applied sciences

MDPI

Article

Designing Multi-Agent System Organisations for Flexible Runtime Behaviour

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

[1] School of Engineering, Information Technology and Physical Sciences, Federation University Australia, Ballarat 3350, Australia
[2] School of Computing and Information Systems, The University of Melbourne, Melbourne 3052, Australia; l.sonenberg@unimelb.edu.au
* Correspondence: k.keogh@federation.edu.au

Received: 12 July 2020; Accepted: 27 July 2020; Published: 2 August 2020

Abstract: We address the challenge of *multi-agent system* (MAS) design for organisations of agents acting in dynamic and uncertain environments where runtime flexibility is required to enable improvisation through sharing knowledge and adapting behaviour. We identify behavioural features that correspond to runtime improvisation by agents in a MAS organisation and from this analysis describe the OJAzzIC meta-model and an associated design method. We present results from simulation scenarios, varying both problem complexity and the level of organisational support provided in the design, to show that increasing design time guidance in the organisation specification can enable runtime flexibility afforded to agents and improve performance. Hence the results demonstrate the usefulness of the constructs captured in the OJAzzIC meta-model.

Keywords: multi-agent systems; organisation; improvisation; flexibility; coordination

1. Introduction

1.1. Motivation

The structure of a human organisation, typically determined through definitions of roles and responsibilities and mechanisms for allocation or selection of tasks, can have great impact on the performance and utility of that organisation [1]. In human situations involving unpredictability and where specific solutions cannot all be fully scripted in advance, those involved improvise, i.e., adjust their plans to fit in with others by sharing relevant information and adopting revised plans [2]. In this paper we describe an approach by which intelligent software agents in organisational settings can similarly improvise and find runtime solutions that were not specified in detail at design time.

In human settings, the domain of emergency management provides an ideal framing of the problem: events unfold with considerable uncertainty and participants are typically a mix of individuals and teams with general and specialised expertise who have trained with agreed protocols that define roles and responsibilities [3]. Coordination of information across a disaster management or emergency management situation is difficult and often the cause of failure [4]. To achieve success, adaptation in the field is often essential, as event specific plans emerge and roles and responsibilities are adapted to fit the unfolding situation [5]. Such adaptation is not a replacement for procedures that are well designed for anticipated situations but is an essential response when an unexpected situation occurs [6]. Accordingly, and with an eye to supporting the future development of agent-based simulations for training in such domains [7], we highlight the agent's behavioural requirements considered in this research by referring to a rescue situation within an emergency management scenario. This requirements analysis provides the motivation for key features in the OJAzzIC (organisations

juggling adaptation and improvised coordination) meta-model, and is also used to highlight differences in our approach compared to earlier work.

In the development of software using multiple software agents working together in an organisation, typically the designer specifies expected roles and responsibilities, and policies or norms are also sometimes used to make behavioural expectations explicit. However, as discussed in detail later, this limits flexibility in behaviour. Specifically, what happens when at runtime there are not agents with an exact match to the design time roles available? What scaffolding is needed to enable agents in organisations to dynamically adapt to the contexts in which they find themselves, including improvised coordination with others?

To enable the necessary runtime reasoning about others, we employ the BDI (beliefs, desires, intentions) model proposed for an individual cognitive agent [8,9] to provide a way to implement agents that use rich models of reasoning (cf. [10]) and that has been used in the context of complex decision making [11–13] and agent-based simulation [14–16]. The BDI model represents agents each possessing individual beliefs about the world and having desires that can be mapped to goal states the agent seeks to achieve. Each agent has a plan library defining actions that can be taken in order to reach a particular desired goal. Based on an agent's current beliefs, the agent selects desires and adopts intentions to reach those desires based on options from the plan library.

We use the BDI architecture in a MAS setting where organisations are explicit; cf. [17–20]. An organisation comprises multiple agents who share an organisational goal, while holding individual beliefs, etc., and each agent has specified capabilities that can contribute toward reaching the organisational goal state or sub-states toward achieving that goal. Our agents are *organisationally adept* [17] in that they are aware of the organisation structure, can reason at the organisation level, and can adjust their behaviour for operational decision-making in keeping with known organisational guidelines (here, policies). We limit attention to cooperative and compliant agents and use the organisation construct to specify relationships and obligations between agents. We do not consider sanctions or penalties if agents do not adhere to an organisation's policies, although these are important in open systems.

The characteristics of the motivating domain of emergency management clearly map to the capabilities just outlined that are available within an organisational MAS. Importantly, the complexity of the domain is such that it is infeasible to specify a comprehensive set of plans covering all plausible scenarios. The focus of this paper is on the design requirements associated with improvisation, that is, what design features will enable agents to work within the constraints of the organisational configuration as established at design time, and also adapt plans from their library to achieve flexible individual runtime behaviour.

The contributions of this paper are to extend prior conceptual descriptions of the OJAzzIC meta-model [21,22] to a concrete implementation, and hence to demonstrate how increasing design time guidance in the organisation specification, in particular, using the constructs proposed in the meta-model, can enable runtime flexibility afforded to agents and improve performance. We situate this analysis with respect to other prior work on adaptive organisations. Specifically, we (a) describe limitations of prior research on meta-models for adaptive agent organisations, in particular, OMACS [23], JaCaMo+ [24,25], MOISEinst [26], OperA+ [27,28], and SharedPlans [29]; and (b) provide evidence that the individual and organisational constructs proposed in OJAzzIC can indeed be deployed in practice to yield effective runtime coordination and flexibility. This was achieved based on experimentation in a simulated rescue scenario. We show that OJAzzIC provides the scaffolding for the designer to implement a multi-agent system in which agents can behave with role flexibility at runtime by finding solutions that were not programmed with exact detail at design time. Key to the outcome is the choice of agent capability as a modelling unit in addition to role, and also the use of ad hoc agent organisations as constructs to both enable and constrain the range of knowledge sharing activity that is needed to support adjustment to runtime commitments.

The structure of the remainder of this paper is as follows. Next we offer a motivating example drawn from the domain of emergency management, wherein many activities can be completed by applying standard operating procedures, but rapidly changing local circumstances mean that participants may need to go beyond their pre-assigned roles and improvise to achieve organisational goals. Section 2 includes a brief account of key concepts in the literature of organisation-oriented multi-agent system design and development, with a particular focus on the use of meta-models in the design process and comparisons with related work. In Section 3 we briefly describe the OJAzzIC meta-model and an associated design method that is adapted from O-MaSE [30,31]. The simulation we use to demonstrate the usefulness of the approach afforded by the use of OJAzzIC was based on the motivating example presented earlier, and is described in Section 4, together with the scenario settings wherein different levels of organisational support for agent interaction and goal achievement are set out. The results are discussed in Section 5, with some overall discussion and concluding remarks in Section 6.

1.2. Demonstration Scenario

To highlight the behavioural requirements for agents to be addressed, we present a rescue situation within an emergency management scenario. The situation involves multiple agencies involved in rescuing injured individuals from a disaster area [32]. The scenario begins following a disaster event and an unknown number of individuals are injured. The rescue task involves searching for the injured and transporting them to ambulances ready to transport them to hospital for further medical treatment. The two rescue agencies involved in this example are a medical agency (medic agents) and a law enforcement agency (officer agents). Medics at the rescue scene are responsible for the rescue of injured parties and delivering them to a particular zone for ambulance pick up. Some of the injured can be rescued by one medic; however, some rescues require a stretcher to be carried by two agents, and thus two medics must work together to perform those rescues. The objective of the officers is to ensure safety and order at the rescue scene. Medic agents can delegate or request that officers clear away bystanders who are impeding rescue progress because they are in an area where a rescue must take place. Officers direct bystander agents into different areas so that they do not impede rescue operations.

The situation demands that the rescue agents behave with some flexibility and potentially improvise in order to rescue the injured in a timely way. For example, if an agent enacting the role of a medic is ready to rescue an injured party on a stretcher and another medic agent is not available to help carry the stretcher, a nearby agent in the officer role, possessing stretcher-carrying capabilities, could be asked to help carry the stretcher, even though that agent is not officially assigned or expected to fulfil the medic role.

2. Organisation-Oriented MAS Design and Development

When designing and implementing a multi-agent system (MAS), generally the process includes adopting a conceptual framework, developing a platform-independent design, detailing the design, and then creating the system [33]. In this section we briefly discuss design methodologies and then consider a number of existing organisation meta-models in terms of their ability to address our requirements.

2.1. MAS Design

Agent oriented software engineering (AOSE) is based around the design of systems using agents and key characteristics and abstractions relating to agents. The software engineering process, particularly the analysis and design process, is coupled with the abstractions and models adopted. A methodology considers the life cycle of development, including analysis, modelling, design, and implementation. A meta-model describes the components that should be included in a successful model design.

The development of multi-agent-based systems can involve multiple dimensions, including modelling the environment and using organisational models as a layer of abstraction above and separate from agents. Recently, attention has been given to merging multiple dimensions of development into a single paradigm aiming toward the development of complex distributed systems [34].

A number of methodologies for agent-oriented MAS design are reviewed in [35–38]. We note a selection of design methodologies, including methodologies related to organisation centred multi-agent systems: O-MaSE [30,31], Tropos [39], Gaia [40], Prometheus [12], INGENIAS [41], GORMAS [42–44], and ASPECS [45]. Dam and Winikoff [46] highlight similarities between of a number of agent-oriented software engineering methodologies. In particular, common elements are the adoption of use case scenarios and adopting models for goals and entities such as roles and capabilities. An overview of AOSE more broadly can be found in [33,46–48]. The reader is also directed to the series of workshops on engineering multi-agent systems (EMASs)—for example, [35,36,49].

If a MAS is to be deployed as part of an open environment in which heterogeneous agents interact, then agents may not share a predefined architecture or organisation structure. A service oriented architecture (SOA) is an integrated system wherein interoperability between distributed resources is achieved. In these systems, agents' functionalities are modelled as services. Previous work has included an organisation level to define social context and functional objectives [50], but the organisational level in such cases does not specify how agents achieve their outcomes, so the coordination of agent interactions has to be modelled separately, outside the organisation.

One approach to coordination within an organisation is to employ middleware or special coordination agents [51]. Another approach is to embed coordination implicitly in the plans that agents use; however, this limits agent awareness and flexibility. More recently, it has been proposed to coordinate agent interactions using artefacts and commitments external to the agents [25]. It is also possible to consider an organisation of agents as if it were an agent at a higher level of abstraction [52]. In that case, the agent organisation can be closed and comprising of homogeneous agents capable of achieving a particular goal. Other coordination approaches are described in [53].

2.2. Organisation Meta-Models for MAS

A recent trend in SE methodologies for MAS design is the use of model-driven engineering of MAS [54] or adopting a model-driven architecture [45]. This involves consideration of models and meta-models as artefacts used in the software engineering of systems. Different meta-models can be adopted to describe different domains or components of the overall system [45]. However even though there are many similar features in different meta-models, there is no standard approach for organisation meta-models. Coutinho and colleagues proposed a high-level common conceptual structure for an organisation model based on existing models and suggested using an organisational interoperability layer to translate between different organisation meta-models [55]. This conceptual model includes functional, structural, interactive, and normative dimensions. Importantly, the idea that agents will adhere to prescribed specifications is not suggested. This is consistent with the aim of enabling some flexibility in agent behaviour, while maintaining social control with norms or policies. They also provided an overview of the approaches adopted in different organisation meta-models for MAS. Our approach is narrower—with a focus on design decisions that may impact flexibility and adaptability in the MAS behaviour at runtime.

A model-driven approach has been used in the disaster management and emergency response domain to define distinct and separate model layers at various levels of abstraction [19,50]. The ALIVE framework defines four layers based on existing systems in a service-oriented architecture: a web-services layer at the bottom, a service layer, a coordination layer, and then an organisation layer at the top level of abstraction [19,50]. Considering the need for flexibility and adaptability in the final solution, designing a solution using different layers of abstraction allows for some autonomy and independence within each layer of abstraction. For example, Schneider and Miller developed

a meta-model for the design, development and specification of human-agent teams in complex systems [56]. This meta-model uses the abstractions of roles and responsibilities to separate agent allocations from goal states to be achieved. Each agent adopts responsibilities that contribute to roles.

Runtime adaptation within a system is valuable and many researchers have investigated software engineering approaches for the development of self-adaptive systems [57], scalable agent organisations [58], and open MAS [59]. Interoperability between models is important to allow for open systems solutions that do not demand homogeneity in agents. Coutinho et al. [55,60] described a tool and process for describing and mapping between different organisational meta-models so that an integrated model view can be created. Providing adaptors between different models would enable an open system of multiple organisations of agents interacting and working together.

Fornara [61] (p. 581) suggests that when choosing an organisation model for a MAS, the following questions should be considered. Can the goals and tasks be divided into independent, formalised, and standardised tasks? If so, how should one approach this best? Which of the tasks and sub-tasks have dependencies that need considering? Can tasks be grouped together, and what are good means to group tasks (function, geographical location, client, process, etc.)? At what level are decisions to be made and controls to be set up? What kind of environment is the organisation located in (open, closed, static, dynamic)? What is the line of reporting in the organisation? Who has authority and what is the chain of command? What rules and formal processes are being required in the organisation? What level of predictability is the organisation to have? In addition, further high-level design considerations may impact the flexibility in agent behaviour at runtime such as the level of agent awareness, the independence of design components, and mechanisms for coordination and communication.

In existing organisation meta-models, mechanisms for agents' awareness of the current organisational structure and their individual responsibilities include using artefacts (e.g., [25,62]), giving agents explicit awareness of the organisational structure as part of shared organisation beliefs (e.g., [63]), and using commitments and conventions [64] to establish mutual beliefs [65]. Artefacts are external to the agents and can be shared amongst agents and used as a tool for sharing knowledge. In contrast, if agents hold individual beliefs without a shared artefact, then agents need to communicate to share their beliefs in order to maintain mutual beliefs within the organisation. This requires that individuals have awareness and beliefs about others in the organisation [66].

Approaches to the functional specification of a problem and alignment of the functionality to agents can be based on a top down design so that agents are allocated to enact organisation roles defined at design time (e.g., OperA+ [28], MOISE+ [67], JaCaMo+ [25], and OMACS [23,68]). Roles are often directly implicitly or explicitly associated with tasks or goals that the system needs to achieve. The specification of system objectives is based on organisational designs specified at design time, with limited scope for agents to behave outside these specifications at runtime. Our requirements demand that we take a different approach in this regard. A related perspective, under which the organisational model is not a structure that simply distributes goals to its agents, but is a way for coordinating responsibility assumption by the agents, is used in the ADOPT framework [69].

It has been argued that the structural dimension of an organisation should be defined separately and independently of the functional dimension so that agents can effectively achieve the organisation's purpose and also reason about each dimension separately [67]. This approach was adopted in MOISE+ [67,70] which also included a deontic specification. In OMACS [23], an organisation can define a separate functional specification of the problem or organisational objectives in terms of a flexible decomposition tree of goals and sub-goals [68]. In the goal tree, multiple pathways can be defined. This introduces some runtime flexibility, as it is possible to select a goal path dynamically at runtime, based on the agents available.

At an organisation level, models can be separated based on how adaptable the organisation structure is and whether the organisation is capable of restructuring at runtime. If organisation structural change is possible, are agents responsible for this or is it controlled externally?

Is reorganisation a structural change to the organisation or a change of allocations of agents to roles or responsibilities?

Tables 1 and 2 position a number of organisation meta-models in terms of the approaches used in each. The coordination approaches are categorised dependent upon how coordination is specified and how much flexibility is afforded to agents. The coordination can be specified in plans written at design time or using roles that define processes. Other points of distinction between models shown in Table 2 include whether functional goals or states are specified distinctly and separately from roles; whether roles define agent capabilities; how roles are described in terms of responsibilities, outputs, and procedures; and whether agents can mutually adjust their behaviour with some autonomy at run time.

Table 1. Comparing meta-models in relation to coordination.

Coordination Mechanism					
Meta-Model	**Design Time Coordination in Plans**	**Role Based Coordination**	**Tasks/Goals Defined Independent of Roles**	**Roles Define Agent Capabilities**	**Agents Adjust Plans at Runtime**
OMACS [23]	✓	✓	✓		
JaCaMo [24], JaCaMo+ [25]		✓	✓	✓	
MOISEinst [26]		✓	✓	✓	
OperA [27], OperA+ [28]	✓	✓	✓	✓	
SharedPlans [29]					✓
OJAzzIC [21]	✓	✓	✓	✓	✓

Table 2. Comparing meta-models regarding organisation structure.

Organisation Structure					
Meta-Model	**Explicit Organisation Structure**	**Agents Explicitly Share Intentions or Outcomes**	**Structural Change to Organisation Possible at Runtime**	**Select a Predefined Structure at Runtime**	**Fixed Structure, Flexible Allocation Process**
OMACS [23]	✓				✓
JaCaMo [24], JaCaMo+ [25]	✓			✓	
MOISEinst [26]	✓				✓
OperA [27], OperA+ [28]	✓			✓	
SharedPlans [29]		✓			
OJAzzIC [21]	✓	✓	✓	✓	✓

The following brief analysis of closely-related organisation meta-models highlights their features and also their limitations with particular reference to requirements for flexibility in agent behaviour at runtime.

2.2.1. Comparison of Existing Organisation Meta-Models and Approaches to Coordination

Agent Stories

The behavioural requirements for agents in the rescue scenario in Section 1.2 are captured using agent stories expressed in the first person in a similar way to user stories that describe requirements. Below, select organisation meta-models for MAS are described and compared with reference to the requirements expressed via these stories.

1. Adopt goals and work with others: *As an agent with capabilities to act to change my environment, I wish to adopt goals that I am capable of achieving so that I support the organisational goals and resolve a problem. If I am capable of achieving part of a designed high-level goal, I am happy to do so in collaboration*

with other agents who can achieve the necessary complementary required sub-goals in order to achieve the mission or organisational goal.

2. Be aware of others and mutually adjust to fit in with others: *As an agent, I wish to be aware of other agents working in the space and able to fit in with others so that we can dynamically coordinate to work together to resolve a problem. This may involve planning a solution, agreeing on roles, and then executing the plan; or partially planning a solution and then adjusting our goals to fit in with others as the detailed plan unfolds. For example, if I as a medic agent have agreed to help with a stretcher rescue with another medic agent with the initial plan that I will meet my partner with the stretcher, but then my partner arrives first, I will adjust my own goals to fit in with the situation of my partner already having begun to put the patient on the stretcher; then I will instead adopt the support role, carrying the stretcher. I will be sure to communicate my intentions with other agents to achieve coordination where necessary*.
3. Prioritise organisational goals: *As an agent, I wish to act autonomously to adopt goals according to my own priorities, whilst giving priority to the organisational goals so that these are achieved*.
4. Rescue injured: *As a medic agent, I wish to find and rescue the injured to move them to the ambulance drop zone so that the rescue is complete*.
5. Coordinate to perform stretcher rescue: *As a medic agent, if I find a seriously injured patient, I wish to work together with another medic agent so that we coordinate a stretcher rescue to carry the injured patient to the ambulance drop zone. If another medic agent is not available (even if we previously agreed to do the rescue together), I am happy to request assistance with carrying the stretcher from another capable agent nearby so that the rescue can be completed in a timely way*.
6. Perform tasks autonomously: *As an officer agent, I wish to clear the area of bystanders so that the medic agents can proceed with search and rescue tasks*.
7. Help others if I can: *As an officer agent, if I am capable of carrying a stretcher; I am willing to help medic agents in this way if asked, even though this is not part of my default role responsibilities, so that the rescue can be completed in a timely way*.
8. Share information about my intentions and actions: *As a collaborating agent, I will share information with* appropriate *other agents to inform them of my intentions and actions performed so that we can all work together efficiently*.

Selected organisation meta-models and frameworks for social coordination are examined in Table 3 with particular reference to how they address the requirements highlighted in these use case scenarios. These were each chosen because they address some element of agent adaptability in a dynamic situation. These models provide a representation of a number of different approaches. These include dynamic allocation of agents to tasks, using policies to guide agent behaviour, using artefacts and commitments to define interactions, using contracts to define responsibilities, and creating dynamic plans between agents. The JaCaMo+ framework includes a number of different underlying models and is broader than one meta-model. It is included because it also addresses some of the requirements. JaCaMo+ uses the MOISE+ [70] organisation meta-model.

Table 3. Agent user story requirements addressed by each approach.

Meta-Model	Agent User Story Requirements Addressed							
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
OMACS [23]	●	◐	●	●	◐	●	○	◐
JacaMo[24], JaCaMo+ [25]	●	◐	●	●	●	●	○	○
Moise [26,70]	●	◐	●	●	●	●	○	○
OperA [27], OperA+ [28]	●	◐	●	●	◐	●	○	◐
SharedPlans[29]	●	●	◐	●	●	●	◐	◐
OJAzzIC [21]	●	●	●	●	●	●	●	●

Legend: ●: yes, it can support the requirement; ◐: partially, by hard-coding into roles or action scripts; ○: no.

Challenges for each meta-model include how to establish agent awareness and how to enable agents to coordinate their individual behaviour at runtime to achieve the completion of tasks that require multiple agents to work together (without relying on detailed scripts defined at design time). JaCaMo+ and OperA+ provide features based on social contracts between agents and separate agent internals from an interaction layer that is specified externally. The interaction layer is used to specify protocols to direct agents in their runtime behaviour. In JaCaMo+, agents use explicit commitment-based interactions, based on the work of Singh [71]; however, in JaCaMo+, these commitments are represented in artefacts external to the agents. The interaction artefact in JaCaMo+ is referenced based on protocols specified at design time and agents rely on the system to notify them rather than receiving direct communication from another agent.

In JaCaMo+, it is not possible for agents to negotiate and create a coordinated plan at runtime in order to reach a goal involving interdependent activities. SharedPlans does provide for dynamic coordination of agent plans to enable dynamic planning. This level of coordination is not provided in the other meta-models under examination. SharedPlans incorporates an explicit level of interaction around intentions and sharing of intentions; however, SharedPlans does not incorporate any organisational knowledge.

OperA+ and OMACS both use capabilities as a way of allocating agents to roles, but both assume that it will always be possible to find an agent possessing all the capabilities required to enact a role fully. Improvisation outside of a role description is not a feature of any of these models.

It is common in organisation meta-models to find that the organisational objectives are described implicitly in the functional description of the system using roles. Effectively, if every role is fulfilled, the organisational objectives will be reached (e.g., [28,52,72]) so that within a group or organisational context, agents are associated with a role to determine the activities in which the agent may participate. In more flexible approaches, based on the particular context, roles may be selected or assigned to agents at runtime in order to gain some flexibility (e.g., [73]). In these cases, the organisational structure is derived to suit the situation. It is possible to achieve flexibility in both organisational roles adopted and in the way goals are decomposed so that potentially, multiple solutions can exist. Existing meta-models have much to offer, but do not provide for sufficient agent awareness and flexibility to addresses all our requirements. From the agent stories, we see the organisation meta-model should enable that:

- Organisation agents can identify relevant other agents;
- Policies or norms specify obligations for sharing beliefs and intentions;
- The organisation is a first class entity with agents having awareness of organisation structure and membership;
- Agents can interact explicitly regarding intentions;
- Tasks can be aligned with the capabilities required to perform them;
- Agents can mutually adjust their individual behaviour and coordinate runtime behaviour to agree on plans;
- Agents reason with consideration of organisational objectives;
- Agents are not unnecessarily constrained by roles and can act based on their capabilities;
- Goals can be broken down into related tasks, so that agents can individually adopt tasks to contribute to the achievement of goals.

3. OJAzzIC Organisation Meta-Model

In this section, we describe the key components in the OJAzzIC meta-model, and describe an adaptation of O-MaSE [30], an organisation-based multi-agent software engineering methodology, applicable to OJAzzIC.

3.1. OJAzzIC Overview

The OJAzzIC meta-model as described conceptually elsewhere [21,22,74] includes a specification of agents in an organisation, an organisational structure, and a set of behavioural policies and guidelines. Policies specify how multiple organisations can be created to form a distributed, coordinated system of interconnected agents and organisations and define explicit agent obligations. Additionally, the OJAzzIC meta-model includes a multi-layered functional goal model that describes goals and sub-goals and tasks that fulfil the achievement of goals. A role model specifies how goals and roles can be aligned. Flexibility in the role model includes consideration of the relationships between roles and constraints regarding agents' adopting roles.

The organisation is used to define contextual boundaries for coordinating groups of agents and making relationships between agents explicit. We achieve this in the structural dimension of an organisation. Organisation roles are defined; however, agents are defined with individual capabilities. The mapping between the structural and functional dimension (how agents adopt or are allocated tasks) allows for some flexibility.

Each OJAzzIC organisation instance includes a number of agents who are individually aware of the organisation, the organisation's policies, and their role responsibilities within the organisation. Agents are also aware of their individual capabilities and have a set of plans that outline how they can use their capabilities in order to achieve particular tasks or goals. Agents individually possess beliefs and use knowledge that is initialised in the agent based on the agent specification.

Within an organisation, agents can interact and create plans to act together. Each organisation can be used to identify the group of agents who are interested in particular information or share goals. As agents can belong to multiple organisations simultaneously, this allows relevant beliefs about the situation and agents' intentions to be distributed and shared amongst and between organisations. As the specification of agent action plans are broken into individual plans for smaller tasks, it is possible to enable agents to adjust their behaviour when engaging with others in coordinated activity, so that if necessary some mutual adjustment takes place at a lower level. In addition to adopting roles, agents may be given permission to adopt tasks based on their capabilities if necessary to reach a desired solution state. At design time, a particular goal state can be decomposed into sub-goals or tasks that lead toward reaching that state. At runtime, agents can select the appropriate pathway of tasks or goals that can successfully be achieved to lead to reaching the final goal.

When agents need to work together, if detailed scripts for coordination are not provided at design time, then agents need a runtime mechanism for negotiating and agreeing upon coordinated behaviour at a task level. SharedPlans [75] are used to achieve coordinated behaviour by agents. The negotiation of a SharedPlan at runtime demands that agents agree at a high level that they have an agreed mutual shared goal and also mutual shared intentions to reach that goal. SharedPlans are created based on explicit commitments between agents at runtime so that agents can agree to adopt a goal, commit to goals that contribute toward an incomplete plan, and revise plans by changing adopted goals if the context changes.

Policies ensure that appropriate commitments are made to ensure that agents share beliefs relating to the plan and their intentions. Policies are also used to specify the level of sharing of general belief and intention sharing that is appropriate within an organisation. At design time, it is necessary to identify the depth of sharing that would be helpful to agents. Policies could be implemented using a normative specification.

Using policies gives agents runtime flexibility but also demands that at design time, appropriate policies are defined. In OJAzzIC, policies written at design time provide mechanisms to coordinate agents working together on interdependent tasks, to coordinate sharing of information, including beliefs and intentions, to create runtime organisations if necessary and to provide guidelines for agents regarding improvisation [22].

The OJAzzIC meta-model has some similarity with MOISE+ and JaCaMo+. The most significant difference with the MOISE+ meta-model is the introduction of flexibility in terms of the role model

in OJAzzIC and introducing explicit coordination compared with MOISE+ that has a reliance on embedded coordination in missions. The use of explicit commitments in OJAzzIC is similar to JaCaMo+ and similar to contracts in OperA+. However, in OJAzzIC, commitments are directed interactions between agents and not stored in an external artefact. Commitments and knowledge of the organisation structure make it possible for agents within the organisation to have some awareness of the intentions and beliefs of others. Importantly, agents in OJAzzIC can adopt individual intentions to reach a goal based on capabilities outside of their allocated role(s). This is a distinguishing difference between OJAzzIC and other meta-models and provides flexibility at runtime that is not available in other approaches.

3.2. OJAzzIC Meta-Model Components

In this section, we present a definition of the OJAzzIC organisation meta-model [21]. In OJAzzIC, an organisation O is a tuple representing a first class entity created at runtime, $\mathtt{O} :< A, C, G, R, Re, Contract, P, \sum >$ although some of the components of the organisation may be specified at design time to populate the organisation instance created at runtime. The core components of the organisation are:

A: Set of Agents.
C: Set of capabilities.
G: Goal Tree, including ordering tasks where necessary. This defines the organisational goals and how they could be decomposed into sub goals and ordered tasks; a task is a leaf goal that cannot be further decomposed. A task may use a resource and can be achieved by an agent who possesses the capability associated with fulfilling that task [76].
R: The role model including a set of domain roles, relationships between roles, and context-based role definitions. This may also include coordination roles. Role relationships define authority between roles.
Re: A dynamic resources list defining objects in the environment that can be used to help perform tasks.
Contract: The contract contains a *social contract* and an *information contract*. The social contract comprises SharedPlans created by agents; a SharedPlan outlines the current selection of tasks to achieve a goal and the explicit agreement and commitments between agents thus far assigning responsibilities for tasks to individual agents. The information contract is a set of agreed policy obligations and commitments to intentions to ensure consistency of beliefs within the agent organisation. The information contract is responsible for the creation of runtime messages between agents regarding β, the set of beliefs about the environment, including resources.
P: Set of policy constraints to apply to all members and to the adoption/allocation of tasks; *authority* policies explicitly define a process for adoption/acceptance of allocations by an agent and *coordination* policies explicitly help agents coordinate multi-agent plans dynamically. These social policies are used in OJAzzIC to specify obligations regarding knowledge cultivation, coordination, and creation of dynamic organisations.
$\sum$: Environmental domain meta-model used to specify environment objects and relationships.

The domain model and resources available in the environment, external to agents, are domain specific. It is necessary for an interface to exist between the agent and the environment as part of the environmental meta-model so that agents can receive percepts from the environment. Agents need to have a model of the environment in order to understand and assess what goals are valid to pursue. Aside from mentioning that there needs to be a model of the environment internally for agents and the interface between agents and their environment, we do not specify anything further about the environmental meta-model.

3.3. OJAzzIC Runtime Execution Model

Agents start by belonging to a large organisation responsible for the main high-level goal (e.g., the entire system). When two or more agents need to coordinate, if they are not already in an organisation, then a new organisation is formed and within that organisation an explicit contract dictates policies, obligations, agreed goals, and agreement to coordinate with others in that organisation. Default policies can be adopted, or specific policies designed for particular types of organisation can be specified at design time. Coordination within the organisation relies on appropriate communication to share relevant information and share plans with others in the organisation. The new organisations that form overlap with existing organisations, so that agents can belong to multiple organisations. In one organisation, multiple smaller organisations are created. These organisations each need to be explicit so that appropriate coordination can be established within each. Each organisation instance created is based on the OJAzzIC organisation meta-model structure.

In the OJAzzIC deliberation cycle, an agent perceives environmental input and updates individual mental attitudes, but, before selecting an intention, the organisational agent will deliberate with others in the organisation. This organisational deliberation is defined by obligations and policies in the social contract within the organisation and includes updating others and maintaining consistent beliefs within the organisation. Following the first stage of organisational deliberation and any needed adjustment to individual attitudes, the agent continues the deliberation process considering others—when the agent may revise individual intentions and plans to ensure that they are not intending anything that will hamper others and possibly to add new intentions to help others. This deliberation is based on the BDI interpreter cycle [77] depicted in Figure 1, and as elaborated in Figure 2.

```
1  intialize-state();
2  repeat
3    % achieved in the event module in the simulation system
4    options := option-generator(event-queue);
5    selected-options := deliberate(options);
6    update-intentions(selected-options);
7    % performed in the main module
8    execute();
9    % achieved in the event module
10   get-new-external-events();
11   drop-successful-attitudes(); %ie drop goals already achieved
12   drop-impossible-attitudes(); %ie drop goals if cannot be achieved
```

Figure 1. BDI interpreter cycle [77].

OJAzzIC organisation agents are autonomous but the organisation structure provides policies that agents abide by. An agent may enact one or more roles defined within the organisation. An agent also possesses individual capabilities, including the ability to communicate directly with other agents. An agent will have an ability to negotiate with other agents dynamically in order to agree on a SharedPlan wherein multiple agents need to coordinate their behaviour. Each agent will create an individual plan that is consistent with other agents' plans so that each individual's behaviour is coordinated.

The basic logic behind an organisation-aware agent processing individual and organisational attitudes is: "If there is a objective for the organisation that is valid in the current context (scene) for a role that I am enacting, then I should adopt the consideration of this objective." Agents adopt goals to perform based on prioritising objectives under consideration.

The designer can anticipate a number of organisation(s) as part of the design process and allocate agents as members of long term organisations. It is also possible that agents may need to form an organisation at runtime. In the next section, we describe the methodology adopted to design a MAS.

```
1  Consider individual objectives (based on existing individual goals).
2  Consider organisation objectives and if one of the following holds
3  adopt an active landmark state as an objective:
4        It is an org objective and I am responsible due to my role.
5        I am individually responsible (not in an organisation).
6        I am capable (not in an allocated role).
7        I am capable of achieving part of the objective.
8        I have been asked to help and I have the capability required.
9  Adopt goals based on landmark objectives.
10 Choose actions based on current goals.
11 Send/receive messages to and from other agents.
12 Perceive updates from the environment.
13 Respond to any events to update beliefs and current goals.
```

Figure 2. Reasoning cycle of organisation-aware agents.

3.4. OJAzzIC Design Methodology

In this section, the design methodology, an adaptation of the O-MaSE methodology [30], is briefly outlined for adopting the OJAzzIC model to create a MAS. The design methodology is described in more detail elsewhere [74,78].

In O-MaSE [30], goals are used to define the objectives of the organisation, roles are used to define positions within the organisation that can achieve a given goal or set of goals, and the interactions within an organisation are specified using interaction protocols. A limitation of O-MaSE (based on OMACS [23,68]) is that there is no provision for defining a system capable of splitting roles or sharing role responsibilities amongst a number of agents. This limitation is addressed in the approach used with OJAzzIC. In addition to role specifications, the design process must also include mechanisms for the modification of agent behaviour at runtime. We use social policies to specify agent obligations to enable internal interactions and commitments between agents within the organisation to support coordination.

The OJAzzIC design process [78] involves five main tasks derived from O-MaSE, as outlined in Figure 3. These design tasks result in a specification or refinement of each of the meta-models defined in the OJAzzIC system. Particular points of difference between the OJAzzIC design process and the O-MaSE design process are the inclusion of tasks in the functional goal model in OJAzzIC and the ability for agents to adopt a task or goal based on capabilities outside of a role. Additionally, the use of social policies in OJAzzIC is directed so that agents are guided regarding improvisation and obligated to create commitments at runtime to address coordination needs.

OJAzzIC Design task	**Related tasks in O-MaSE**
Define the functional goal model	Model goals, Refine goals, Model domain, Model plans, Model protocols
Define the organisation model	Model organisational interfaces, Model roles
Define the agent capabilities model	Define roles, Model agent classes, Model capabilities
Define the role model for the organisation	Define role goals
Establish social policies in the coordination model	Define protocols, Model policies

Figure 3. Design tasks in OJAzzIC, and their relationship to the O-MaSE approach.

In OJAzzIC, the design of a problem solution includes 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 organisations. By keeping these distinct, we aim for more flexibility at runtime. In OJAzzIC, a direct relationship between roles responsible for a goal and agents available to adopt roles is not presumed. If there is not a direct match between the goals and the available agents' assigned roles, then the matching of goals to agents can emerge at a lower level based on agents' capabilities and the capabilities required to achieve a task or sub-goal.

In the next section, we provide an overview of a demonstration MAS system created based on the OJAzzIC meta-model and the experimental scenarios used.

4. Demonstration System

4.1. The Simulation Scenario

The scenario introduced in Section 1.2 was implemented using agents interfacing with the block world for the team (BW4T) simulation system [79], thereby controlling robots responsible for searching for and rescuing injured coloured blocks in a number of rooms in the environment. Figure 4 shows the interface of the BW4T map at the beginning of a crisis for two different scenarios. The simulation scenario can be configured to change the number of agents, the number of rooms, and the number of injured blocks requiring rescue. This figure shows on the left a configuration with three *medic* agents, three *officer* agents and eight *bystander* agents. There are four rooms and a total of two orange blocks and two red blocks. The red blocks are in room A1 and room A2. The orange blocks are in room A1 and room B2. The configuration on the right map is more complex. There are five *medic* agents, three *officer* agents, six *bystander* agents, and 12 rooms. Increasing the number of medic agents increases the potential number of agent pairs who could perform a stretcher rescue. There are five orange blocks and five red blocks to be rescued. Orange blocks indicate that the rescue is complex and requires a stretcher (i.e., two rescue agents). Bystander agents are labelled as feye*n* or ajax*n*. To complete a rescue, the robots pick up the blocks and move them to a special drop zone representing the ambulance drop zone.

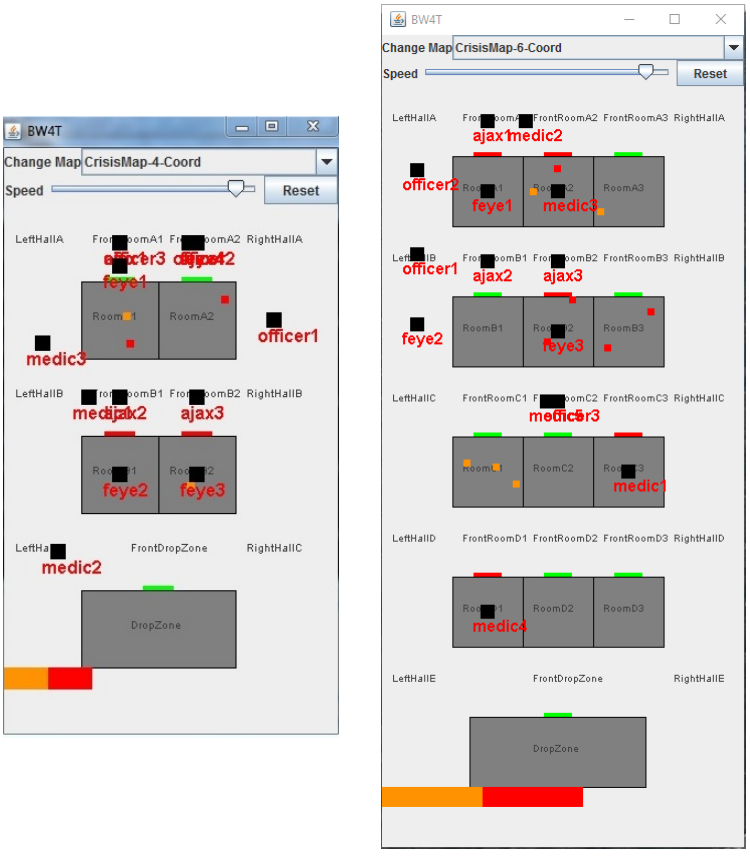

Figure 4. BW4T crisis maps.

In the simulation, the actors are implemented as one of three possible agent types: medic, officer, and bystander. Table 4 specifies the capabilities allocated to each agent and each role in the

organisations created at design time. The default role for an actor named medic*x* is to be in the role of a medic agent within the MedicOrg. As part of this role, the medic agent is given the capability to locate the injured and perform rescues to move the injured to the ambulance drop zone. The officer role, in the OfficerOrg, has the capability to clear bystanders away from the disaster area so that the medics can enter rooms for search and rescue operations. Officer agents were in some cases, given the capability to assist with a stretcher rescue in addition to the default capability set of the officer role. However, as part of the role of an officer, the officer agent is not responsible for stretcher rescues.

Table 4. Roles and capabilities of agents in a rescue scenario.

	Role	**Role Capabilities**	**Organisation**
	medic	locate injured, perform rescue, perform stretcher rescue	MedicOrg
	officer	clear bystanders	OfficerOrg
Actor	**Role**	**Additional Individual Actor Capabilities**	**Organisation**
medic1	medic	none	MedicOrg
medic2	medic	none	MedicOrg
medic3	medic	none	MedicOrg
officer1	officer	perform stretcher rescue	OfficerOrg
officer2	officer	perform stretcher rescue	OfficerOrg
officer3	officer	perform stretcher rescue	OfficerOrg

In our rescue responder system, we deliberately specify roles at a high level and do not specify detailed plans for joint tasks. For example, two agents involved in a joint stretcher rescue must at runtime negotiate which particular role they will adopt when enacting a shared rescue together. One agent must adopt the role of support (waiting at the door and then "carrying the stretcher") whilst the collaborating partner agent adopts the role of picking up and holding the injured block. This runtime behaviour requires that the agents mutually adjust to adopt the correct role to fit in with their partner agent. The stretcher rescue task involves two agents first agreeing that they have a mutual SharedPlan to adopt that rescue task, and then both dynamically enacting (and perhaps adjusting) plans to ensure the rescue is completed. The task requires that the two agents communicate and collaborate to complete the joint task.

The process of deciding who will perform a rescue involves communication between the rescue agents. First one agent proposes to another that they might consider collaborating on a rescue. As each medic agent is behaving autonomously, there is no control over which agent may propose a rescue. An agent may instigate a rescue proposal or may receive a rescue proposal from another agent. When two agents agree that they both have the same potential rescue in mind, then they agree that this potential intention can become a mutual intention: that the rescue be completed together. There is also a third possibility which enables an agent to directly invite another agent to accept and create a mutual intention—that regarding a rescue. When agents have an agreed mutual intention that they work together on a rescue, they then create a SharedPlan to enact a rescue.

4.2. The Agents

The agents in our rescue response system were implemented using the GOAL programming language [80]. GOAL is a high-level language based on the BDI paradigm that enables agents to make rational choices about the goals they will adopt based upon their mental attitudes, such as beliefs and knowledge about the world. The GOAL language provides an interface to interact with BW4T and enables the agents to communicate with each other by sending and receiving messages. GOAL agents follow a blind commitment strategy, so they commit to adopted goals until they are achieved or until new beliefs change the agent's decision to adopt the goal. This may occur if the conditions leading to the adoption of the goal have changed, which may happen, for example, if the agent perceives changes in the environment or if another agent informs this agent that the goal has

already been reached. Agents may have a number of possible action choices at any time based upon their current beliefs and goals. The goal list for an agent specifies what the agent wants to achieve at a point in time and is similar to a desire in the BDI framework [9]. When an agent selects a particular goal to adopt, then that goal describes a state that the agent intends to achieve. Agents choose actions from the provided action rules based on their current goals. In our simulation system, if a medic agent updated a belief (for example, no longer believed that there was a particular block needing rescuing in a room) and if that agent had an existing goal adopted to perform that rescue, then the updated belief led the agent to drop the existing rescue goal. Additionally, if an agent was informed by another agent of an intention to perform a particular rescue, then the agent would drop any intention to perform that rescue; that has the effect of loosening the blind commitment strategy to behave more like an "open-minded commitment" [77]. Ideally, rational agents should be able to reassess existing goal commitments when a conflicting higher priority goal exists [81]. In our simulation, that was not implemented. When an agent began a stretcher rescue, for example, unless the rescue partner agent sent an explicit message to drop the shared plan, or beliefs changed to make that goal no longer valid, the agent blindly committed to completing the rescue.

The reasoning cycle of our organisation-aware agents is outlined in Figure 2. The order in which goals to consider are adopted is based on considering organisational goals first, and then if there are no organisational goals based on role, then consider goals to adopt based on capability. The agent will always perform the action generated by the first rule possible, so as soon as a goal is adopted, this module is completed and the agent cycle continues, so by placing the highest priority rule first, agents will always select the highest priority goals for consideration.

The reasoning cycle for agents in the implemented MAS is based on the GOAL system [80], with the addition of an organizationalEvents module

1. The event module (containing event or percept action rules) is run to update the agent's mental state (effectively this module handles events and messages); after the perceptual events are processed, the last step in the event module is to call the organizationalEvents module.
2. The organizationalEvents module is responsible for taking action that is immediate, sending organisational messages based on policies, updating beliefs and goals based on messages received, revising goals that are outdated, and negotiating with other agent regarding SharedPlans.
3. The main module is entered and rules in that module are applied to define action selection, so the agent selects an action using the action rules (based on existing beliefs and goals). The action rules are evaluated in linear order (top–bottom); the first valid rule is selected and applied. By placing the rules in priority order in the main module, the agent will choose the highest priority valid selection (matching preconditions on the rule). If there are no existing individual goals, the agent will consider adopting organisational goals (identified in the organizationalEvents module). If there are no organisational goals for consideration at that point, the agent will then call the orgReasoning module.

As shown in Figure 5, the agents in our simulation follow a slightly modified version of the BDI interpreter cycle. The execute step is achieved in the main GOAL program. The first three steps in the repeat loop (lines 4–6 of Figure 1) are achieved in the event module in the simulation system code. There are also some prioritised actions and decisions to drop outdated goals embedded in between deliberation in the event module.

When an agent is capable of achieving part of an objective, or agents outside a formal organisation structure agree to work together to achieve an objective, this requires coordination. Agents can form SharedPlans to agree on how they will work together. Additionally, to ensure that appropriate information is shared between the relevant agents, then the agents can create an adhocracy (a runtime ad hoc organisation). An example of a situation requiring an adhocracy is when a medic agent hands over responsibility to an officer agent to carry a stretcher.

```
repeat
  perform event module:
      run all percept action rules
      update agent beliefs and goals based on new input
  perform organisational events module
      send and receive messages
      react to received messages by updating mental state (beliefs and  goals)
  perform main module (only fire first valid rule)
      select next action to perform, based on active goals
      if no existing individual goals, consider organisational goals
      if no organisational goals considered yet, perform organisation reasoning
end repeat
```

Figure 5. Agent reasoning cycle in implemented system.

In order to present agents with opportunities for improvisation, in some cases, following the creation of a SharedPlan to perform a coordinated stretcher rescue, a medic agent could decide to handover the responsibility for the stretcher-carrying task to an officer agent. This involves delegation with a handover message so that the SharedPlan agreement is transferred to the new officer partner. The handover requires a sharing of beliefs and plans relating to the rescue between the medic and the officer. As the officer is not part of the default medic organisation, if a handover was completed without also creating a new adhocracy, issues would arise in terms of communication of beliefs regarding the completion of the objective. The agents completing the rescue may not know to inform the original medic that the objective has been reached successfully. This was addressed by creating an adhocracy involving the officer and medic agents involved in the rescue to ensure appropriate sharing of beliefs. Without this adhocracy, the alternative would be to specify the stretcher rescue plan with more detail to include explicit rules in the agent plan to specify that rescue agents will inform others (and explicitly state which others) when the rescue has successfully completed.

Simulation scenarios were created with different levels of organisational support and varying problem complexity. Internal parameters such as policies adopted in the simulation system were varied to change the level of organisational support given to agents. To vary problem complexity, the number of agents/rooms/injured were changed. For each setting, the simulation was run a number of times, to observe the behaviour of agents in the MAS system. This produced results that demonstrated the impacts these parameters and inputs had on the agents' behaviour.

The following list describes the policies implemented in the system:

- spolicyA specifies that an officer agent will be obligated to commit to a medic agent if requested to help with a task that the officer is capable of achieving.
- spolicyB specifies that any agent in an organisation is obligated to share new beliefs with other agents in that organisation.
- spolicyC specifies that when an agent is in an organisation and creates an commitment to achieve a particular objective in that organisation, then that agent is obligated to inform others in the organisation of their intention.
- spolicyD obligates an agent, agent1 in an organisation to commit to another agent in the organisation who requests help with a particular objective; agent1 is enacting a role responsible for the objective.
- spolicyE states that when an agent has an existing rescue commitment in an organisation and is busy, then that agent is obligated to inform other agents in that organisation that it is currently busy. This is implemented by creating SharedPlans to make commitments explicit between agents.
- spolicyF is a policy to create a new adhocracy, (a new ad hoc organisation) if there are two agents with a shared goal who are not both members of an existing common organisation.

4.3. Implemented Scenarios

The key lessons of these experiments are captured in the summary of simulations presented below; further details are available in [78]. The first simulation involved medic agents searching four rooms for four injured parties. This simulation was run multiple times with three medic agents and then four medic agents in order to validate the value of using the organisation structure. The second simulation involved five medic agents searching 12 rooms for 10 injured parties. In each case, half of the injured were significantly injured and so required stretcher rescues. Stretcher rescues demand that two agents coordinate their intentions and actions. Each agent adopts a distinct role in the stretcher rescue: one agent collects the (virtual) stretcher and puts the injured onto the stretcher; the other agent is the helper agent and waits at the door of the room until the injured is "picked up" by the other agent, before both agents travel to the drop zone together and the injured block is placed in the drop zone.

Across all scenarios, situations were created to test the ability of our agents to: use initiative in adopting a task based on capability not role responsibilities; share and create mutual knowledge of situation and plans; and mutually adjust individual plans to fit in with others. We used situations wherein improvisation was possible based on defining policies to adopt tasks based on capabilities possessed rather than role adopted. Additionally, policies were implemented for situations in which agents at runtime decided to work together outside of an organisation. In those situations, agents created an adhocracy to ensure appropriate coordination.

Tables 5 and 6 show the different organisation level policies that were applied. Each setting is given a numeric category, shown as *Org index* in the table. The configurations in Table 5 involved varying the level of organisational support by toggling on/off different policies and changing organisation membership. The range of organisational support included no organisation through to full organisational support, including policies for sharing beliefs and intentions within the organisation. The configurations in Table 6 involved varying the policies and allowing improvisation outside of roles, enabling an agent: to handover (delegate) an agreed joint task to another agent; and to find a new partner if the original partner was unavailable. In the simulation, when these policies were active, permission was given to a medic agent to use initiative to handover the completion of a particular rescue to a capable officer agent.

Table 5. Organisation support level policy settings—4-room simulations.

Org Index	Policy
0	no organisations, no policy support
1	organisation exists, but no policies for sharing
2	organisation exists, partial policies for sharing rescue completion only
3	one capable medic agent not allocated role of medic and not part of the medic organisation, but capable of rescue. This agent can adopt based on capabilities, but is excluded from some sharing of beliefs within the organisation
4	full organisational support with all medic agents belonging to the one medicOrg organisation with policies turned on for sharing beliefs and sharing completion status when rescues are complete
5	medic agents are in an organisation. One of the medic agents when performing a stretcher rescue will handover the stretcher-carrying task by delegation to an officer agent

A higher organisation index value indicates that the agents were given more organisational support with more policies to support the organisation. Configurations with index values below 4 had some changes made to the organisation membership or to remove policies for sharing of beliefs, intentions, or rescue outcomes. All configurations with index values 4 or above had organisations for medics and officers and policies for sharing beliefs, rescue intentions, and rescue completion. Configurations 5–9 had full organisation support with various policies for handover toggled on or off to observe the impact.

Table 6. Organisation handover policy settings—12-room simulations.

Org Index	Policy: Full Organisation Support, with Agents Sharing Beliefs and Rescue Completion Plus
6	no additional policies
7	handover policy allows medic to delegate if an officer is nearby
8	handover policy allows medic to delegate stretcher carrying, if any officer is available, the officer then remains unavailable for any further rescues; share rescue intentions with all others in the organisation
9	the stretcher handover policy allows handover by a medic only in response to a busy message sent by the original stretcher-carrying partner, full sharing of beliefs and intentions

5. Results

Agent performance was measured in terms of time and the number of rooms entered. The best performance was observed when the agents were provided the full features of the OJAzzIC meta-model, implementing policies for sharing intentions and beliefs regarding rescues to be completed and the successful completion of rescues.

Figure 6 provides a summary of a number of 4-room simulation runs involving coordination of a stretcher rescue with the policies in Table 5. This plots time vs. level of organisation support vs. number of rooms entered. Green results are for simulation runs involving three medic agents; blue results are for simulation runs involving four medic agents.

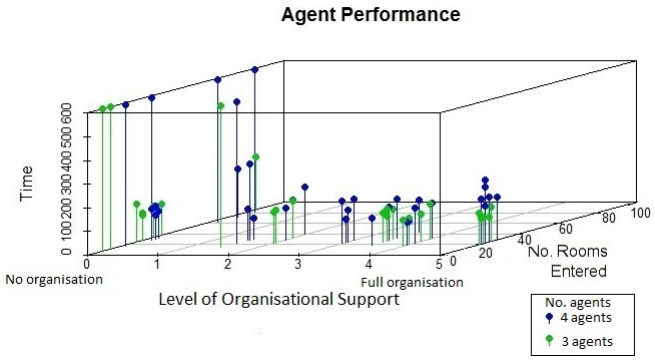

Figure 6. Agent performance in 4-room search and rescue.

The data illustrates that the agents with no organisation awareness were not as successful as those with organisation support. The successful runs occurred with Org index level 2 and up. These successful simulations involved organisations in which agents were communicating. The time taken to complete all rescues was similar across these simulations. Generally small time improvements were observed as more beliefs were shared within the organisation. The highest level of organisational support in this group of simulations was for Org index 4 and 5. Org index 5 simulations had better performance overall regarding the number of rooms entered during the simulation.

Exploring the policies in Table 6 provides an opportunity to investigate more closely the social policies for improvisation within an organisation to scrutinise whether the trade off between the extra overhead of communication driven by organisational policies is beneficial for improving performance. Figures 7–10 show the number of messages sent, number of rooms entered, time taken, and time spent idle by agents in simulations with these configurations.

Observations of agents' behaviour during the simulations show that performance is more efficient in the achievement of the objectives when agents belong to an organisation and have the support of organisational policies to ensure coordination of beliefs and intentions. When a handover to an officer to carry a stretcher was permitted in response to a situation wherein a medic agent suddenly became unavailable and could not fulfil a previously created SharedPlan, performance improvements were observed.

The results at Org index 7 and 8 indicate that there is a time and communication cost involved in enabling agents the flexible runtime behaviour of handover when compared to organisational agents who do not handover. However the overall performance in terms of time was improved with the introduction of policies that allowed agents to adapt to the situation at runtime. When agents are informed of the rescue intentions of others, they can respond to adjust their own intentions. This improves time and number of rooms entered; however, it does increase the number of messages sent. The ability to adjust social policies at design time to enable agents at runtime to adapt is a trade-off between improved performance and the relative cost of activities such as entering a room and sending a message. Additionally, the effort required at design time must be considered.

These results show that performances of agents can be positively improved by introducing policies. The policies used in Org index 8 and 9 seem to have positive impacts on the idle time of medic agents and the efficiency in terms of the total number of rooms entered by agents. There is some variability in different runs regarding time taken to complete the simulation; however, it appears that generally, the time taken improves when policies for handover are introduced.

In the simulation runs involving five stretcher rescues, when polices for medic agents were specified to enable handover in response to a busy message (Org index 9), medic agents took advantage of this option for at least two out of five stretcher rescues in all cases. In 80% of those simulations, an individual medic agent chose to delegate to an officer in three of the five stretcher rescues. The average idle time of medic agents was reduced in those simulations compared to other simulations when this policy was not enabled. Additionally, when agents were programmed to automatically handover to a nearby officer if an officer was nearby (index 7), a similar number of handovers occurred; however, the average idle time of agents was higher (314.38 compared to 282.54), as shown in Table 7.

Table 7. Comparison of agent idle times.

Org Index	Average Idle Time	Average Number of Handovers
6	289.49	0
7	314.38	2.8
8	285.41	3
9	282.54	2.75

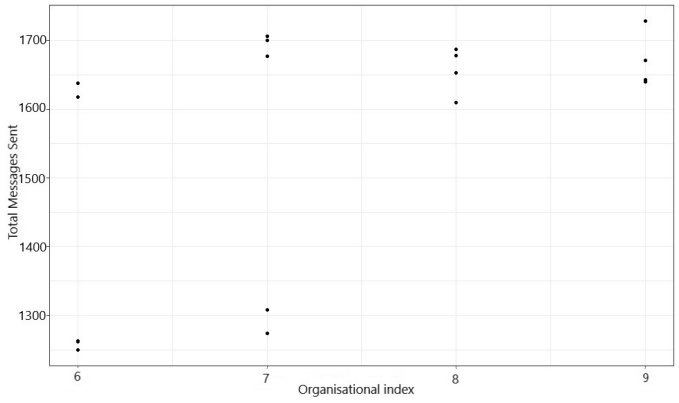

Figure 7. Total messages sent vs. Org index for 12-room simulations.

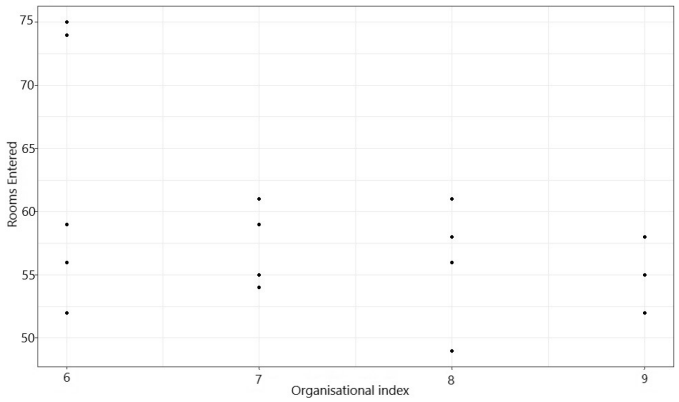

Figure 8. Rooms entered vs. Org index for 12-room simulations.

The data indicate the number of rooms entered decreases with more organisational support and flexibility in terms of delegation of the stretcher-carrying task. The optimum performance was with level 9, when agents had the flexibility to respond to a potential delay and would only delegate in response to a busy message. When there was no organisation existing for the medic agents, the number of rooms entered was much higher, as was the time taken to complete the rescue.

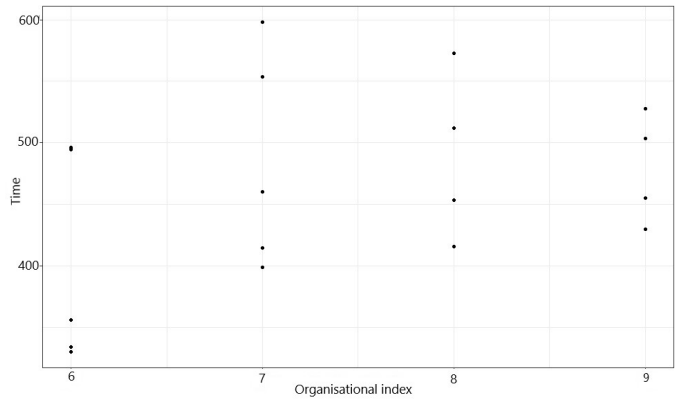

Figure 9. Time taken to complete rescue vs. Org index for 12-room simulations.

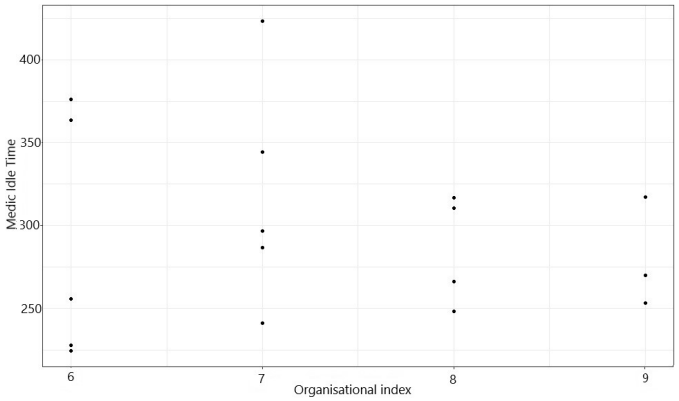

Figure 10. Medic idle time vs. Org index for 12-room simulations.

Because the adoption of tasks in OJAzzIC are not limited to agents specified in organisation roles at design time, agents in the simulation could use initiative to adopt a task to complete that was outside of their allocated role (if they had the capability and permission to do so). This was demonstrated in simulation levels 7, 8, and 9 by a design time decision to give an officer agent the capability to carry a

stretcher and enabling policies allowing opportunities for a medic officer to handover or delegate to an officer to assist with carrying a stretcher.

The best time performance occurred when there was full organisational support but no stretcher handover at all (level 6). These results support the idea that when the design time plan is available, it should be adopted. However if a solution cannot be found, although improvisation may increase time taken, a solution can be found by allowing agents this flexibility.

The behaviour of the MAS model described provides demonstration of the value in allowing agents to improvise. The idle officer agent was able to step in and help a medic agent to complete a stretcher rescue when asked. There were cases wherein a medic agent was stalled during a stretcher rescue, waiting for a potential partner (with whom an agreement had been made to complete that rescue) because the carrying partner was busy completing another rescue. When a nearby officer agent was instead delegated to take over that stretcher-carrying task, the time taken improved to produce the best overall performance. Using other meta-models, the officer role would need to be expanded to include the stretcher-carrying responsibility to enable an officer to complete a stretcher rescue in the same way.

6. Discussion

In this section, benefits provided in the OJAzzIC agent organisation meta-model are highlighted based on the behaviour observed in the demonstration MAS system. Three key features are identified as providing the designed runtime flexibility—specifying functional solution in terms of tasks and sub-goals that could align with agent capabilities; using policies to make explicit the runtime commitments that agents should adopt to share beliefs and intentions, along with improvisation guidelines; and adopting a runtime coordination model using explicit runtime commitments between agents.

Using capabilities as an explicit abstraction provided flexibility for agent adaptation at runtime. Individual agents could identify sub-goals or even tasks that lead toward reaching an organisational goal with some flexibility at runtime. It was not necessary to identify which agent was to be allocated to a particular task or goal, but rather, agents could individually adopt goals with some autonomy. OJAzzIC matches capabilities to tasks to enables agents to adopt tasks and coordinate intentions with others. This requires effort by the designer to identify capabilities and design agent code that enables agents to coordinate operational details and create a SharedPlan, but provides more flexibility at run time.

OJAzzIC policies guided agents to share beliefs and plans so that they had awareness within the organisation. Having this awareness, agents could adjust their individual intentions so that they did not compete with others and were more efficient. The use of explicit policies specified by the designer ensures the knowledge sharing obligations are clear. Using explicit commitments and SharedPlans enabled agents to communicate to coordinate their individual behaviour for joint rescue tasks at runtime with whichever other agent was available. The operational details were also determined at runtime using commitments. It was not necessary at design time, to specify which agent would perform each rescue, nor which agent would adopt each particular role in the stretcher-carrying task.

In other organisation meta-models, agents rely on planned coordination that is built into the agent's plans, scene scripts, protocols, or coordination artefacts that are specified at design time. As the coordination is effectively specified at design time, the agents do not have flexibility at runtime to adjust if the context does not match that anticipated at design time. By contrast, agents in OJAzzIC can adopt individual intentions to reach a goal based on capabilities outside of their allocated role(s). This is a distinguishing difference between OJAzzIC and other meta-models and provides flexibility at runtime that is not available in other approaches.

Enabling agents in our simulation to coordinate dynamically at runtime using explicit direct communication between agents was shown to benefit agent performance. For example, available rescue agents at runtime could negotiate a SharedPlan for a stretcher rescue and agree on low level

coordination rather than needing detailed coordination specifications in advance. In addition, agents could draw on policies to allow for improvisation in handing over the stretcher-carrying task to other capable agents. For example, in some scenarios policies were specified so that medic agents could use initiative to improvise and find a new officer partner to complete a rescue when a message was received to say that the original medic partner was busy. In those cases, rather than wait for a medic partner, the rescue could go ahead if the medic agent delegated the stretcher-carrying role to an officer agent. This was possible because the officer agents were granted individual capabilities to carry a stretcher. In all simulation runs wherein that was enabled, medic agents were observed to take advantage of that option. The average idle time of medic agents was reduced in those simulations compared to other simulations when this policy was not enabled.

The level of organisation awareness that agents in OJAzzIC possess is a distinguishing feature. When agents are not part of an organisation (level 0), the agents' performance is not as good in terms of time or activity (rooms entered). OJAzzIC agents know who else is in the organisation and this awareness enables focused interactions to coordinate beliefs and intentions. Agents in the OJAzzIC MAS organisation directly communicated with each other in order to complete the rescue scenario effectively. Policies obligated the agents to share beliefs and intentions, so there was some efficiency in the rescue approach. When a medic agent was informed that another pair of agents was committed to complete a particular rescue, then that medic agent dropped any individual intentions to complete and potential intentions to consider that same rescue. This feature improved agent efficiency. That agent moved on to different rescues. The number of rooms entered dropped when agents shared beliefs and intentions because there was no duplication of effort.

Built using the the OJAzzIC meta-model, the agents in the implemented MAS successfully achieved the behavioural requirements represented by the agent stories (Section 2.2.1). Table 3 sets out the gaps in other approaches. Meeting the requirements for stories 2, 7, and 8 in particular, is achieved as follows: the organisation framework enables the designer to explicitly describe a meso-layer of rules (social policies) that guide agents in runtime deliberation regarding goal selection; accordingly, SharedPlans can be created using runtime commitments so agents work together to (for example) perform stretcher rescues; design time policies guide runtime improvisation so that agents can help others when asked, even outside pre-defined role responsibilities, provided they are within their capabilities; and agents communicate and adjust their intentions at runtime, relying on goals defined at design time, but using runtime awareness of who to communicate with.

Implementing a MAS based on the OJAzzIC meta-model also provided the platform for the designer to think about policies that could help agents at runtime. As agents in the OJAzzIC organisation apply the policies within their organisation(s), the organisations provide a context for agent interaction and help agents to know who else to share information with. The benefits of this interaction and knowledge cultivation were visible in the simulation system. Even though agents within the organisation were obliged to send more messages than if they were not in an organisation, the overall performance increased when agents were in an organisation.

Our results demonstrate that the increased design time guidance available through the specification of organisational constructs and policies yields runtime flexibility of agent action. This puts the onus on the OJAzzIC designer to consider potential failures and anticipate policies that should be created for improvisation. Accordingly, there will be domains where the presence of "standard operating procedures" and anticipatable gaps can be so encoded—such as emergency management and related serious gaming scenarios [7]—and others where the environment is simply too dynamic for this level of pre-design to be justified or successful. Further work should investigate ways to describe this boundary.

Author Contributions: Conceptualisation, K.K.; investigation, K.K.; methodology, K.K.; supervision, L.S.; writing—review and editing, K.K. and L.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

MAS	multi-agent system
AOSE	agent oriented software engineering
BW4T	block world for teams simulation environment [79]
BDI	belief, desire, and intention [8,9]

References

1. Mintzberg, H. *Structure in Fives: Designing Effective Organizations*; Prentice-Hall: Upper Saddle River, NJ, USA, 1983.
2. Mendonca, D.; Wallace, W.A. A Cognitive Model of Improvisation in Emergency Management. *IEEE Syst. Man Cybern. Part A* **2007**, *37*, 547–561. [CrossRef]
3. Bigley, G.A.; Roberts, K.H. The Incident Command System: High Reliability Organizing for Complex and Volatile Task Environments. *Acad. Manag. J.* **2001**, *44*, 1281–1299.
4. Comfort, L.K. Crisis management in hindsight: Cognition, communication, coordination, and control. *Public Admin. Rev.* **2007**, *67*, 189–197. [CrossRef]
5. Valentine, M.A.; Edmondson, A.C. Team Scaffolds: How Meso-Level Structures Support Role-based Coordination in Temporary Groups. *Organ. Sci.* **2015**, *26*, 405–422. [CrossRef]
6. Trotter, M.; Salmon, P.; Lenné, M. Improvisation: Theory, measures and known influencing factors. *Theor. Issues Ergon. Sci.* **2013**, *14*, 475–498. [CrossRef]
7. Yilmaz, L.; Ören, T.; Aghaee, N.G. Intelligent agents, simulation, and gaming. *Simul. Gaming* **2006**, *37*, 339–349. [CrossRef]
8. Bratman, M.; Israel, D.; Pollack, M. Plans and resource-bounded practical reasoning. *Comput. Intell.* **1988**, *4*, 349–355. [CrossRef]
9. Rao, A.S.; Georgeff, M.P. Modeling rational agents within a BDI-architecture. In *Proceedings of Knowledge Representation and Reasoning*; Fikes, R., Sandewall, E., Eds.; Morgan Kaufmann: San Mateo, CA, USA, 1991; pp. 473–484.
10. Balke, T.; Gilbert, N. How do agents make decisions? A survey. *J. Art. Soc. Soc. Simul.* **2014**, *17*, 13. [CrossRef]
11. Benfield, S.S.; Hendrickson, J.; Galanti, D. Making a Strong Business Case for Multiagent Technology. In Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multiagent Systems, Hakodate, Japan, 8–12 May 2006; pp. 10–15. [CrossRef]
12. Padgham, L.; Winikoff, M. *Developing Intelligent Agent Systems: A Practical Guide*; John Wiley & Sons: Hoboken, NJ, USA, 2005; Volume 13.
13. Tidhar, G.; Heinze, C.; Selvestrel, M. Flying together: Modelling air mission teams. *Appl. Intell.* **1998**, *8*, 195–218. [CrossRef]
14. Adam, C.A.; Gaudou, B. BDI agents in social simulations: a survey. *Knowl. Eng. Rev.* **2016**, *31*, 207–238. [CrossRef]
15. Larsen, J.B. Going beyond BDI for agent-based simulation. *J. Inf. Telecommun.* **2019**, *3*, 446–464. [CrossRef]
16. Singh, D.; Padgham, L.; Logan, B. Integrating BDI agents with agent-based simulation platforms. *Auton. Agents Multi-Agent Syst.* **2016**, *30*, 1050–1071. [CrossRef]
17. Corkill, D.; Durfee, E.; Lesser, V.; Zafar, H.; Zhang, C. Organizationally Adept Agents. In Proceedings of the 12th International Workshop on Coordination, Organization, Institutions and Norms in Agent Systems COIN, AAMAS 2011, Taipei, Taiwan, 3 May 2011; pp. 15–30.
18. Dignum, V. (Ed.) *Handbook of Research on Multi-Agent Systems: Semantics and Dynamics of Organizational Models*; Information Science Reference; IGI Global: Hershey, PA, USA, 2009.

19. Quillinan, T.B.; Aldewerald, H.; Dignum, F.; Dignum, V.; Penserini, L.; Wijngaards, N. Developing Agent-based Organizational Models for Crisis Management. In Proceedings of the 8th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2009), Budapest, Hungary, 10–15 May 2009.
20. Tidhar, G.; Sonenberg, E.A. Observations on Team-Oriented Mental State Recognition. In Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI99) Agent Workshop on Team Behaviour and Plan Recognition, Stockholm, Sweden, 31 July–6 August 1999.
21. Keogh, K.; Sonenberg, E.A. Adaptive Coordination in Distributed and Dynamic Agent Organizations. In *International Workshop on Coordination, Organizations, Institutions, and Norms in Agent Systems*; Cranefield, S., Birna van Riemsdijk, M., Vázquez-Salceda, J., Noriega, P., Eds.; Springer: Berlin/Heidelberg, Germany, 2012; Volume 7254, pp. 38–57.
22. Keogh, K.; Sonenberg, E.A. Coordination Using Social Policies in Dynamic Agent Organizations. In *International Workshop on Coordination, Organizations, Institutions, and Norms in Agent Systems*; Balke, T., Dignum, F., Riemsdijk, M.B., Chopra, A.K., Eds.; Springer: Berlin/Heidelberg, Germany, 2014; Volume 8386, pp. 1–20.
23. DeLoach, S.A. OMACS: A Framework for Adaptive, Complex Systems. In *Multi-Agent Systems: Semantics and Dynamics of Organizational Models*; Dignum, V., Ed.; IGI Global: Hershey, PA, USA, 2009; ISBN 1-60566-256-9.
24. Boissier, O.; Bordini, R.H.; Hübner, J.; Ricci, A. Unravelling Multi-agent-Oriented Programming. In *Agent-Oriented Software Engineering*; Shehory, O., Sturm, A., Eds.; Springer: Berlin/Heidelberg, Germany, 2014.
25. Baldoni, M.; Baroglio, C.; Capuzzimati, F.; Micalizio, R. Commitment-based Agent Interaction in JaCaMo+. *Fundam. Inform.* **2018**, *159*, 1–33. [CrossRef]
26. Gâteau, B.; Khadraoui, D.; Dubois, E.; Boissier, O. MOISEInst: An Organizational Model for Specifying Rights and Duties of Autonomous Agents. In Proceedings of the Third European Workshop on Multi-Agent Systems EUMAS 2005, Brussels, Belgium, 7–8 December 2005; pp. 484–485.
27. Dignum, V. A Model for Organizational Interaction: Based on Agents, Founded in Logic. Ph.D. Thesis, Universiteit Utrecht, Utrecht, The Netherlands, 2004.
28. Jiang, J.; Dignum, V.; Tan, Y.H. An Agent Based Inter-organizational Collaboration Framework: OperA+. In *Coordination, Organizations, Institutions, and Norms in Agent System VII*; COIN 2011. Lecture Notes in Computer Science; Cranefield, S., van Riemsdijk, M.B., Vázquez-Salceda, J., Noriega, P., Eds.; Springer: Berlin/Heidelberg, Germany, 2012; Volume 7254.
29. Grosz, B.; Sidner, C. Plans for discourse. In *Intentions in Communication*; MIT Press: Cambridge, MA, USA, 1990; pp. 417–444.
30. DeLoach, S.A.; Garcia-Ojeda, J.C. O-MaSE: A Customizable Approach to Designing and Building Complex, Adaptive Multiagent Systems. *Int. J. Agent Oriented Softw. Eng.* **2010**, *4*, 244–280. [CrossRef]
31. DeLoach, S.A. O-MaSE: An Extensible Methodology for Multi-agent Systems. In *Agent-Oriented Software Engineering*; Shehory, O., Sturm, A., Eds.; Springer: Berlin/Heidelberg, Germany, 2014.
32. Jensen, A.S.; Alderwereld, H.; Dignum, V. Dimensions of Organizational Coordination. In Proceedings of the 25th Benelux Conference on Artificial Intelligence, Delft, The Netherlands, 7–8 November 2013; pp. 80–87.
33. Sterling, L.; Taveter, K. *The Art of Agent-Oriented Modeling*; MIT Press: Cambridge, MA, USA, 2009.
34. Weyns, D.; Mascardi, V.; Ricci, A. (Eds.) Designing Multi-Agent Systems from Ontology Models. In *Engineering Multi-Agent Systems, Proceedings of the 6th International Workship EMAS 2018, Stockholm, Sweden, 14–15 July 2018*; Revised Selected Papers, LNAI 11375; Springer: Berlin/Heidelberg, Germany, 2019; pp. 76–95.
35. Cossentino, M.; Hilaire, V.; Molesini, A.; Seidita, V. (Eds.) *Handbook on Agent-Oriented Design Processes*; Springer: Berlin/Heidelberg, Germany, 2014.
36. Dalpiaz, F.; Dix, J.; van Riemskijk, M. (Eds.) *Engineering Mult-Agent Systems*; Springer: Berlin/Heidelberg, Germany, 2014; Volume 8758.
37. Julian, V.; Botti, V. Multi-Agent Systems. *Appl. Sci.* **2019**, *9*, 1402. [CrossRef]
38. Shehory, O.; Sturm, A. (Eds.) *Agent-Oriented Software Engineering—Reflections on Architectures, Methodologies, Languages, and Frameworks*; Springer: Berlin/Heidelberg, Germany, 2014.
39. Bresciani, P.; Giorgini, P.; Giunchiglia, F.; Mylopoulos, J.; Perini, A. Tropos: An agent-oriented software development methodology. *J. Auton. Agents Multi-Agent Syst. JAAMAS* **2004**, *8*, 203–236. [CrossRef]

40. Zambonelli, F.; Jennings, M.; Wooldridge, M. Developing Multi Agent Systems The Gaia Methodology. *ACM Trans. Softw. Eng. Methodol.* **2003**, *12*, 317–370. [CrossRef]
41. Pavon, J.; Gomez-Sanz, J.; Fuentes, R. The INGENIAS methodology and tools. In *Agent-Oriented Methodologies*; Henderson-Sellers, B., Giorgini, P., Eds.; Idea Group Publishing: Hershey, PA, USA, 2005; pp. 236–276.
42. Argente, E.; Botti, V.; Julian, V. GORMAS: An Organizational-Oriented Methodological Guideline for Open MAS. In *Agent-Oriented Software Engineering X*; Gleizes, M.P., Gomez-Sanz, J., Eds.; Springer: Berlin/Heidelberg, Germany, 2011; pp. 32–47.
43. Esparcia, S.; Argente, E.; Julián, V.; Botti, V.J. GORMAS: A Methodological Guideline for Organizational-Oriented Open MAS. In *Handbook on Agent-Oriented Design Processes*; Cossentino, M., Hilaire, V., Molesini, A., Seidita, V., Eds.; Springer: Berlin/Heidelberg, Germany, 2014; pp. 173–218.
44. García, E.; Argente, E.; Giret, A. EMFGormas: A CASE tool for developing service-oriented open MAS. In Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems AAMAS 2010, Toronto, ON, Canada, 10–14 May 2010.
45. Cossentino, M.; Hilaire, V.; Gaud, N.; Galland, S.; Koukam, A. The ASPECS process. In *Handbook on Agent-Oriented Design Processes*; Springer: Berlin/Heidelberg, Germany, 2014; pp. 65–114.
46. Dam, H.; Winikoff, M. Towards a next-generation AOSE methodology. *Sci. Comput. Program.* **2013**, *78*, 684–694. [CrossRef]
47. Sturm, A.; Shehory, O. The Landscape of Agent-Oriented Methodologies. In *Agent-Oriented Software Engineering*; Shehory, O., Sturm, A., Eds.; Springer: Berlin/Heidelberg, Germany, 2014.
48. Abdalla, R.; Mishra, A. Comparing the Artifacts of Agent Methodologies. *TEM J.* **2018**, *7*, 433–438.
49. Weyns, D.; Mascardi, V.; Ricci, A. *Engineering Multi-Agent Systems, Proceedings of the 6th International Workshop, EMAS 2018, Stockholm, Sweden, 14–15 July 2018*; Revised Selected Papers; Springer: Berlin/Heidelberg, Germany, 2019; Volume 11375.
50. Nieves, J.; Padget, J.; Vasconcelos, W.; Staikopoulos, A.; Cliffe, O.; Dignum, F.; Vàzquez-Salceda, J.; Clarke, S.; Reed, C. Coordination, Organisation and Model-driven Approaches for Dynamic, Flexible, Robust Software and Services Engineering. In *Service Engineering European Research Results*; Dustdar, S., Li, F., Eds.; Springer: Berlin/Heidelberg, Germany, 2011.
51. Tapia, D.I.; Rodríguez, S.; Bajo, J.; Chorchado, M. FUSION@, A SOA-Based Multi-agent Architecture. In *International Symposium on Distributed Computing and Artificial Intelligence 2008 DCAI 2008*; Corchado, J., Rodríguez, S., Llinas, J., Molina, J.M., Eds.; Springer: Berlin/Heidelberg, Germany, 2009; pp. 99–107.
52. Odell, J.; Nodine, M.; Levy, R. A Metamodel for Agents, Roles and Groups. In Proceedings of the International Workshop on Agent-Oriented Software Engineering (AOSE) V, New York, NY, USA, 19 July 2005; Volume 3382, pp. 78–92.
53. Aldewereld, H.; Boissier, O.; Dignum, V.; Noriega, P.; Padget, J. (Eds.) *Social Coordination Frameworks for Social Technical Systems*; Law, Governance and Technology Series; Springer: Cham, Switzerland, 2016; Volume 30.
54. Da Silva, A.R. Model-driven engineering: A survey supported by the unified conceptual model. *Comput. Lang. Syst. Struct.* **2015**, *43*, 139–155.
55. Coutinho, L.; Brandão, A.; Boissier, O.; Sichman, J. Towards Agent Organizations Interoperability: A Model Drive Engineering Approach. *Appl. Sci.* **2019**, *9*, 2420. [CrossRef]
56. Schneider, M.F.; Miller, M.E.; McGuirl, J.M. Towards a Meta-Model to Specify and Design Human-Agent Teams. In Proceedings of the 36th International Symposium on Aviation Psychology, Dayton, OH, USA, 7–10 May 2019; pp. 97–98.
57. Weyns, D. Software Engineering of Self-Adaptive Systems: An Organised Tour and Future Challenges. In *Handbook of Software Engineering*; Taylor, R., Kang, K.C., Cha, S., Eds.; Springer: Berlin/Heidelberg, Germany, 2017.
58. Schatten, M.; Ševa, J.; Tomičić, I. A roadmap for scalable agent organizations in the Internet of Everything. *J. Syst. Softw.* **2016**, *115*, 31–41. [CrossRef]
59. Uez, D.; Hübner, J. Modeling for openness in MAS. In Proceedings of the 5th International Workshop on Engineering Multi-Agent Systems, EMAS, International Conference on Autonomous Agents and Multiagent Systems (AAMAS), Sao Paulo, Brazil, 8–9 May 2017.
60. Coutinho, L.; Brand, A.; Sichman, J.; Boissier, O. Model-Driven Integration of Organizational Models. In Proceedings of the Agent-Oriented Software Engineering IX, Estoril, Portugal, 12–13 May 2008.

61. Fornara, N.; Balke-Visser, T. Modelling Organizations and Institutions in MAS. *J. Appl. Log. IFCOLOG* **2018**, *5*, 565–590.
62. Hübner, J.; Kitio, R.; Ricci, A. Instrumenting multi-agent organisations with organisational artifacts and agents 'Giving the organisational power back to the agents'. *Auton. Agents Multi-Agent Syst. JAAMAS* **2010**, *20*, 369–400. [CrossRef]
63. Tidhar, G. Organization-Oriented Systems: Theory and Practice. Ph.D. Thesis, Department of Computer Science, University of Melbourne, Melbourne, Australia, 1999.
64. Jennings, N.R. Commitments and Conventions: The Foundation of Coordination in Multi-Agent Systems. *Knowl. Eng. Rev.* **1993**, *8*, 223–250. [CrossRef]
65. Tuomela, R. Joint Intention, We-Mode and I-Mode. *Midwest Stud. Philos.* **2006**, *30*, 35–58. [CrossRef]
66. Tuomela, R. Shared Belief. In *International Elsevier Encyclopedia of the Social and Behavioral Sciences*; Elsevier: Amsterdam, The Netherlands, 2006.
67. Hübner, J.; Sichman, J.; Boissier, O. A Model for the Structural, Functional, and Deontic Specification of Organizations in Multiagent Systems. In *Advances in Artificial Intelligence*; Bittencourt, G., Ramalho, G.L., Eds.; Springer: Berlin/Heidelberg, Germany, 2002; Volume 2507, pp. 118–128.
68. DeLoach, S.; Miller, M. A Goal Model for Adaptive Complex Systems. *J. Adv. Comput. Res.* **2017**, *2*, 83–92.
69. Baldoni, M.; Baroglio, C.; May, K.; Micalizio, R.; Tedeschi, S. Computational Accountability in MAS Organizations with ADOPT. *Appl. Sci.* **2018**, *8*, 489. [CrossRef]
70. Hübner, J.; Sichman, J.; Boissier, O. Developing organised mult-agent systems using the MOISE+ model: Programming issues at the system and agent levels. *Agent Oriented Softw. Eng.* **2007**, *1*, 370–395. [CrossRef]
71. Torroni, P.; Yolum, P.; Singh, M.P.; Alberti, M.; Chesani, F.; Gavanelli, M.; Lamma, E.; Mello, P. Modelling Interactions via Commitments and Expectations. In *Handbook of Research on Multi-Agent Systems: Semantics and Dynamics of Organizational Models*; Dignum, V., Ed.; IGI Global: Hershey, PA, USA, 2009; pp. 263–284.
72. Beydoun, G.; Low, G.; Henderson-Sellers, B.; Mouratidis, H.; Gomez-Sanz, J.J.; Pavón, J.; Gonzalez-Perez, C. FAML: A Generic Metamodel for MAS Development. *IEEE Trans. Softw. Eng.* **2009**, *35*, 841–863. [CrossRef]
73. DeLoach, S.A.; Oyenan, W.H.; Matson, E.T. A capabilities-based model for adaptive organizations. *Auton. Agents Multi-Agent Syst. JAAMAS* **2008**, *16*, 13–56. [CrossRef]
74. Keogh, K.; Sonenberg, E.A. Designing for Planned Emergence in Multi-agent Systems. *Lect. Notes Comput. Sci.* **2015**, *9372*, 97–113.
75. Grosz, B.; Kraus, S. The evolution of SharedPlans. In *Foundations and Theories of Rational Agency*; Springer: Dordrecht, The Netherlands, 1999; pp. 227–262.
76. Acay, D.L.; Sonenberg, L.; Tidhar, G. Formalizing tool use in intelligent environments. *J. Ambient Intell. Humaniz. Comput.* **2019**, *10*, 1597–1610. [CrossRef]
77. Rao, A.S.; Georgeff, M.P. BDI Agents: From Theory to Practice. In Proceedings of the First International Conference of Multi-Agent Systems, San Francisco, CA, USA, 12–14 June 1995; Volume 95, pp. 312–319.
78. Keogh, K. Improvised Coordination in Agent Organisations. Ph.D. Thesis, The University of Melbourne, Melbourne, Australia, 2018.
79. Johnson, M.; Jonker, C.; van Reimsdijk, B.; Feltovich, P.J.; Bradshaw, J.M. *Joint Activity Testbed: Blocks World for Teams(BW4T)*; LNAI, ESAW 2009; Aldewereld, H., Dignum, V., Picard, G., Eds.; Springer: Berlin/Heidelberg, Germany, 2009; Volume 5881, pp. 254–256.
80. Hindriks, K.V. Programming rational agents in GOAL. In *Multi-Agent Programming*; Springer: Boston, MA, USA, 2009; pp. 119–157.
81. Thangarajah, J.; Padgham, L.; Harland, J. Representation and Reasoning for Goals in BDI Agents. In *Proceedings of the Twenty-fifth Australasian Conference on Computer Science—Volume 4*; Australian Computer Society, Inc.: Darlinghurst, Australia, 2002; pp. 259–265.

Sample Availability: Code for the demonstration system is available in the public GitHub repository github.com/OJAzzIC.