no specific 'Medic in charge', the Medic agents may agree amongst themselves who is rescuing each injured agent. Questions around leadership roles or domain specific roles and responsibilities should be considered in the design. The system design can be configured with flexibility where it is anticipated that agents may need to dynamically revise objectives and agent allocations to roles or tasks.

Which potential adhocracies can be identified at design time? Adhocracies emerge dynamically during a scenario and can cross existing organizational boundaries. These organizations persist over some time to assist with coordination of particular objectives and to facilitate inter-organizational coordination. The motivation to create an adhocracy is triggered by a need for coordinated behaviour or knowledge sharing commitments. During design, anticipate situations when adhocracies may form and triggers for their creation. Create social policies to define the triggers for creating and finalising adhocracies.

3.3 Designing an OJAzzIC based system

Based on our experience and considering the questions posed in section 3.2, we adopt steps in the design process based on an adaptation on the O-MaSE methodology. (For each step, the equivalent task in O-MaSE is shown in brackets.) Tasks in O-MaSE map well to produce corresponding OJAzzIC components and are consistent with our approach. Our approach is not linear, refinement and review may result in repeating steps. In O-MaSE, goals are used to define the objectives of the organization, whilst roles are used to define abstract positions within the organization that can achieve a given goal or set of goals. In O-MaSE, unlike OJAzzIC, there is no provision for splitting roles. Our approach differs from others in separating the design of a problem solution into two distinct design components: the problem design represented as a set of goals and tasks and the resources available described in terms of agent types and organizations. By keeping these distinct, we aim for more flexibility at run time. We do not presume a direct relationship between roles responsible for a goal and agents available to adopt roles. If there is not a direct match between the goals and the available agents' roles, then adoption of goals can emerge at a lower level based on agent capabilities and the capabilities required to achieve a task. We have implemented a simple incident response system using this process in order to clarify the design approach.

1. Define the Goal Model (O-MaSE:Model goals, Refine goals, Model domain, Model plans, Model protocols)
   - Create a high level goal decomposition of system objectives.
   - Break objectives into tasks that may be achieved by agents individually.
   - Where possible identify multiple alternatives to achieving an objective.
   - Identify dependencies between tasks and objectives, paying attention to requirements of synchronisation - e.g. before(task1,task2), concurrent(task1,task2).
   - Identify autonomy and control associated with each objective or task. Identify for each task or objective if it must be associated with any particular role.
2. Define the Organizational Model (O-MaSE:Model organizational interfaces, Model roles)
   - Identify long term organizations agents may belong to.
   - Define default agent types and domain roles associated within each organization.
   - Identify any inter-agent relationships.
3. Define the Agent Capabilities Model (O-MaSE:Define roles, Model agent classes, Model capabilities)
   - List capabilities to be given to particular agent types.
   - Identify capabilities required to achieve each task and thus required to fulfill each domain role.
4. Define the Role Model (O-MaSE:Define role goals)
   - Identify roles that agents of a particular type may be able to adopt within each organization (domain roles and structural roles). e.g. Medic, Leader
   - Identify responsibilities associated with roles within each organization. Map organizational roles to objectives they are responsible for.
   - Identify role relationships (e.g. dependency, authority, right to delegate etc.).

5. Establish Social Policies to be adopted within the run-time organizational contract (O-MaSE:Define protocols, Model policies)
   - role adoption responsibilities. e.g. Medic will prioritise locating injured then rescuing injured
   - knowledge sharing obligations. e.g. Medic will tell other Medics when an injured agent has been located or a rescue has been completed
   - organizational adhocracy creation triggers e.g. in rescue domain, if inter-agency coordination is required, a new adhocracy will be created to ensure appropriate communication and coordination occurs.
   - obligations between agents to establish shared organizational plans for coordinated tasks before goal actions are adopted.

In Section 4, we describe each stage in more detail using illustrations from our case study.

4 Incident Response Demonstration System

4.1 Define the goal model

Defining the goal model involves the following steps: Create a high level goal decomposition of system objectives and where possible, break objectives into tasks that may be achieved by agents individually. After decomposing objectives into a Goal Tree, identify if multiple alternatives plans exist to achieving an objective; Identify dependencies between tasks and objectives, paying attention to requirements of synchronisation; and Identify autonomy and control associated with each objective or task. Identify for each task or objective if it must be associated with any particular role. Based on the objectives and tasks, design plans and action specifications for how to achieve these.

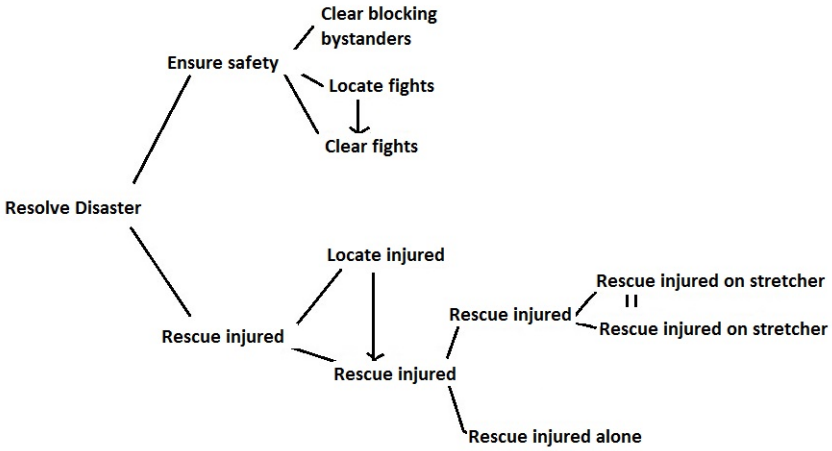

Fig. 3. Goal tree - incident response system

Figure 3 shows a goal decomposition tree for the incident response scenario.

A design decision should be made regarding autonomy and initiative: For each task - can it be actioned by any agent with the necessary skills (capabilities) or

must it be adopted only by one specific type of agent or an agent fulfilling a particular role? Our system may be configured to treat the task of removing blocking bystanders as a task that is only allocated to Officers, either as part of their role, or as a task delegated by request from a Medic, or as a task that Medic agents may also adopt by initiative if they are available. In our system, the feature allowing a Medic agent to clear bystanders using their initiative if available to do so may be turned on or off.

For each objective (landmark), it should be possible to identify at least one plan for how that objective can be achieved. The plan contains a list of states that must be achieved toward the final objective and a list of tasks or goals that will lead to successfully reaching the objective state.

For example, within our test system the plan for locating injured involves checking all rooms for injured and at least locating one injured agent. The plan definition is as follows:*plan(injuredLocPlan, injuredLocatedLmk, [checkedRooms, injuredLocated], injuredLocatedGoal)*. The corresponding Landmark objective injuredLocatedLmk is defined as:

$$landmark(injuredLocatedLmk, [checkedRooms, injuredLocated], [at(Ag, \_), injured(Ag)]).$$

4.2 Define the organizational model

It is necessary to identify long term organizations agents may belong to and the agent types associated with each organization. Also, it is important to consider adhocracies that may form and anticipate these and incorporate these at design time. Although this hasn't been implemented in our test system at this stage, the OJAzzIC model allows for the dynamic formation of adhocracies at run time.

In our demonstration run-time system, three organizational structures are created at design time: medicOrg, officerOrg and combinedOrg. For each organization, a list of objectives, a list of member agents, a set of roles and plans are identified. The organizational belief set is initialised to include the name of each org. The syntax *org(Org, Objlist, Memberlist, Rolelist, CurrPlanID, BeliefSet)* defines an organization. The specification of each org is as follows:

org(medicOrg, [injuredRescuedLmk], [medic1,medic2,medic3], [medic], [injuredRescuePlan], [orgname(medicOrg)]).
org(officerOrg, [injuredLocatedLmk, blockingBystanderRemovedLmk, fightLocatedLmk, fightStoppedLmk], [officer1,officer2,officer3], [officer], [injuredLocPlan,blockingBystanderRemovedPlan,fightLocatedPlan,fightStoppedPlan], [orgname(OfficerOrg)]).
org(combinedOrg, [blockingBystanderRemovedLmk], [medic1, officer1], [medic,officer], [blockingBystanderRemovedPlan], [orgname(combinedOrg)]).

Based on the initial goal decomposition, high level objectives are allocated to particular organizations. For example, the overall goal objectives allocated to

the Medic organization are to rescue injured and transport injured to hospital using ambulances. The Officers organization has responsibility for the ensure safety objective. Within each organization agent types can be identified. These types may have a set of associated related roles they are capable of enacting. Do all agents have the same capabilities or are some more specialised? For each agent type identified, what capabilities does that agent type have? The answer to this design question may impact upon flexibility in the final system when adapting to changes in agent availability. In our system, all Medic agents have the capabilities required to locate injured and enact a basic rescue. Medic agents also have the capability to remove blocking bystanders. We can also allocate to particular Medic agents the capability to perform a rescue on stretcher. Based on the capability set, the agents may be allocated to roles in the run time model or allocated (or adopt) responsibility for specific tasks.

When basic organizations have been identified, the designer needs to think about adhocracies that may form during a simulation. Adhocracies are organizations that are created in order to facilitate coordination between agents across organizational boundaries. For example, an Officer and two Medic agents need to work together to clear a safe exit for a complex rescue in an area of the rescue zone where access is limited due to a room collapse[3]. This complex rescue may require multiple coordinated activities and so an agreed plan for action and communication between these agents is required. In such a case, forming an adhocracy organizational structure is beneficial to ensure that agents are coordinated. At design time, if such an adhocracy can be anticipated, then triggers for the creation of an adhoc organization with members from both Medic and Officer agents based on the anticipated particular domain situation can be specified. The combinedOrg is a relatively simple test adhocracy based on Medic agent medic1 and Officer agent officer1 both being in an organization responsible for removing blocking bystanders. The presence of this organization means that when these agents perform the task of removing blocking bystanders, they will be obligated by social policies within the organization to keep each other updated about progress on this task.

4.3 Define the agent capabilities model

For each agent type, we list the capabilities they possess. The capabilities can be used to match agents to tasks in the system. The capabilities can also be associated with roles that are responsible for particular objectives as shown in Figure 1.

The agent capabilities model provides knowledge to enable organization-aware reasoning in terms of goal selection. In our example, Medic agents have capabilities to locate injured, rescue injured and remove blocking bystanders. Officer agents have capabilities to find fights, clear fights and remove blocking bystanders. In addition to the domain capabilities, organization aware agents

[3] room collapse is not implemented in the current system

have reasoning abilities to consider the organizational objectives when choosing to adopt a goal. An agent will first consider adopting an active landmark objective if it is an organizational objective and the agent is in a role that is responsible for that particular objective. Second, the agent will consider adopting an active landmark from within an active scene in which the agent is involved (no organizations involved). Third, the agent will consider adopting an objective if the agent is capable of fulfilling all tasks in an objective (apart from role allocations). Fourth, the agent will consider adopting a task that is part of a current objective if the agent is capable of achieving that task. When an agent has a list of considered objectives, the agent will select a goal to adopt based on a prioritisation of these objectives. For example, the Officer agent will prioritise locating fights over stopping fights and lastly removing blocking bystanders. These priorities, decided at design time, are specified by the order of rules in the program module or by explicitly defining priorities as policies.

4.4 Define the role model

The role model describes roles and responsibilities (objectives) associated with each role within the organization. A role has an associated capability set that defines the capabilitie(s) required in an agent to fulfill that role. Figure 2 shows the role model in OJAzzIC and how it relates to the capabilities. An agent may adopt or be allocated a role. In our system, for example, a medic role is defined as follows:

$$role(medic, [injuredLocatedLmk, injuredRescuedLmk, \\ blockingBystanderRemovedLmk, rescueOnStretcherLmk]).$$

This specifies that the medic role is responsible for the named objectives.

When specifying the role model, it is necessary to identify roles that agents may be able to adopt within each organization. These may include domain roles such as Medic or Officer and structural roles e.g. Leader. For each role, identify the responsibilities that should be associated with that role in terms of which objectives that role is responsible for achieving. In the above example, the medic role is responsible for four landmark objectives: injuredLocatedLmk, injuredRescuedLmk, blockingBystanderRemovedLmk and rescueOnStretcherLmk. Following the definition of roles, then role relationships can be identified (e.g. dependency, authority, right to delegate etc.). For example, Medic agents can delegate to Officer agents to clear blocking bystanders as there is a hierarchical dependency between the Medic and the Officer, defined as follows:

dependency(medic, officer, [blockingBystanderRemovedLmk], hierarchical).

4.5 Establish social policies

At this stage, we establish social policies to be adopted within the organizational contract. We are focused on social policies to facilitate coordination between

cooperating agents. We are not concerned with defining sanctions to impose on non-compliant agents although clearly in a broader open application, defining such consequences may be essential to controlling an agent society. In OJAzzIC agents are aware of the policies at an internal agent-level. The types of social policies to consider include [17]:

- role adoption responsibilities. e.g. Medic will prioritise locating injured then rescuing injured
- knowledge sharing obligations. e.g. Medic will tell other Medics when an injured agent has been located or a rescue has been completed
- organizational adhocracy creation triggers e.g. in rescue domain, if inter-agency coordination is required, a new adhocracy will be created to ensure appropriate communication and coordination occurs.
- obligations between agents to establish shared organizational plans for coordinated tasks before goal actions are adopted.

Social policies also make explicit the priorities to aid agents in their reasoning, selection and adoption of goals. Priorities for goal adoption eg. Medic agents low priority to remove blocking bystanders; priorities for role adoption e.g. Medic agents can be allocated the Medic Role; and priorities for communication - e.g. within an organization, inform all others of task progress.

The follow social coordination policies are directly implemented in our test incident response system:

- An agent A can delegate a task to agent B in order to achieve an objective if the agent A is playing a role with authority to delegate to role that B is enacting.
- If an agent A completes a task which another agent B is dependent upon, then agent A should tell agent B the task is completed.
- If agent A and agent B share an objective and agent A completes the objective, then agent A should tell agent B it has been completed.
- If agent A and agent B are both involved in the same scene, then when an objective in that scene is completed, then the agent, A should inform other agents in the scene, B that it has been completed.
- If agent A and agent B are both members of an organization O, then when an objective for that organization is completed, then the agent, A should inform other agents in the scene, B that it has been completed.

5 Observations

We implemented our scenario with organizationally aware Officer and Medic agents, and also built a comparison implementation using unaware Medic and Officer agents with the capabilities of a Medic and Officer respectively, but no organizational reasoning, awareness or capabilities. The latter agents could be coordinated by using an external coordinator, an organizational middleware agent, to allocate objectives to these agents. Our observations are primarily based on the performance of the organizationally aware agents. Following the design approach and considering which requirements can be given flexibility helped the

run time system to behave with that flexibility. Agents could show initiative to adopt tasks outside role allocation. Agents engaged in knowledge sharing within each organization so that coordinated behaviour occurred.

In the following discussion, we focus on the objective: rescuing an injured agent, to highlight the knowledge sharing benefits gained by our organizationally aware agents. The organizational instance defines which other agents to share with. Unsurprisingly, the organizationally aware agents share information that enables them to be more coordinated in their behaviour than a more basic agent. In the unaware system, with no coordinated knowledge sharing, each basic medic agent, when allocated the rescue task has to first individually search the potential locations and identify where the injured agents are, before planning a rescue. However, in the organization-aware agent system, as soon as any medic agent locates an injured agent, this knowlege is shared with all other medic agents by sending a message to each. This allows the organization-aware medic agents to focus on the rescue task sooner. Further analysis is required as our test system is expanded to focus on agents coordinating their actions to collaborate and achieve a shared goal (when a role is split or when the goal requires multiple agents working together). We are currently working to implement the shared rescue task where 2 medic agents perform a complex rescue using a virtual stretcher. In this case, the 2 collaborating agents create commitments to each other to form the agreement on a shared plan to work together with the stretcher rescue. Once they have both adopted this goal, they will remain committed to each other until the objective is reached.

When rescuing, the medic agent creates a specific goal to rescue a particular agent based on current beliefs as to the location of that injured agent. When an injured agent is delivered to the ambulance, the environment changes and all agents are able to perceive that change and update their beliefs. When a medic agent no longer believes that a particular agent is injured, any active goal to rescue that agent is dropped. When allocated the rescue objective, organization-aware Medic agents, due to their social policy for sharing beliefs send a message to all other medic agents when the rescue of a particular injured agent has been completed. In this case, with access to beliefs about rescued agents, the organization-aware agent can choose to adopt goals to rescue injured agents only if they have not already been rescued. This avoids the creation of redundant goals. Table 1 shows a sample of messages sent and received by medic agent, medic3 during a run of the system. medic3 shares relevant beliefs with medic1 and medic2 because they are in an organization: medicOrg. Medic1 and medic2 in turn update their own beliefs when informed by medic3. Social policies describe these obligations.

6 Related work

Within the field of agent oriented software engineering, there have been proposals for structures and concepts in general meta-models that could be used as components to design and build MAS [22]. Some have attempted to create generic meta-models for MAS that could be adapted to particular situations,

messages	beliefs
sent(medic1,rescued(23))	clearingRoom('DropZone')
sent(medic2,rescued(23))	occupiedclearRoom('DropZone')
received(medic2,rescued(23))	rescued(23)
received(medic1,rescued(23))	injuredRescued(23)
sent(medic1,rescued(22))	informed(rescued(23))
sent(medic2,rescued(22))	rescued(22)
received(medic2,rescued(22))	informed(rescued(22)), at('DropZone')

Table 1. Selection of messages and beliefs from medic3

for example, FAML [3]. These approaches are helpful to provide process and perhaps automate the design and implementation of systems. However, being generic, these meta-models do not address specific details or requirements such as adaptability and flexibility. Agüero and colleagues propose an organizational meta-model that could be used to create an organizational model, however the inner specifics of the organizational structure are left to a lower level of specification [1]. In our work, we use OJAzzIC as a meta-model that defines the organizational structure and behavioural policies used to instantiate MAS organizations. OJAzzIC is not a generic meta-model, but a meta-model with specific features that allow for flexibility in task allocation and improvisation of roles.

A number of methodologies for agent-oriented MAS design have been proposed. A good overview is provided in [23]. We draw attention particularly to organization centered approaches: OperA+ [14] and OMACS [7]. Determining an appropriate organizational MAS design for any given scenario is an open research problem, with some taking an empirical approach [10] and others defining generic meta-models by combining existing models e.g. JaCaMo [4] and FAML [3]. FAML does not attempt to address the specific requirements of adaptability and flexibility. Flexibility in terms of role adoption is addressed within OperA by including capabilities in role specifications and using a gate-keeper agent to allocate roles dynamically [2]. In this case, the gate-keeper agent selects an agent to play a role based on the agent matching the required capabilities. If an agent does not possess all the required capabilities, the role is not assigned. Similarly, OMACS achieves flexibility enabling goals and agent roles to be linked by matching capabilities dynamically. In OJAzzIC, an individual agent or set of agents may possess the capabilities to achieve a task or objective without necessarily being allocated directly to a role. We take the approach that agents do not need to be formally allocated to all roles, particularly when an unexpected situation emerges requiring an individual agent to improvise. We are not alone in specifying individual agent types to describe the system requirements [18].

Within an organizational model, social relationships are defined using abstractions such as roles, interactions, norms and policies. In models such as OperA+ and OMACS, the organizational model defines a set of roles that achieve the system goals. OperA+ represents multi-organizational interactions in two dimensions: specification (the organizational structure defined in terms of roles) and enactment (agents enacting roles). In FAML, there is a distinction made between design-time specification of organizations and run-time instantiation

models. This is similar to OJAzzIC. The social models in OperA+ and JaCaMo are similar to our social contract. OperA+ does not suit our requirements because it needs agents available at run time with an exact match to be able to enact the organizational roles, so flexibility relies on careful specification of the organization at design time defining alternative atomic or composite roles. JaCaMo is also built on an assumption that roles can be predefined at design time, although it makes explicit the possibility that a number of potential schemes can be defined with high level guidelines for instance stating the number of roles required. This enables some flexibility at run time, although still requires that agents who are able to enact the required roles are available.

Norms, rights and rules can be defined to constrain agent behaviour within roles, (e.g. OperA+) or policies can be defined at design time with associated commitments enacted at run time (e.g. OJAzzIC, O-MaSE). In FAML, the design time organization has associated policies defined for it. OJAzzIC social policies are consistent with the organizational policies in FAML except that in FAML policies are agent-external design time classes, whilst in OJAzzIC agents are aware of the policies at an internal agent-level. In OJAzzIC, organizations are run time entities created based on organizational definitions. Policies can be defined for a particular organization at design time, then are adopted in the organization in a run time contract of commitments between agents in the organization.

7 Conclusion

The previously introduced organizational MAS meta-model OJAzzIC specifies components and relationships intended to support the development of adaptive organisation-oriented MASs. In this paper we outlined a process for the design of such MASs. We proposed a number of questions to be considered during the design stage, in particular, we suggested explicitly considering flexibility, coordination, adaptability, autonomy and adhocracies that could be created. The process of making explicit choices about elements of the system that can be specified at design time was helpful in clarifying the requirements of the system overall. In particular, in trying to identify where flexibility and potential emergence can be anticipated and planned at design time, we can create a framework for run time instantiation of organizations. Each organization provides a context for agents regarding knowledge sharing and coordination. Additionally, we have found it possible within the described meta-model to achieve flexibility in terms of agents adopting tasks outside predefined roles. The organizational reasoning model when identifying potential goals to consider includes goals that the agent is capable of as well as goals the agent is responsible for due to role enactment. We did not focus on dynamic coalition formation or optimisation of coordination algorithms for the dynamic formation of MAS (e.g. [21]). However such work is relevant and could inform the automatic creation of adhocracies in response to dynamic and complex situations. Rather, here we focus on the organizational meta-model and processes to be considered in designing MAS with agents that embody appropriate awareness of the organizational structure. We address issues

around the knowledge sharing and coordination related knowledge that agents require in order to successfully coordinate behaviour.

Our approach shows promise for building MASs capable of flexible run time behaviour and we plan to conduct further trials to assess how organization-aware agents cope with other challenges: for example if an agent leaves the system, tasks are potentially unallocated creating a setting where remaining agents make a run time decision to adopt unallocated tasks. In future, we also intend to support implementation of policies that enable agents to achieve appropriate coordination by creating adhocracies at run time.

Our design approach addresses the requirements of flexibility and improvisation. At design time to enable run-time adaptation, macro level roles and tasks that achieve system objectives are to be specified. At run time, adhocracy formation and instantiation of policies guide the sharing of knowledge and plans between organization-aware agents in a particular context. This moves us closer to the aim of facilitating planned emergence within agent organizations.

Acknowledgments We thank the anonymous reviewers and those who have given feedback on the presentation – all of which helped improve this paper.

References

1. J. Agüero, M. Rebollo, C. Carrascosa, and V. Julián. Developing virtual organizations using MDD. In *Proceedings of Workshop on Agreement Technologies (WAT2009)*, pages 130–141, 2009.
2. H. Aldewereld, V. Dignum, C. M. Jonker, and M. B. van Riemsdijk. Agreeing on role adoption in open organisations. *KI-Künstliche Intelligenz*, 26(1):37–45, 2012.
3. G. Beydoun, G. Low, B. Henderson-Sellers, J. J. H. Mouratidis, Gomez-Sanz, J. Pavón, and C. Gonzalez-Perez. FAML: A generic metamodel for MAS development. *IEEE Transactions on Software Engineering*, 35(6):841–863, Nov. 2009.
4. O. Boissier, R. Bordini, J. Hübner, and A. Ricci. Unravelling multi-agent-oriented programming. In O. Shehory and A. Sturm, editors, *Agent-Oriented Software Engineering*, Heidelberg, 2014. Springer-Verlag.
5. D. Corkill, E. Durfee, V. Lesser, H. Zafar, and C. Zhang. Organizationally Adept Agents. In *12th International Workshop on Coordination, Organization, Institutions and Norms in Agent Systems (COIN@AAMAS 2011)*, pages 15–30, Taipei, Taiwan, May 2011.
6. D. D. Corkill and V. R. Lesser. The use of meta-level control for coordination in a distributed problem solving network. In A. Bundy, editor, *Proceedings of the 8th International Joint Conference on Artificial Intelligence. August 1983*, pages 748–756. William Kaufmann, 1983.
7. S. A. DeLoach. OMACS: a framework for adaptive, complex systems. In V. Dignum, editor, *Multi-Agent Systems: Semantics and Dynamics of Organizational Models*. IGI Global: Hershey, PA. ISBN: 1-60566-256-9, 2009.
8. S. A. DeLoach. O-MaSE: An extensible methodology for multi-agent systems. In O. Shehory and A. Sturm, editors, *Agent-Oriented Software Engineering*, Heidelberg, 2014. Springer-Verlag.
9. S. A. DeLoach and G.-O. J. Carlos. O-MaSE: A customizable approach to designing and building complex, adaptive multiagent systems. *International Journal Agent Oriented Software Engineering*, 4(3), 2010.

10. M. R. Franco and J. S. Sichman. Comparing and evaluating organizational models: A multi-agent programming contest case study. In *Pre-proceedings The 17th International Workshop on Coordination, Organisations, Institutions and Norms, AAMAS*, 2014.
11. B. Grosz and S. Kraus. The evolution of SharedPlans. In M. Wooldridge and A. Rao, editors, *Foundations of Rational Agency*, volume 14 of *Applied Logic Series*, pages 227–262. Springer Netherlands, 1999.
12. K. Hendriks and J. Dix. GOAL: A multi-agent programming language applied to an exploration game. In O. Shehory and A. Sturm, editors, *Agent-Oriented Software Engineering*, Heidelberg, 2014. Springer-Verlag.
13. A. Jensen, H. Alderwereld, and V. Dignum. Dimensions of organizational coordination. In K. Hindriks, M. de Weerdt, B. van Riemsdijk, and M. Warnier, editors, *Proc. of the 25th Benelux Conf. on Artificial Intelligence*, pages 80–87, 2013.
14. J. Jiang, V. Dignum, and Y.-H. Tan. An agent based inter-organizational collaboration framework: OperA+. In *Proceedings of Web Intelligence/IAT Workshops*, pages 21–24, 2011.
15. M. Johnson, C. Jonker, B. van Riemsdijk, P. J. Feltovich, and J. Bradshaw. Joint activity testbed: Blocks world for teams (BW4T). In H.Aldewereld et al., editors, *LNAI, ESAW 2009*, volume 5881, pages 254–256. Springer-Verlag, 2009.
16. K. Keogh and E. Sonenberg. Adaptive coordination in distributed and dynamic agent organizations. In S. Cranefield, M. Birna van Riemsdijk, J. Vázquez-Salceda, and P. Noriega, editors, *Coordination, Organizations, Institutions, and Norms in Agent System VII, LNAI*, volume 7254, pages 38–57. Springer-Verlag, 2012.
17. K. Keogh and E. Sonenberg. Coordination using social policies in dynamic agent organizations. In T. Balke, F. Dignum, M. Riemsdijk, and A. Chopra, editors, *Coordination, Organizations, Institutions, and Norms in Agent System IX, LNAI*, volume 8386, pages 1–20. Springer, 2014.
18. T. Miller, B. Lu, L. Sterling, G. Beydoun, and K. Taveter. Requirements elicitation and specification using the agent paradigm: The case study of an aircraft turnaround simulator. *IEEE Trans. Software Eng.*, 40(10):1007–1024, 2014.
19. J. Odell, M. Nodine, and R. Levy. A metamodel for agents, roles and groups. In *Proceedings of Agent-Oriented Software Engineering (AOSE) V*, volume 3382, pages 78–92. Springer, 2005.
20. J. Pitt, A. Bourazeri, A. Nowak, M. Roszczynska-Kurasinska, A. Rychwalska, I. Rodriguez Santiago, M. Lopez Sanchez, M. Florea, and M. Sanduleac. Transforming big data into collective awareness. *Computer*, 46(6):40–45, June 2013.
21. S. Ramchurn, A. Farinelli, K. Macarthur, and N. Jennings. Decentralized coordination in RoboCup Rescue. *The Computer Journal*, 2010.
22. V. Seidita, M. Cossentino, V. Hilaire, N. Gaud, S. Galland, A. Koukam, and S. Gaglio. The metamodel: a starting point for design processes construction. *International Journal of Software Engineering and Knowledge Engineering*, 20(04):575–608, 2010.
23. L. Sterling and K. Taveter. *The Art of Agent-Oriented Modeling.* MIT Press, 2009.
24. M. A. Valentine and A. C. Edmondson. Team scaffolds: How meso-level structures support role-based coordination in temporary groups, 2014. Working Paper, Harvard Business School.