A Generic Agent Organisation Framework for Autonomic Systems

Ramachandra Kota, Nicholas Gibbins, and Nicholas R. Jennings

School of Electronics and Computer Science
University of Southampton, Southampton, UK
{rck05r,nmg,nrj}@ecs.soton.ac.uk

Abstract. Autonomic computing is being advocated as a tool for managing large, complex computing systems. Specifically, self-organisation provides a suitable approach for developing such autonomic systems by incorporating self-management and adaptation properties into large-scale distributed systems. To aid in this development, this paper details a generic problem-solving agent organisation framework that can act as a modelling and simulation platform for autonomic systems. Our framework describes a set of service-providing agents accomplishing tasks through social interactions in dynamically changing organisations. We particularly focus on the organisational structure as it can be used as the basis for the design, development and evaluation of generic algorithms for self-organisation and other approaches towards autonomic systems.

Keywords: Organisation, Autonomic Systems, Organisation Model.

1 Introduction

Autonomic systems are envisaged as self-managing, distributed computing systems containing several service-providing components interacting with each other over large networks to accomplish complex tasks. The features of such systems are that they are robust, decentralised, adapting to changing environments and self-organising. Within this, a central concern that needs to be focused on is the interactions between the various computing entities involved. In particular, the interactions within the system are critical for it to achieve its system-wide goals as the tasks tend to be too complex to be accomplished by any single component or entity alone. Given this, and taking inspiration from self-organisation principles, the development of effective autonomic systems involves, to a significant extent, adapting local interactions towards achieving a better performance globally [18]. By so doing, the system can robustly reconfigure itself to the changing requirements and environmental conditions. Therefore, the self-management aspect of the system requires that the individual components of the system are allowed the freedom to adapt their local interactions with other components. In particular, adapting these interactions is necessary because, purely changing the internal characteristics of the components will not be sufficient for improving performance as most of the tasks and goals involve multiple components and interactions across them.

A.V. Vasilakos et al. (Eds.): AUTONOMICS 2009, LNICST 23, pp. 203–219, 2010.

For example, consider the interconnected network of a university as a form of an autonomic grid system. Being a university, it contains various labs with their own specialised computing systems, as part of the overlaying network of the university. That is, there might be a graphics lab containing computers with some high end graphics cards for rendering rich intensive images. Similarly, some computers in the geography lab might contain various GIS maps and related software. Also, there will be complex computing tasks that need several computers (possibly situated in different labs) providing specialised services for their accomplishment. A task might need statistical analysers from the mathematics department for analysing data available from the sociology department in order to predict natural resource, like water and wood, usage as needed by the institute on environmental conservation. Thus, the computers on the university network, need to interact with one another to perform these complex tasks. Moreover, as these individual computers are controlled by different people in different labs, the respective loads on them, at any time, cannot be known or predicted. Also, some might go offline, some might be upgraded and so on. Hence, the computers need to continuously adapt their interactions with others in the network to keep up with the changes and, also optimise the overall performance.

Now, these social interactions of the components can be quite reactive and not guided by definite regulations or they can be structured using an explicitly depicted network or organisation. That is, the individual components of the system will be modelled as autonomous agents participating in an organisation and the interactions between the components are governed by the structure of this organisation. In such a context, regulating the interactions in the system through the organisation structure will aid in the design of adaptation techniques by suitably representing the recurring interactions between the components. For example, consider the autonomic system being used to maintain the computing systems in a university, as discussed earlier. Now, given the large number of computers or components in the system, one computer can hope to maintain links with just a limited number of those in the network. Now, say a computer in the geography lab regularly needs computers with good graphics capabilities for rendering its maps. It has to choose between maintaining links with just one computer or with many in the graphics lab. The former case will lead to less processing at the geography computer during allocation but might lead to delays in task completion when that particular graphics computer is busy. In contrast, the latter case will require more processing at the geography computer every time it has to allocate a task,but might help in getting quicker outputs once the task is allocated. Now, if provided with the structure, the geography computer can smartly choose how many graphics computers to maintain links with by evaluating the possible delays that might occur when accessing most of the graphics computers indirectly and compare that with the resources saved at itself in terms of processing cycles per each allocation task. Once the social interactions are explicitly depicted by the organisation structure, designers seeking to embed adaptation into the system can then use and focus on this organisation as a whole rather than working on each of the individual components separately. Thus, the

organisation model will provide a better overview of the global performance of the system without compromising on the individuality of the constituent entities.

In summary, we argue that a formally modelled organisational representation of the components will help in managing their social interactions [22], and at the same time, provide insights into possible avenues for self-organisation and adaptation towards improving global behaviour. More specifically, we contend that depicting the distributed computing systems, including the service providers, their social interactions and the task environment, using an abstract organisation framework will provide a suitable platform to develop and test techniques attempting to bring about autonomicity into the system.

Against this background, we seek to develop a problem-solving agent organisation model that serves as a fitting abstract representation of such distributed computing systems. Our model will provide an appropriate simulation framework for distributed systems by modelling the task environment, the computational entities and their interactions along with their performance evaluation measures. Such a platform can then be used for developing and evaluating generic approaches designed for autonomic systems. In this context, by problem-solving agent organisations we refer to those containing some service-providing agents that receive inputs, perform tasks and return results. We chose to use problem-solving agent organisations because they can be decentralised with autonomous and independent agents which accomplish tasks by providing services and collaborating with each other through social interactions governed by the organisation's structure. Thus, it models the salient features of distributed computing systems and, at the same time, contains the flexibility required to make them autonomic. Following the reasons detailed above, the focus of the model is on the inter-agent interactions; that is, the organisation structure and its effect on the system. Moreover, we also present an evaluation mechanism for the performance of the organisation based on the tasks or goals achieved by it. This method is developed such that the critical role played by the organisation structure on the performance is made explicit and clear. Therefore, any designer of self-organisation techniques, especially those focusing on the structure or network, will be able to see their method in action and evaluate its performance before being transported and put in the actual domain specific autonomic systems.

In more detail, our organisation models a set of service-providing, resource constrained agents facing a dynamic stream of tasks requiring combinations of services with some ordering. The agents only posses local views and need to interact with each other to accomplish the tasks. These interactions are governed by the relations existing between the agents (organisation structure) and affect task allocation and organisation's performance. Finally, keeping in mind the development of adaptation techniques based on the structure, we also provide a method of representing the costs and resources involved in reorganisation.

In the next section (Sec. 2), we discuss the current literature in our context. Then in Sec. 3, we present our organisation framework together with the task environment, agents, organisational characteristics and performance measures. We illustrate our model with an example in Sec. 4 and conclude in Sec. 5.

2 Related Work

As we are seeking to develop an organisation framework that suitably represents distributed computing systems, it should provide an abstract representation of the components of the system, their social interactions and the tasks that they perform along with the environment that they are based in. Correspondingly, organisation modelling involves modelling the task environment, the organisational characteristics (structure and norms), the agents and the performance measures. In the following, we study the current literature in each of these aspects.

2.1 Modelling Tasks

The tasks faced by the organisation can be atomic or made up of two or more tasks (or subtasks) which, in turn, may be composed of other tasks. The tasks may have dependencies among them, resulting in a temporal ordering of the tasks in the organisation. In this context, [19] identifies three kinds of such dependencies— pooled, sequential and reciprocal. Two or more tasks whose results are jointly required to execute another task are said to be in a pooled dependency relation with each other. A sequential dependency exists between tasks if they have to be performed in a particular sequence. Finally, a reciprocal dependency exists if the tasks are mutually dependent on each other and have to be executed at the same time. However, the tasks dependencies as suggested by Thompson have subsequently been interpreted in different ways in different models. In particular, [11] model the task dependencies in their 'Virtual Design Team (VDT)' closely following Thompson's model. In fact, they even extend the sequential and reciprocal dependencies by classifying each of them into different types. In contrast, in the PCANS model, [13] demonstrate that both pooled and reciprocal dependencies, as described by Thompson, can be derived from sequential dependencies. Thus, their representation enables the designer to model just a single dependency type. For our present requirements, we just require a simple task model containing dependencies, and hence we will use the PCANS model.

2.2 Modelling Organisational Characteristics

Approaches towards organisational design in multi-agent systems can be considered to be either agent-centric or organisation-centric [14]. The former focus on the social characteristics of agents like joint intentions, social commitment, collective goals and so on. Therefore, the organisation is a result of the social behaviour of the agents and is not created explicitly by the designer. On the other hand, in organisation-centric approaches, the focus of design is the organisation which has some rules or norms which the agents must follow. Thus, the organisational characteristics are *imposed* on the agents. As we are primarily interested in problem-solving agent organisations, we only study organisations in multi-agent systems whose design is modelled explicitly.

In this context, the OperA and OMII frameworks [2,21] formally specify agent societies on the basis of social, interaction, normative and communication structures. However, in both of these frameworks, the agents are not permitted to

modify the pre-designed organisational characteristics. Hence, they do not provide a suitable platform for self-organisation. Islander [16] also uses a similar approach by expecting the designer to pre-design the roles and the possible interactions between them thus delivering fixed patterns of behaviour for the agents to follow. Thus, this too is not flexible enough to incorporate reorganisation.

A more useful and simpler model developed by [3] provides a meta-model to describe organisations based on agents, groups and roles (AGR). While their model mainly pertains to groups of agents and intra-group dynamics (which does not apply to our requirements), they model organisation structure as defining the possible interactions between the groups. This interpretation of the structure matches our purpose and lies behind our model as well. A somewhat similar approach is followed by Moise [8], which considers an organisation structure as a graph defined by a set of roles, a set of different types of links and a set of groups. An agent playing a role must obey some permitted behaviours comprising the role. Organisation links are arcs between roles and represent the interactions between them. These links can be of three types— communication, authority and acquaintance. However, the links have a context associated with them and are valid within that context only. We seek an organisation structure that is not so specific or bounded. Nevertheless, some of the ideas used in this model, especially those relating to the organisation structure will be used while developing our model. A slightly different approach is followed by the Virtual Design Team (VDT) framework [11]. Its purpose is to develop a computational model of real-life project organisations. It does not use the agent-role paradigm. Instead, the agents are fixed to their duties and are called actors. The organisation structure is composed of a control structure and a communication structure. Evidently, VDT attempts to model a problem-solving organisation, and therefore, very relevant for our requirements. However, it lacks flexibility in the organisation structure, as it only permits purely hierarchical organisations. Therefore, we do not directly use the whole VDT model but only some parts of it.

In contrast to the above models, mathematical approaches have been developed for creating organisations [10,17]. However, they produce an instantiated organisation according to complex and elaborate specification of organisational requirements but not the generic model we need. In a more relevant work, [15] aim at an organisation framework that is flexible enough for self-organisation. However, they take a strictly emergent view of self-organisation and focus mainly on the social delegation aspects in agent organisations. Furthermore, their method specifies a set of organisation models, and the participating agents choose, whether or not, to join such organisations. Therefore, it does not inherently aid the development of problem-solving agent organisations. Another work [20] follows a norm based approach for modelling hierarchical organisations in which every role has a *position profile* associated with it. This profile is specified by positional norms and an agent can take up a role by changing its own set of norms to conform to these positional norms. However, the model requires that all positions and norms are specified at the outset itself thereby not allowing for the flexibility in the interactions as sought by us.

2.3 Modelling Agents

An overview of modelling agents in the context of organisations is presented by [1]. From this, it is apparent that the modelling of agents varies across different organisation models. In particular, agents may be homogeneous or belong to different classes, be cooperative or competitive. Their abilities may be represented as a simple vector or as a complex combination of skills, strategies, preferences and so on. Against this background, while all the organisation design approaches described above, with the exception of VDT, leave the agent development to the designer, VDT models the members of the organisation called actors in great detail. The main characteristics of the actors are attention allocation (determines the decision making behaviour of how the actor chooses among several task alternatives) and information processing (determines the skills, capacity and other processing characteristics). This design of agents will be partly used in our organisation model as it meets our requirements for modelling agents in the context of problem-solving organisations. Another concept that we will use is obtained from [6] where the agents are required to perform task assignment but can only address one request per time-step. Thus, we will also make use of this concept of agents possessing limited computational capacities so that the efficiency of the agents plays a prominent role in the performance of the organisation, thereby, reflecting the real-life scenarios where the components of the autonomic systems often possess small and limited computational power.

2.4 Evaluating an Organisation's Effectiveness

Organisation characteristics play a major role in the performance of the organisation [5]. Therefore, there are a number of existing methods for evaluating an organisation's characteristics based on parameters like robustness of the structure, connectivity and degree of decentralisation [9,7]. However, these measures are independent of the tasks being handled by the system and thus, fail to capture the suitability of the organisation according to the environment it is situated in. A contrasting criterion is to measure the performance of the organisation on the basis of how well it performs its tasks [4]. We believe this provides a good indication of the organisation's efficiency during run-time. In this context, in VDT, the measure of the performance of the organisation is on the basis of the load on the organisation. The load on the organisation is represented in units of *work volume*, thereby providing a common calibration for different tasks. The total work volume of a task depends partly on the task specification and partly on the organisational characteristics. Therefore, the resultant load on the organisation is a function of the tasks and the organisational characteristics and acts as a performance indicator. Therefore, the approach chosen by VDT is more suitable for our requirements and will be taken into account.

3 The Agent Organisation Framework

We describe our organisation framework by first detailing the task environment. Then we describe the agents and the organisation structure, before discussing the performance evaluation mechanism.

3.1 Task Representation

The task environment contains a continuous dynamic stream of tasks that are to be executed by the organisation. A task can be presented to the organisation at any point of time and the processing of the task must start immediately from that time-step. Thus, the organisation of agents is presented with a dynamic incoming stream of tasks that they should accomplish. In detail, the organisation of agents provides a set of services which is denoted by S. Every task requires a subset of this set of services. Services are the skills or specialised actions that the agents are capable of. We model the tasks as work flows composed of a set of several service instances (SIs) in a precedence order, thereby representing a complex job as expected in autonomic systems. We define a service instance si_i to be a 2-tuple: $\langle s_i, p_i \rangle$ where $s_i \in S$ (i.e. s_i is a member of the services set S), $p_i \in \mathbb{N}$ denotes the amount of computation required.

Following the PCANS model of task representation (see Sec. 2.1), we only consider sequential dependencies between the service instances. Thus, the SIs of a task need to be executed following a precedence order or dependency structure. This dependency structure is modelled as a tree in which the task execution begins at the root node and flows to the subsequent nodes. The task is deemed complete when all its SIs have been executed in the order, terminating at the leaf nodes. The complete set of tasks is denoted by W and contains individual tasks w_i which are presented to the organisation over time.

3.2 Organisation Representation

Since, we aim to model the agent organisation to represent a distributed computing system, our organisation framework consists of a set of computational agents representing the individual components. An agent is an independent computational entity that can provide one or more services. We model our agents by simplifying the agent model used by VDT (see Sec. 2.3) and consider only the information processing characteristics of the agents by overlooking the attention allocation characteristic. The attention allocation characteristic enables an agent to schedule its allocated tasks. The task scheduling algorithms at an agent will depend on the system that is being represented. However, this aspect is internal to an agent and independent of the organisational dynamics which is our primary focus. Therefore, we do not need to model this aspect.

In more detail, the agents are associated with particular sets of services (like say, in the example home-management system, a controller manages the heating system and can also access the internet for communication, thus containing two services in its service set). These sets can be overlapping, that is two or more

agents may provide the same service. Also, building on the agent model used by Gershenson (see Sec. 2.3), every agent also has a computational capacity associated with it. The computational load on an agent (explained later), in a time-step, cannot exceed this capacity. This modelling of resource constrained agents is necessary because, generally the components of an autonomic system are small embedded devices with low computational power. Formally, let A be the set of agents in the organisation. Every element in this set is a tuple of the form:- $a_x = \langle S_x, L_x \rangle$ where the first field, $S_x \in S$ denotes a set of services that belong to the complete service set S and $L_x \in \mathbb{N}$ denotes the capacity. The agents, their service sets and their capacities may change during the lifetime of the organisation.

The other features of an agent organisation, in general, are its structure and norms. The structure of an organisation represents the relationships between the agents in the organisation, while the norms govern the kind of interactions and messages possible between the agents. However, since we are developing a problem-solving organisation, the agents are all internal to the organisation and share the same goals. Moreover, all the agents will be designed in the same way, and therefore, their interaction protocol will be similar and can be internally specified. Therefore, an explicit definition of norms is not required to regulate their interactions. Thus, in our model, the relationships between the agents (denoted by the structure) also determine the interactions between the agents. Formally, an organisation is defined as consisting of a set of agents and a set of organisational links. It can be represented by a 2-tuple of the form:- $ORG = \langle A, G \rangle$ where A, as stated above, is the set of agents, G is the set of directed links between the agents (will be described later in this section).

As mentioned previously, the organisation is presented with a continuous stream of tasks which are completed by the agents through their services. Tasks come in at random time-steps and the processing of a task starts as soon as it enters the system. Task processing begins with the assignment of the first SI (root node). The agent that executes a particular SI is, then, also responsible for the allocation of the subsequent dependent SIs (as specified by the task structure) to agents capable of those services. Thus, the agents have to perform two kinds of actions: (i) execution and (ii) allocation. Moreover, every action has a load associated with it. The load incurred for the execution of a SI is equal to the computational amount specified in its description, while the load due to allocation (called management load) depends on the relations of that agent (will be explained later). As every agent has a limited computational capacity, an agent will perform the required actions in a single time-step, as long as the cumulative load (for the time-step) on the agent is less than its capacity. If the load reaches the capacity and there are actions still to be performed, these remaining actions will be deferred to the next time-step and so on. We allow the agents to perform more than one action in a time-step to de-couple the time-step of the simulation with the real-time aspect of the actual computing systems. Thus, the time-step of the model places no restrictions whatsoever and can represent one or several processor cycles in the actual system.

As stated earlier, agents need to interact with one another for the allocation of SIs. The interactions between the agents are regulated by the structure of the organisation. Inspired from the Moise approach (see Sec. 2.2), we adopt the organisational links paradigm to represent the structure. However, unlike in Moise, the links in our case are not task-specific because we do not assume that the agents will be aware of all the tasks at the outset itself. Moreover, instead of using several graphs to represent particular aspects, we use an *organisation graph* (G) to represent the structure. The nodes in the graph represent the agents of the organisation while the links represent the relations existing between them. Thus, the structure of the organisation is based on the relations between the agents that influence their interactions.

In more detail, we classify the relationships that can exist between agents into four types — (i) `stranger` (not knowing the presence), (ii) `acquaintance` (knowing the presence, but no interaction), (iii) `peer` (low frequency of interaction) and (iv) `superior-subordinate` (high frequency of interaction). The superior-subordinate relation can also be called the authority relation and depict the authority held by the *superior* agent over the *subordinate* agent in the context of the organisation. The peer relation will be present between agents who are considered equal in authority with respect to each other and is useful to cut across the hierarchy. Also, the relations are mutual between the agents, that is for any relation existing between two agents, both the concerned agents respect it. The type of relation present between two agents determines the information that they hold about each other and the interactions possible between them. The information that an agent holds about its relations is:

1. The set of services provided by each of its peers (S_y of each peer a_y)
2. The accumulated set of services provided by each of its subordinates. The *accumulated service set* of an agent is the union of its own service set and the accumulated service sets of its subordinates, recursively. Thus, the agent is aware of the services that can be obtained from the sub-graph of agents rooted at its subordinates though it might not know exactly which agent is capable of the service. AS_x denotes the accumulated service set of agent a_x.

Whenever an agent finishes the execution of a particular SI, it has to allocate each of the subsequent dependent SIs to other agents (this may include itself). The mechanism for allocating SIs to other agents is also mainly influenced by the agents' relations. The decision mechanism of an agent is as follows:

- When an agent needs to allocate a SI, it will execute the SI if it is capable of the service and has no waiting tasks (capacity is not completely used up)
- Otherwise, it will try to assign it to one of its subordinates containing the service in its accumulated service set. This involves the agent traversing through the accumulated service sets (AS_x) of all its subordinates and then choosing one subordinate from among the suitable ones. If the agent finds no suitable subordinate (no subordinate or their subordinates are capable of the service) and it is capable of the service itself (but did not initially assign to self because its capacity is filled), then it will add the SI to its waiting queue for execution.

- If neither itself nor its subordinates are capable of the service, then the agent tries to assign it one of its peers by traversing through their service sets and choosing from among the suitable ones (those capable of the service).
- If none of the peers are capable of the service either, then the agent will pass it back to one of its superiors (who will then have to find some other subordinates or peers to execute the service).

Therefore, an agent mostly delegates SIs to its subordinates and seldom to its peers. Thus, the structure of the organisation influences the allocation of the SIs among the agents. Moreover, the number of relations of an agent contributes to the management load that it incurs for its allocation actions, since an agent will have to sift through its relations while allocating a SI. One unit of management load is added to the load on the agent every time it considers an agent for an allocation (mathematically modelled in Sec. 3.3). Therefore, an agent with many relations will incur more management load per allocation action than an agent with fewer relations. Also, a subordinate will tend to cause management load more often than a peer because an agent will search its peers only after searching through its subordinates and not finding a capable agent. Generally, it is expected that an agent will interact more frequently with its subordinates and superiors than its peers. This process of assigning a SI to an agent requires sending and receiving messages to/from that agent. Thus, task allocation also requires inter-agent communication which adds to the cost of the organisation.

In summary, the authority relations impose the hierarchical structure in the organisation, while the peer relations enable the otherwise disconnected agents to interact directly. It is important to note that while we present only these kinds of relations, the model allows the flexibility to depict more relation types in a similar fashion. Thus, the set of the relation types presented here can be expanded or contracted depending on the domain that is to be represented by the organisation model. Using this model, we abstract away the complex interaction problems relating to issues like service negotiation, trust and coordination. We do so, so that the model keeps the focus on the essence of self-organisation and autonomicity and isolates its impact on system performance.

Formally denoting the structure, every link g_i belonging to G is of form:- $g_i = \langle a_x, a_y, type_i \rangle$ where a_x and a_y are agents that the link originates and terminates at respectively and $type_i$ denotes the type of link and can take any of the values in the set $\{Acqt, Supr, Peer\}$ to denote the type of relation existing between the two agents. The absence of a link between two agents means that they are `strangers`.

3.3 Organisation Performance Evaluation

The performance of a computing system denotes how well it performs its tasks. In terms of an agent organisation, the performance measure can be abstracted to the profit obtained by it. In our model, the profit is simply the sum of the rewards gained from the completion of tasks when the incurred costs have been subtracted. In more detail, the cost of the organisation is based on the amount

of resources consumed by the agents. In our case, this translates to the cost of sending messages (communication) and the cost of any reorganisation taking place within the organisation. Thus, the cost of the organisation is:

$$cost_{ORG} = C. \sum_{a_x \in A} c_x + D.d \tag{1}$$

where C is the communication cost coefficient representing the cost of sending one message between two agents and D is the reorganisation cost coefficient representing the cost of changing a relation. c_x is the number of messages sent by agent a_x and d is the number of relations changed in the organisation.

As stated earlier, agents have limited capacities and their computational load cannot increase beyond this capacity. Since, an agent might perform three kinds of actions in a time-step (task execution, task allocation, adaptation), the load on an agent is the summation of the computational resources used by the three actions and can be represented by three terms. Thus, the load l_x on agent a_x in a given time-step is:

$$l_x = \sum_{si_i \in W_{x_E}} p_i + M \sum_{si_j \in W_{x_F}} m_{j,x} + R.r_x \tag{2}$$

- p_i is the amount of computation expended by a_x for executing SI si_i.
- $m_{j,x}$ is the number of relations considered by a_x while allocating SI si_j.
- W_{x_E} is the set of SIs (possibly belonging to many tasks) executed by a_x.
- W_{x_F} is the set of SIs being allocated by a_x.
- M is the 'management load' coefficient denoting the computation expended by an agent to consider one of its relations while allocating a single SI.
- R is the 'reorganisation load' coefficient, denoting the amount of computational units consumed by an agent while reasoning about adapting a relation.
- r_x is the number of agents considered by a_x for adaptation, in that time-step.

In this way, M and $m_{j,x}$, together represent the computational load for task allocation that is affected by the relations possessed by the agent, thereby providing a simple and explicit method of denoting the effect of the organisation structure on the individual agents. Similarly, R and r_x are used to represent the load caused by reasoning about adaptation (if any). Thus, the coefficient R denotes the amount of resources at the agent that are diverted for adaptation rather than performing tasks and help in deciding about when to reason about adaptation (meta-reasoning).

Since, the load l_x of a_x cannot exceed its capacity L_x, any excess SIs will be waiting for their turn, thus delaying the completion time of the tasks. The rewards obtained by the organisation depend on the speed of completion of tasks. In particular, a task w completed on time accrues the maximum reward b_w which is the sum of the computation amounts of all its SIs:

$$b_w = \sum_{i=0}^{|si_w|} p_i \tag{3}$$

where si_w is its set of SIs. For delayed tasks, this reward degrades linearly with the extra time taken until it reaches 0:

$$reward_w = b_w - (t_w^{taken} - t_w^{reqd}) \tag{4}$$

where t_w^{taken} represents the actual time taken for completing the task, while t_w^{reqd} is the minimum time needed. Thus, the total reward obtained by the organisation is the sum of the rewards of the individual tasks completed by the organisation:

$$reward_{ORG} = \sum_{w \in W} reward_w \tag{5}$$

where W is the set of all tasks. The organisation's performance is measured as:

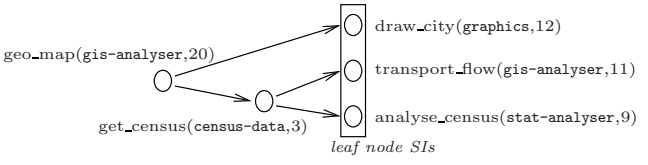
$$profit_{ORG} = reward_{ORG} - cost_{ORG} \tag{6}$$

Thus, for higher profits, the reward should be maximised, while the cost needs to be minimised. Both of these are affected by the allocation of tasks between the agents which, in turn, is influenced by the organisation structure. It is important to note that the agents only have a local view of the organisation. They are not aware of all the tasks coming in to the organisation (only those SIs allocated to them and the immediately dependent SIs of those allocated SIs) and neither are they aware of the load on the other agents. In spite of this incomplete information, they need to cooperate with each other to maximise the organisation profit through faster allocation and execution of tasks. Therefore, by modelling both the decentralisation and individual agent load along with inter-agent dependence and global profit, this evaluation mechanism suitably models the requirements faced by a designer while developing autonomic systems. In the same vein, reasoning and adapting the organisation also take up resources (as denoted by R and D) in our model, thus reflecting real-life scenarios.

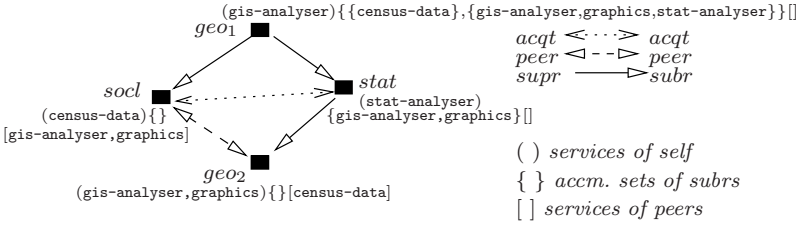
4 Applying the Agent Organisation Framework

We illustrate our framework by using it to depict an autonomic system in charge of the university grid network system as outlined in Sec. 1. First, to illustrate our task model, consider a sample task possibly faced by this system. Assume that a project involves producing a predictive model of a given city. Such a task will involve analysing the GIS data of the required city, obtaining the population density of the city over the past years and then using some kind of statistical analysers on this data to estimate the population distribution in the future. It will also involve predicting the changes to the city transport system using the GIS information on this estimated population, and alongside render the map of the city graphically. In more detail, let us assume that the first part of this task will be to obtain the geographical data of the city and analyse it. In terms of our model, this can be designated as SI `geo_map` needing service `gis-analyser` with computation 20 (very intensive job). After this, let us say that the subsequent sub-tasks are obtaining the historical population data of

the city and rendering the city-map graphically. These will form SIs get_census and draw_city requiring services `census-data` (getting and cleaning the census information from the archives) and `graphics` (graphically modelling to result in an image). Finally, execution of get_census might reveal that further statistical analysis is required to properly predict the population growth in the future and also that the growth caused by immigration depends on the transport incoming to the city. These can be designated as analyse_census and transport_flow requiring services `stat-analyser` and `gis-analyser` (as the transport network of the city can be obtained by analysing the GIS data) respectively. Also, note that the computation required for geo_map is much higher than that required for transport_flow even though both SIs need the same service. The task structure for this scenario, including the SIs and the dependencies is shown in Fig. 1(a). Representing this task formally:



(a) Task structure



(b) Organisation structure

Fig. 1. Representation of an example task and organisation

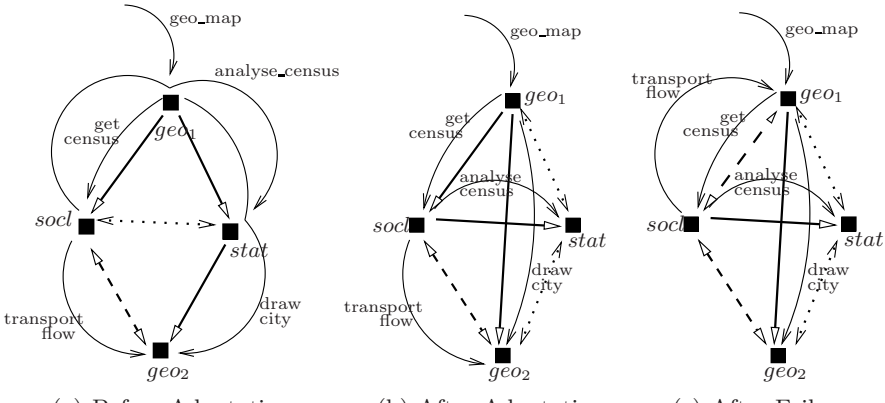
In the same vein, consider a sample agent organisation to represent the autonomic system. Taking a limited view, let us focus on only four agents in this organisation— geo_1 and geo_2(two computers in the geography department), *socl* (a computer in the sociology department) and *stat* (an analyser in the statistics department). The services provided by the agents are basically their capabilities in terms of hardware, software and data accessible to them. Therefore, let us assume that geo_1 provides service `gis-analyser`. Similarly *socl* provides `census-data`, which is the population data of various places in all the past years, and *stat* provides `stat-analyser` service. However, geo_2 is capable of providing both `gis-analyser` (just like geo_1) and `graphics` (because it also contains high end graphics cards for rendering maps). Given this, let us look at the possible structure of the organisation. Let *socl* and geo_2 have a peer relationship. Also, assume geo_1 has two subordinates — *socl* and *stat* (because, say, often GIS based jobs are followed by either census information or statistical analysis).

stat, in turn, has geo_2 as a subordinate. Moreover, while *socl* and *stat* are acquaintances of each other, geo_2 and geo_1 are not aware of each other. The G for this organisation contains 5 organisational links:

$$G = \{\langle geo_1, socl, Supr\rangle, \langle geo_1, stat, Supr\rangle, \langle stat, geo_2, Supr\rangle, \langle socl, geo_2, Peer\rangle, \langle socl, stat, Acqt\rangle\}$$

For this organisation, the organisation graph is shown in Fig. 1(b). The absence of an arrow between two agents means that they are strangers. In addition, the information possessed by the agents about the services provided by their relations is also shown. For example, the accumulated service set (*AccmSet*) of agent geo_1, in turn, contains three sets representing its own service (`gis-analyser`), *AccmSet* of its subordinate *socl* (`census-data`) and of its other subordinate *stat* (`gis-analyser,graphics,stat-analyser`).

As an illustration of the allocation process, consider the sample organisation in Figure 1(b) executing the task shown in Figure 1(a). The allocation of SIs across the agents occurs as shown in Figure 2(a). In detail, we assume that the task arrives at agent geo_1. Hence, geo_1 checks that it is capable of geo_map (as it is capable of service `gis-analyser` and has available capacity) and therefore, allocates geo_map to itself. After execution, geo_1 needs to allocate the two dependencies of geo_map which are get_census and draw_city to capable agents. For allocating get_census, it checks the accumulated service sets of its two subordinates (*socl* and *stat*) and allocates to *socl* (because it is the only one capable of service `census-data`). Similarly, it allocates draw_city to *stat* because this subordinate contains service `graphics` in its accumulated service set. However, *stat* has to reallocate draw_city to its subordinate geo_2 which is actually capable of that service. Similarly, after *socl* executes get_census, it needs to allocate the two dependencies (transport_flow and analyse_census) to appropriate agents. So, *socl* allocates transport_flow to its peer geo_2 as it has no subordinates. It also



(a) Before Adaptation (b) After Adaptation (c) After Failure

Fig. 2. Allocation of the task in Fig. 1(a) in the organisation

hands back analyse_census to its superior geo_1 as it has found no subordinates or peers with that service (`stat-analyser`). geo_1 then assigns analyse_census to its subordinate *stat* (capable of `stat-analyser`) which then proceeds to execute it.

Thus, the structure of the organisation influences the allocation of service instances among the agents. Therefore, an efficient structure can lead to better and faster allocation of tasks. We see that in Figure 2(a), the allocation of draw_city and analyse_census was indirect and needed intermediary agents (*stat* and geo_1 respectively). Now, suppose on the basis of some adaptation method (such as that detailed in [12]), the agents modify their relations to form the structure as shown in Figure 2(b). That is, geo_1 and geo_2 decide to form a superior-subordinate relation and so do *socl* and *stat*. Meanwhile *stat* ends up becoming only an acquaintance of geo_1 and geo_2 as they decide to change the previously existing authority relations. With this new structure, the allocation of the SIs turns out to be much more efficient as all the allocations end up being direct one-step process. Therefore, they take shorter time because intermediary agents are not involved. Moreover, this allocation process requires less computation and communication because, for any SI, only a single agent performs the allocation and sends only one message. Compared to previous structure, this decreases the load on geo_1 and *stat* without putting extra load on other agents.

Now, let us suppose that after some time has passed, geo_2 is reconfigured (perhaps, the OS is reinstalled) such that it is no longer able to to provide `gis-analyser`. In such a scenario, *socl* will no longer be able to delegate transport_flow to geo_2 and will be handing back the SI to its superior geo_1. *socl* does so only after unsuccessfully considering its own subordinates and peers for allocation, thus causing more load onto itself and also taking more time. Under these changed circumstances, *socl* and geo_1 should realise that it is better to change their current relation into a peer relation so that *socl* can delegate to geo_1 quicker. Reversing the existing superior-subordinate relation will not be as useful because geo_1 also needs to continue delegating SIs like get_census to *socl*. Hence, these two agents change their relation as shown in Fig. 2(c).

In this way, the performance of the organisation can be improved by modifying the organisation structure through changes to the agent relationships. This will involve changes to the organisation graph G.

With this example and the sample adaptation scenarios, we see that adaptation of the structure plays an important role in the performance of the organisation. Furthermore, we illustrated that our framework not only provides a well-suited platform to represent autonomic systems, but also gives insights into possible avenues for self-organisation and permits the agents to perform the required adaptation. Here, while we showed how a more efficient structure can lead to the better allocation of a task, we should note that the organisation is performing several tasks at any given time and that the structure is common to all these possibly dissimilar tasks. Given this, the adaptation method should be such that the agents are able to identify which set of relations will be most suitable for their current context on the basis of the kind of tasks facing them in addition to their own service sets and allocation patterns.

5 Conclusions

In this paper, we introduced an abstract organisation framework for depicting distributed computing systems to aid in the development of autonomic systems. We presented our model by detailing our representation of the task environment and the organisation along with a performance evaluation system. The tasks are made up of service instances, each of which specifies the particular service and the computation required. The organisation consists of agents providing services and having computational capacities. The structure of the organisation manifests the relationships between the agents and regulates their interactions. Any two agents in the organisation could be strangers, acquaintances, peers or superior-subordinate. The relations of the agents determine what service information is held by the agents about the other agents and how to allocate service instances to them. We also presented the coefficients that affect the environment (communication cost, management load, reorganisation load) and the functions for calculating the organisation's cost and reward, thus enabling us to evaluate the profit obtained by it when placed in a dynamic task environment.

Our organisation framework provides a simulation platform that can be used by designers to implement and test their adaptation techniques before porting them to real and domain-specific systems. In particular, we designed our model such that the agents, though generic, realistically represent the components that would compose autonomic systems. The organisation is decentralised and agents possess local views and limited capacities like any large distributed computing system. Nevertheless, the agents interact with each other based on the organisation structure, which also influences the task allocations and thereby the organisational performance. This presents a suitable environment for self-organisation, which we have illustrated by using it to represent an intelligent and adapting, autonomous home-management system. In this context, our framework provides sufficient flexibility for the agents to modify their characteristics and social interactions, that is, manage themselves, just as expected in autonomic systems. Furthermore, we also provided the reorganisation cost (D) and load coefficients (R) to represent the price of adaptation. Thus, we have presented a suitable organisation framework that can be used as a platform for developing adaptation techniques, especially focusing on the agents' social interactions.

References

1. Carley, K.M., Gasser, L.: Computational organization theory. In: Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence, pp. 299–330. MIT Press, Cambridge (1999)
2. Dignum, V.: A model for organizational interaction: based on agents, founded in logic. Ph.D. thesis, Proefschrift Universiteit Utrecht (2003)
3. Ferber, J., Gutknecht, O., Michel, F.: From agents to organizations: An organizational view of multiagent systems. In: Proc. of 4th Intl. Workshop on Agent Oriented Software Engineering, Melbourne, Australia, pp. 214–230 (2003)
4. Fox, M.S.: An organizational view of distributed systems, pp. 140–150. Morgan Kaufmann Publishers Inc., San Francisco (1988)

5. Galbraith, J.R.: Organization Design. Addison-Wesley, Reading (1977)
6. Gershenson, C.: Design and control of self-organizing systems. Ph.D. thesis, Vrije Universiteit Brussel (2007)
7. Grossi, D., Dignum, F., Dignum, V., Dastani, M., Royakkers, L.: Structural evaluation of agent organizations. In: Proc. of the 5th AAMAS (2006)
8. Hannoun, M., Boissier, O., Sichman, J.S., Sayettat, C.: Moise: An organizational model for multi-agent systems. In: Proc. of the 7th Ibero-American Conf. on AI (IBERAMIA-SBIA 2000), pp. 156–165 (2000)
9. Horling, B., Lesser, V.: A survey of multi-agent organizational paradigms. The Knowledge Engineering Review 19(4), 281–316 (2005)
10. Horling, B., Lesser, V.: Using quantitative models to search for appropriate organizational designs. In: Autonomous Agents and Multi-Agent Systems (2008)
11. Jin, Y., Levitt, R.E.: The virtual design team: A computational model of project organizations. Computational & Mathematical Organization Theory (1996)
12. Kota, R., Gibbins, N., Jennings, N.R.: Self-organising agent organisations. In: The Eighth International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2009), pp. 797–804 (2009)
13. Krackhardt, D., Carley, K.M.: A pcans model of structure in organizations. In: Proc. of Intl. Symp. on Command and Control Research and Technology (1998)
14. Lematre, C., Excelente, C.B.: Multi-agent organization approach. In: Proc. of 2nd Ibero-American Workshop on DAI and MAS, Toledo, Spain (1998)
15. Schillo, M., Bettina Fley, M.F., Hillebrandt, F., Hinck, D.: Self-organization in multiagent systems: from agent interaction to agent organization. In: Proc. of Intl. Workshop on Modeling Artificial Societies and Hybrid Organizations (2002)
16. Sierra, C., Rodriguez-Aguilar, J.A., Noriega, P., Esteva, M., Arcos, J.L.: Engineering multi-agent systems as electronic institutions. UPGRADE The European Journal for the Informatics Professional V(4), 33–39 (2004)
17. Sims, M., Corkill, D., Lesser, V.: Automated organization design for multi-agent systems. Autonomous Agents and Multi-Agent Systems 16(2), 151–185 (2008)
18. Tesauro, G., Chess, D.M., Walsh, W.E., Das, R., Segal, A., Whalley, I., Kephart, J.O., White, S.R.: A multi-agent systems approach to autonomic computing. In: Proc. of 3rd AAMAS (2004)
19. Thompson, J.D.: Organizations in Action: Social Science Bases in Administrative Theory. McGraw-Hill, New York (1967)
20. Montealegre Vázquez, L.E., López y López, F.: An agent-based model for hierarchical organizations. In: Noriega, P., Vázquez-Salceda, J., Boella, G., Boissier, O., Dignum, V., Fornara, N., Matson, E. (eds.) COIN 2006. LNCS (LNAI), vol. 4386, pp. 194–211. Springer, Heidelberg (2007)
21. Vazquez-Salceda, J., Dignum, V., Dignum, F.: Organizing multiagent systems. Autonomous Agents and Multi-Agent Systems 11(3), 307–360 (2005)
22. Zambonelli, F., Jennings, N.R., Wooldridge, M.: Developing multiagent systems: The gaia methodology. ACM Trans. Softw. Eng. Methdol. 12(3) (2003)