A Specification Method for the Scalable Self-Governance of Complex Autonomic Systems

Martin J. Randles

A thesis submitted in partial fulfilment of the requirements of Liverpool John Moores University for the degree of Doctor of Philosophy

School of Computing and Mathematical Sciences

Liverpool John Moores University

Liverpool

August 2007

ORIGINAL COPY TIGHTLY BOUND

THE FOLLOWING HAVE NOT BEEN COPIED ON INSTRUCTION FROM THE UNIVERSITY

Table 3.1 page 60

Figure 6.2 page 113

Figure 8.4 page 160

Appendix 2 pages 210 - 220

To Donna

Abstract

IBM, amongst many others, have sought to endow computer systems with self-management capabilities by delegating vital functions to the software itself and proposed the Autonomic Computing model. Hence inducing the so-called self-* properties including the system's ability to be self-configuring, self-optimising, self-healing and self-protecting. Initial attempts to realise such a vision have so far mostly relied on a passive adaptation whereby Design by Contract and Event-Condition-Action (ECA) type constructs are used to regulate the target systems behaviour: When a specific event makes a certain condition true then an action is triggered which executes either within the system or on its environment. Whilst, such a model works well for closed systems, its effectiveness and applicability of approach diminishes as the size and complexity of the managed system increases, necessitating frequent updates to the ECA rule set to cater for new and/or unforeseen systems' behaviour.

More recent research works are now adopting the parametric adaptation model, where the events, conditions and actions may be adjusted at runtime in response to the system's observed state. Such an improved control model works well up to a point, but for large-scale systems of systems, with very many component interactions, the predictability and traceability of the regulation and its impact on the whole system is intractable. The self-organising systems theory, however, offers a scaleable alternative to systems control utilising emerging behaviour, observed at a global level, resulting from the low-level interactions of the distributed components. Whereby, for instance, key signals (signs) for ECA style feedback control need no longer be recognised or understood in the context of the design time system but are defined by their relevance to the runtime system. Nonetheless this model still suffers from a usually inaccessible control model with no intrinsic meaning assigned to data extraction from the systems operation. In other words, there is no grounded definition of particular observable events occurring in the system. This condition is termed the *Signal Grounding Problem*. This problem cannot usually be solved by analytical or algorithmic methods, as these solutions generally require precise problem formulations and a static operating domain. Rather cognitive techniques will be needed that perform effectively to evaluate and improve performance in the presence of complex, incomplete, dynamic and evolving environments.

In order to develop a specification method for scalable self-governance of autonomic systems of systems, this thesis presents a number of ways to alleviate, or circumvent, the Signal Grounding Problem through the utilisation of cognitive systems and the properties of complex systems. After reviewing the specification methods available for governance models, the Situation Calculus dialect of first order logic is described with the necessary modalities for the specification of deliberative monitoring in partially observable environments with stochastic actions. This permits a specification method that allows the depiction of system guards and norms, under central control, as well as the deliberative functions required for decentralised components to present techniques around the Signal Grounding problem, engineer emergence and generally utilise the properties of large complex systems for their own self-governance. It is shown how these large-scale behaviours may be implemented and the properties assessed and utilised by an Observer System through fully functioning implementations and simulations. The work concludes with two case studies showing how the specification would be achieved in practice: An observer based meta-system for a decision support system in medicine is described, specified and implemented up to parametric adaptation and a NASA project is described with a specification given for the interactions and cooperative behaviour that leads to scale-free connectivity, which the observer system may then utilise for a previously described efficient monitoring strategy.

Acknowledgements

I am indebted to Professor Taleb-Bendiab, my supervisor, for his constant support, encouragement and friendship. He has contributed in many ways to the work described herein. Besides arranging the financial support that facilitated my completion of this thesis, he is responsible for my development as a researcher and provided the vital scientific inspiration for the work.

I would like to thank Philip Miseldine (Now Dr. Miseldine) for his support and friendship throughout these past three years. The many discussions between Taleb, Phil and myself provided many interesting and fruitful lines of research and clarified many technical aspects of the work.

My thanks and appreciation is due to Professor Merabti, Director of the School of Computing and Mathematical Sciences at Liverpool John Moores University for providing the necessary environment and opportunities to complete the work.

Acknowledgement is also due to all my colleagues; academic staff, administrative staff, technicians and research students in the School for their help and support.

The work, in the first instance (2004-2005) was supported by EPSRC, as part of the 2nrich Project, grant number GR/R86782/01.

I dedicate this work to my wife, Donna, for her psychological support, patience and understanding and for ultimately making it all worthwhile.

Finally a thank you to Lucy whose long walks in the sand dunes, provided some contemplative time for the development of these ideas, and for her quiet solicitude during the writing up of this work.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

At present facilities to enable distributed computer systems to exhibit self-governance remain at a rudimentary level of sophistication: Many major adaptations and adjustments still need to be accomplished offline. This work investigates the specification of self-governance, by the system itself, which can be applied to systems of varying scales. This will become increasingly necessary with the advent of large-scale distributed computer systems incorporating ubiquitous and pervasive computing environments within planet scale systems, such as the Internet. A paradigm shift in this direction has been proposed by IBM with its vision of Autonomic Computing (Horn, 2001) to address complexity and endow systems with self-governance capabilities by delegating vital functions to additional software components, inducing the so called self-* properties including the system's ability to be Self: Configuring, Optimising, Healing and Protecting.

1.1 Motivation

For computer software systems to possess the ability to control and influence their own performance requires the provision of additional software components to the system itself. These components (meta-systems), in line with the Aspect Oriented Programming (AOP) (Kiczales et al, 1997) principles, ought to be treated as completely separate concerns from the system. In the autonomic computing reference models (Randles et al, 2005, Kephart and Chess, 2003) such meta-control capabilities are achieved through feedback and control loops, where monitors are set on chosen key values and specified actions are triggered when pre-programmed thresholds are exceeded, or other events, are detected by the system. These autonomic systems, in effect, become libraries of rules (policies) or production systems working in a similar manner to Event-Condition-Action (ECA) (Widom and Ceri, 1995) rules, first used in active databases, where an event causes a condition to be fulfilled triggering the specified action.

Evidently this has led to further software design complexity including increasing the workload of control and communication functions, with networks of autonomic systems being applied to monitor increasingly complicated distributed application level systems. Hence the size of these systems exceeds the capabilities of the meta-systems to maintain

a sufficiently agile and efficiently organized rule set. When so many rules are defined within a system, there is likely to be many conflicts, amongst the rules, and their interactions, in general, are very difficult to analyse. For instance, the execution of one rule may cause an event, which triggers another rule or set of rules. These rules may in turn trigger further rules and there is a potential for an infinite cascade of rule firings to occur. Additionally these rules are static, in nature, in that there is usually no provision for rule refinement or analysis. A system rule requiring alteration or adjustment necessitates the system, or system part, being taken offline, reprogrammed and deployed back into the system. Thus this work highlights the need for techniques and tools for the meta-systems to analyse rule behaviour and its impact on the system. In particular the adoption of a cognitive systems design model is proposed to enable the deliberation on the effect of rule enactment on the system, to refine or delete the rules in the light of new data and to determine new rules arising from the systems evolutionary operation. In other words, rather than the meta-systems blindly reacting to pre-defined stimuli, the cognitive systems provide the meta-system with knowledge of the intrinsic meaning of perceived events and the likely outcomes of its actions. Thus, as the complexity of a system increases, the "top-down" command and control approach becomes increasingly difficult, if not impossible, to program. It is, indeed, impossible to program, at design time, for all system eventualities. So unless sufficiently powerful cognitive systems are deployed, as meta-systems, on large-scale complex systems failure will inevitably occur, with increasing likelihood, as the system progresses further and further from its starting state or configuration.

1.2 Challenges

As introduced in the previous section it is not currently possible to provide highly agile and adaptable meta-systems for self-governance for larger scale distributed computing. Thus the challenge addressed in this work is to consider how best to specify a meta-system for self-governance that can be applied to scale from a simple closed system up to planet wide completely open systems.

Systems, in this work, are viewed as being composed of many individual components. These components may be atomic or may consist of further components. They are programmable entities operating according to their own specific programming model or dynamic rule set. Hence component interactions are directly related to rule interactions,

component conflicts are akin to the rule conflicts and any component may be defined, at a specific snapshot in time, by its rule set at that time. Thus the systems are considered as federations of interacting components. A component can also be viewed as a software agent. This is as far as this work goes in defining the concept of an agent, the term will be used when it is necessary to draw results from other work or make clear certain points. There are many proposals for frameworks to specify agent/component behaviour. But none provide a totally integrated, unified approach for specifying command and control top-down management, which may be used for less complex closed systems and the more complex deliberative capabilities required when component interactions cause system behaviour to emerge from the bottom-up. One of the earliest attempts at defining an agent's behaviour is Belief-Desire-Intentions (BDI) (Bratman, 1987), but no allowance is made for co-operative or social interaction between BDI agents and intention reconsideration is not always possible (Cohen and Levesque, 1990). More recent improved approaches, such as Epistemic-Deontic-Axiologic (EDA) (Filipe and Liu, 2000) or Extendable-Belief-Desire-Intentions (EBDI) (Badr et al, 2004) agents, handle interactions, team-work and conflict resolution but there are no facilities for handling emergent global events or behaviour. As systems become larger and more complex it becomes increasingly unfeasible for a meta-system to function in monitoring and controlling the system participants as well as assessing the global state and environment of the system. Furthermore it is impossible to detect weaknesses within the system structure or operation simply by analysis of the rule base for each participant. In the first instance, a new statistical paradigm is required where the state of the system is assessed and influenced by likelihood, rather than certainty, giving a high confidence in the predictability of outcomes. Thus exhaustive monitoring is no longer required, rather system participants are governed by simple rules and observation modules enforce a global model based on the likelihood of the results of the participants' interaction. In this way a top-down approach is combined with a bottom-up assessment. Thus the research challenges, addressed by this work, involve the provision of a unifying formalism, and associated tools and techniques, which is flexible enough for both the specification of system members' behaviour and the analysis of global emergence and novelty. The formalism is required to provide the cognitive facilities, necessary for deliberation, of reasoning, deduction, abduction, induction and inference as well as providing high assurance for correctness of the specification. This then provides the required

functionality to implement the deliberative capabilities and runtime adaptation necessary for future autonomic networks on large-scale systems.

1.3 Research Hypothesis

The challenges, which the research hypothesis needs to address through this work, are mainly concerned with providing a means of specifying self-governance capabilities for computational systems of varying scales. A specification is, thus, sought that permits not only the formalisation of rule sets to handle command and control type meta-systems but also includes associated cognitive capabilities to reason on observed system outcomes and provide the self-governance meta-system with the knowledge of the actual meaning of observed phenomena within the system and its domain. Current meta-system provision does not allow a sufficiently flexible representation of the knowledge within the system to achieve this. The production type systems currently employed, as meta-systems, represent knowledge as states and transitions, where each piece of knowledge and each rule for transition must be explicitly enumerated. To facilitate a more flexible representation, akin to human management, cognitive systems are required to deliberate and reason on the system knowledge and its impact on system management. Thus a basic objective to be addressed by the research hypothesis, throughout this work, is to provide a specification technique for system management that possesses the deliberative skills of a human operator but with the superior data handling capacities and speed of a computational system. It is therefore necessary to consider the acquisition and handling of knowledge within a computer system. Brian Smith (1982) formulated the Knowledge Representation Hypothesis:

"Any mechanically embodied intelligent process will be comprised of structural ingredients that (a) we as external observers naturally take to represent a propositional account of the knowledge that the overall process exhibits, and (b) independent of such external semantical attribution, play a formal but causal and essential role in engendering the behaviour that manifests that knowledge."

Thus a specification in mathematical logic is called for as a foundation for "the propositional account" and additionally this specification then gives the system's behaviour through logical entailment. Furthermore it is observed that the participant interactions, in a normative environment, causally effect this normative environment. The ensuing evolution, as the system adapts to new situations, causes the creation,

modification or deletion of norms. It is postulated that the meta-system ought to adjust to these system and environmental requirements and not vice-versa. In this way the meta-systems possess an observational role with possibilities to influence components in response to observed global topology, assessed through its cognitive facilities. This requires that the system must attribute meaning to its observations in order to account for realistic future actions or events: Signals emanating from the system need to have grounding within the system. Finally this approach means that the deployed system ought to possess the increased functionality and capabilities to perform its adaptations and adjustments mostly at runtime. Incorporating all these notions into a single statement gives the research hypothesis for this work:

Self-governance functions for Autonomic computer systems ought to be intelligent processes, separately embodied within the computer system, which are better modelled through cognitive observation systems, based on a unified specification, for all scales of system, in mathematical logic. This specification both enforces limits for the bounded autonomy of the system participants (from the top-down) and provides for deliberation on the global implications and actions of these participants as well as the emerging features of the participants' interactions (from the bottom-up) through the establishment of intrinsic meanings for observed system signals giving a grounded definition of the events occurring within the system.

1.3.1 Research Questions

The type of observation system envisaged for the meta-system includes a hierarchy of observation components that possess deliberative functions, on signatures of emergent self-organisation for instance, giving abilities to pass deliberation upwards through the hierarchy. In this way deliberation can be performed at the most suitable point in the observation system and the result propagated back through the monitoring system. The observation hierarchy is conceived as culminating, in the worst-case scenario, with human intervention. Thus it is intended to present a specification method for a formal model that permits a direct implementation to allow the scaling of monitoring and Self-* functions up to large-scale complex Worldwide systems.

This and the above stated research hypothesis naturally leads to the two major research questions:

- What is the problem that prevents current autonomic self-governance systems from being applied to large-scale complex systems?
- How may grounded definitions of system signals/events be automatically assessed by the system itself? Can a system achieve an automatic grounded definition of its observed events: Is it possible for the meta-system to have an understanding of the actual meaning to the system of observed phenomena?

A number of supplementary research questions, to aid the investigation, then follow:

- How can a formal logic inspired account of cognitive systems contribute towards the design, specification and analysis of deliberative meta-systems to influence and monitor, for the automatic runtime maintenance, tuning and security, as required in autonomic self-* functions, of large-scale, complex, distributed computing systems?
- Can a formal specification be achieved that encompasses the modelling requirements of normative and socially interactive components and permits the emergence of a provably correct observable collective behaviour not apparent just from individual component monitoring?
- What would be a suitable observer/system topology and how would deliberation proceed?
- How are the defining characteristics and signatures for classes of self-organising behaviour represented and assessed without the need for exhaustive monitoring?
- Can a design method, which may be implemented, be produced that includes deliberation on global system state, scaleable to very wide distribution/complexity and permits the formal specification of normative component behaviour?

1.4 Research Approach

The methodological foundation of this approach, for a formal definition of an observational meta-system in a large-scale computer system, is founded on developing a mathematical and computational treatment of dynamical systems (Reiter, 2001). Thus it is proposed to commence by assuming the Knowledge Representation Hypothesis (Smith, 1982). This has a number of immediate implications:

1. Mathematical logic, in contrast to more algebraic descriptions, is the natural choice of language for the expression of the, "propositional account of the knowledge that the overall process exhibits."

2. Separated observation systems, to best mimic the deliberative processes that we as humans use, ought to be based on this propositional account.

3. What a system component (agent) "believes" to be true inherently provides the overall system "knowledge" through the comprised formally constituted collective.

4. The usual state-based approach is redundant here. Rather than explicitly enumerating states and their transition functions the Knowledge Representation Hypothesis calls for a propositional account: A representation based on sentences describing what is true in the system and its environment and of the causal laws in operation for the same. Thus a full design time specification is not necessary.

5. Part (b) of the hypothesis makes a causal connection between these sentences and the system's behaviour. In logic entailment is used to derive one sentence from another. Thus system behaviour may be viewed as a logical consequence of the "propositional account": Assessing how a system operates is equivalent to deducing how it must behave given its description. So system design is simplified to logically describing the system.

6. An abstract logical specification is gained from a propositional account. Thus it is possible to automatically prove, within the logic, properties of the specification: Logical deduction can be used to establish correctness properties for the system.

7. Efficient deduction leads to an executable system specification. Thus a logical specification gives a simulator for the system.

Thus it is proposed to use a propositional approach, based on the Knowledge Representation Hypothesis, to specify both a more traditional top down, command and control, formalism and an analogous bottom up specification. The micro-scale interactions can be specified in the logic as well as the deliberative functions of a cognitive system for the monitoring and assessment of both recognised and novel emergent macro-scale features.

1.4.1 Research Position

The research position, of this work, is based on using a unified logical approach to model any dynamical system with: A norm-based approach, an emphasis on the promotion and detection of signatures for emergent self-organising behaviour, or a combination of the two. The research problems detailed are very similar to those encountered in Cognitive Robotics (Reiter, 2001). In the early 1980s Hector Levesque argued that feasible implementations for large complex systems require what were termed *Vivid Representations* (Levesque, 1986). These require, as fundamental elements: What Levesque calls a base theory in *database form,* which has, as a crucial property, methods available to store and query very large amounts of contained data and which supports a series of associated definitions. This means that predicates and functions of the base theory, specified in the *database form,* are treated as atomic elements and novel elements, introduced into a theory, are defined in terms of these atomic primitives. The evaluation of a query thus amounts to decomposing the query, through the definitions, into a query expressed solely in terms of the atomic elements, which can then be evaluated against the base theory. The present, most widely used representative of such a form is the relational database. However the *vivid representation* asks only that the *database form* have the capability to efficiently handle large quantities of information. Thus the base theory is not necessarily required to be complete, as would be the case for a relational database. In the field of Cognitive Robotics this led, eventually, to the use of an updated version of John McCarthy's Situation Calculus (McCarthy, 1963) where the process of regression is used, with the base theory of the axioms of Situation Calculus, to reduce a situation to the initial situation (Levesque et al, 1998). Thus system properties may be proved simply by proving properties of the initial situation. It is for similar reasons that this work adopts the Situation Calculus to specify and develop the required formalism for the deliberative functions of meta-systems, including reasoning and analysis on autonomic system behaviour. The requirement for the system to provide grounded definitions of events, occurring at runtime, means that there is a similar prerequisite here for the cognitive reasoning system to be able to handle large amounts of data.

In particular this work postulates that such deliberative capabilities, gained through the Situation Calculus model, may also be used to reason on the emergence of events or behaviour over the entire system. One of the most prevalent forms of self-organisation, seen in many natural and large-scale man-made systems is the emergence of scale-free

(Barabasi and Albert, 1999) topologies in the connectivity of components. In order to represent component, bounded autonomy, social norms and this form of global system emergence it is proposed to take a collectivist approach, instead of the usual simplification of reductionism. In this way deliberation on the emergence of scale-free behaviour can be achieved through considering the perceived global situation as the social interactions of the participating components, as well as the specified individual norms, actions and behaviours of the participants.

1.4.2 Research Method

The research methods used in this work are inspired by the design and specification of meta-system observational and normative functions with a logical approach to modelling dynamical systems using first/second order logic, in the guise of Situation Calculus, proposed by John McCarthy (McCarthy, 1963) and later extended to be more of a calculus of situations by Levesque et al (1998). This is allied to a rigorous assessment of signatures for the global emergent property of scale-free connectivity between system participants in order to solve the research questions detailed above in relation to providing scaleable, autonomic type meta-systems. Using contributions and combinations from these areas of research; formalism has been developed that permits both individual and global deliberation through a collectivist model that can be shown to work through fully implemented simulation software.

1.4.3 Research Scope

This research includes many strands of enquiry; some of which are obviously interconnected whereas others appear to be from very diverse disciplines. This section is concerned with drawing the strands together and making clear the interconnectedness of the various lines of thought.

In the first instance this work requires a perspective on how systems are viewed from a philosophical standpoint. To overcome the current failings of computer systems to possess scaleable autonomic, self-* meta-systems it is proposed to take a collectivist approach to systems thinking. In this the tasks performed and social interactions of the system components lead to the perceived outcomes of the enclosing system. Indeed the goals of the enclosing system are seen as being of a higher priority than the individual goals of the components (Ratner and Lumei, 2003). Through this treatment emergence of behaviour and outcomes becomes of greater importance and the whole is seen to be

greater than the sum of its parts. This also means that a positivist view is also rejected, in that whole system ontology emerges from a socially constructed meaning emanating from the interaction of the system components. That is observations and knowledge are of an individual nature but an objective world is not sufficient to explain all outcomes: There is also a need for an observing subject. The philosophy underlying the construction of an observation system, in large-scale computing systems, may be more adequately represented by a form of collectivism termed "hierarchical reductionism" by Richard Dawkins (1996) in his book The Blind Watchmaker. Here complex systems can be described with a hierarchy of abstractions where each level in an organizational structure is only describable in terms of objects one level down in the hierarchy. Thus the goals of the enclosing system, at the top of the hierarchy with human intervention, have no perception of the goals of the low-level components, in the lower reaches of the hierarchy. The top-level requirements emerge from the systems whole operational environment and function.

Gödel (1931) showed the futility of an approach based on reductionism to mathematics in general. Chomsky (1965), in defining cognitive science, also challenged the approach of reductionism, apparent in behaviourism, suggesting that cognitive function plays a greater part. Cognitive science, however, also focuses on the deliberative functions of an individual with mental states. The related field of Artificial Intelligence also sees cognitive abilities as relating to an individual's mental or information processing powers. This work promotes a shift of emphasis to more completely encompass the collectivist approach in what has been termed; "Mind Out of Programmable Matter" (Randles et al, 2006a): The system consists of its own global identity as well as its constituent parts.

In the area of Distributed Artificial Intelligence the study of multi-agent systems is included within the scope of this work. This provides for the study of computational agents within the context of their situated environment and their social relationships with other agents. As previously stated the term agent will be merely shorthand for programmable computing entity or system component throughout this work. In 1987 Bratman established an approach to agent intentions with BDI (Bratman, 1987). There are a number of documented weaknesses in this approach, not least the lack of support for social activity in multi-agent systems or facilities to allow reconsideration of intentions. In 1990 the notion of agents acting together and the formalisation of joint intention in collaborative multi-agent settings was introduced (Levesque et al, 1990).

Throughout the work presented in this thesis mathematical logic is the most important tool in specifying and reasoning on the behaviour of computer systems. When it is required to simulate, control, analyse or in any way investigate a dynamical system the first step is to axiomatise it in a suitable logic. All features will follow through logical entailment. Although it is not claimed that this is completely obvious it is hoped this work goes some way to showing this in action. In taking the approach of Cognitive Robotics, in utilising the formalism of Situation Calculus, many problems and solutions for the provision of meta-system scalability can be addressed; the most important being that explicit pre-defined state transitions are not necessary. Thus the ideas of emergence and the unpredictable outcomes of component interactions can be included in the formalism through the appropriate use of stochastic and statistical techniques, where necessary, to ground the definition of system actions/events.

Ashby introduced the term self-organisation (Ashby, 1947) and it has been widely studied initially from a cybernetic and general systems theory perspective. It has only very recently come to prominence in physics and the study of increasingly more complex computing systems as exemplified by the World Wide Web and Internet, through statistical mechanics. Here the prominent form of observed organisational behaviour is the emergence of power-law (Barabasi and Albert, 1999) topologies in the graphs representing many various relationships across these systems. The absence of a representative mean, in these distributions, has led to the systems being described as "Scale-Free".

Recent research in mathematical logic, has been informed and inspired by the requirements of Distributed Artificial Intelligence to model social agents in terms of responsibility, normative assertions, autonomy and other notions of relationship such as; joint intentions, commitments and obligations. Modal logics have been suggested for the modelling of these concepts; deontic, action, or default logics, for instance. Such formalisms prove less than versatile in application to the wide ranging and real world concerns of complex systems. Similarly process algebras suffer from having no correctness formalism built into the representation and also require explicit enumeration of state spaces.

This thesis will provide background material on these subjects. But it is the vivid representations to handle very large amounts of data, ultimately realised through the Situation Calculus, that is chosen as the most promising and unifying approach for a top-

down observer/influencer and a bottom up account of component interaction and global emergence.

1.4.4 Research Aims and Objectives

This thesis has a number of aims and objectives to maintain the vision of autonomic middleware for future large-scale, complex, interwoven and pervasive computing systems, based on the reasoning set out in the preceding sections:

- An identification of the major obstacle(s) making self-governance difficult or impossible in these widely distributed computer systems.
- Proposals and demonstrable methods to address the identified problem(s).

In keeping with the previously stated research position this gives rise to a number of additional objectives:

- The provision of a suitable formal specification method to encompass federated behaviour, stochastic actions, sensing and knowledge and the reasoning over event sequences such as Markov Chains.
- The account of a formalism that is equally applicable to "top-down" engineering approaches, for the representation of norm bounded autonomy and socially interactive frameworks, and the specification of deliberative functions for reasoning over "bottom-up" emergence.
- An assessment of the evolution and properties prevalent in large-scale system networks in order to account for the likelihood of emergence.
- The specification of efficient methods for manipulating knowledge data.
- The implementation or simulation of key features for the type and scale of the systems under consideration.
- The statement and analysis of suitable case studies as evaluation.

1.5 Contribution to Knowledge

This thesis presents the following novel contributions:

- A statement of the Signal Grounding problem and its consequences for control and monitoring in large-scale systems of systems.

- A specification method to support autonomic software engineering for deliberative meta-system governance of, and reasoning on, autonomic systems behaviour for varying scales of system to include some addressing of the Signal Grounding Problem.
- A unifying formalism used to represent system and individual component norms as well as the deliberative capabilities necessary for Signal-Grounding, engineering emergence and harnessing the properties of large-scale networks. Thus aiding the development of models for the monitoring, influence and self-* functions of large, planetary scale systems.
- A new formal model of an observational meta-system for the monitoring and influencing of large-scale computer systems based on deliberation of both the normative bounding of autonomy, for components, and analysing the global emergence and events that emanate from the whole system.
- A software engineering specification method for the implementation of deliberative observers over a complex system to include provision for the likelihood of emergent behaviour as a consequence of the normative interactions of participants.

From the outset Signal Grounding is recognised in this work as presenting a major obstacle, preventing the efficient utilisation of monitoring data, in establishing the recognition and recurrence of emergent phenomena in large, complex systems. This means that without addressing the Signal-Grounding problem the adaptations necessary for larger-scale systems self-governance will not be achievable. Thus this work initially takes an approach to the specification of group behaviour in agent frameworks and seeks further ways to provide the functionality necessary to handle and allow whole system properties to emerge in a way that is recognisable to, and subsequently detectable by, the governance system. In this way the normal functions of system control can also be used to influence and assess the emerging environment: The system is a controller of itself in its situated environment. The use of a formal specification method that permits the representation of deliberative mechanisms as well as the description of normative positions, through mathematical logic, allows assessments of system signals with recurrence based on reinforcement. This is further refined to include the reuse of action histories to reproduce desirable signals (engineer emergence) and the analysis, detection and utilisation of widespread properties of large-scale networks of systems.

1.6 Thesis Structure

The thesis commences with a review of complex systems and their associated meta-systems, with approaches to the representation, analysis and formal specification, encompassing the approaches sketched out in the research scope section of this introduction. Then the Situation Calculus is introduced and the contribution of this thesis is presented in the form of a directly implemented formalism for an observational meta-system that is scaleable to World Wide Web size systems, using features and novel metrics to inform the efficient deliberative properties of a propositional account. A number of case studies are presented as an empirical evaluation and evidence of the implemented systems.

Chapter 2 starts with an overview of various philosophical standpoints on the administration of large systems of interacting entities. It is shown that a collectivist model provides the necessary features for the cognitive meta-systems demanded here. In this chapter the relevant background is also given for the construction of multi-entity systems through agent architectures, normative environments and federated behaviour for teamwork that underpin this research with a modal logic of mutual beliefs. It is further shown how self-organisation may result from this federated behaviour, which is then demonstrated using the World Wide Web as an example. The philosophical approaches to large systems and social interactions are evaluated with respect to individual autonomy versus normative governance and global outcome.

In Chapter 3 a literature review is provided on the state of the art for models of self-management in distributed systems, autonomic computing applications and self-management in middleware services. The chapter continues with a discussion on the currently unachievable goal of allowing these methods to scale up to World Wide Web proportions because of an identified Signal-Grounding problem.

Chapter 4 forms a literature review with relevant background on the state of the art formalisms for specifying representing and reasoning on complex and agent based system models. It concludes by observing that a propositional account of the domain gives the more flexible and concise representation for the required cognitive meta-systems, yet no such formalism exists, at present, to represent the top-down command and the deliberative capabilities necessary for emergent behaviour from the bottom-up, required for a meta-system on a large complex system. For this propositional account, requiring no

prior state enumeration, Situation Calculus is described in detail with its support for counterfactual reasoning.

Chapter 5 introduces the Observer System for systems exhibiting emergent outcomes. It is shown how a specification can be handled by the Observation System to engineer a previously discussed, naturally occurring emergent behaviour and a specification of deliberation is described to address the Signal-Grounding problem.

Chapter 6 further introduces background properties and evolutionary features of large-scale systems and networks that can be utilised by the Observer System when such behaviour is detected. The classes of behaviour and connectivity are reviewed through a model of phase transitions leading from a regular lattice to a scale-free connected topology. A metric is proposed, the Hub Connection Density Measure, to determine what topological phase a particular system is exhibiting, permitting the deployment of the most appropriate monitoring strategy. In particular, when a scale-free topology is detected, a newly defined Acquaintance Monitoring Selection algorithm can be employed by the observation system to ensure optimum system knowledge gathering. Moreover a representation of the situation space that gives a scale-free topology can utilise similar methods to remove less important or redundant data.

Chapter 7 introduces the implementation and simulations for realising the specifications from a normative based autonomic scenario to the handling of large complex systems in a statistically mechanical way, demonstrating the use of the previously defined metrics, tools and techniques through simulations predominately in the Netlogo language.

Chapter 8 brings all the approaches together in two empirically evaluated case studies. Firstly the approach has been tested and fully implemented through a number of medical collaborative research projects. Secondly the methods have been applied to analyse swarm type behaviour in a future proposed NASA space mission.

Finally for Chapter 9 a conclusion is given with a summary of the results and contributions leading to the proposal of future work.

Chapter 2

Complex Systems

Complex computing systems are becoming increasingly prevalent because of three main factors: Functionality, ubiquity and manageability. Firstly there is a general desire to endow systems with as many functions as possible. Currently requirements engineering actively seeks to satisfy the vast majority of system user requests and there is a much effort devoted to design time fine-tuning in seeking to predict and deal with every system eventuality. Secondly the sectors of application for computer systems are rapidly enveloping all areas of human activities. Computing power is being embedded in many more devices and the networking of formerly isolated or non-computational objects is leading to a pervasive complex computing environment. Thirdly, as a result of this functionality and ubiquity, the management of computer systems is outstripping the information processing capacities of human operators. This has led to the establishment of additional computing systems, sitting above the application level systems, to provide management functions through the runtime of the system. These systems have been termed self-adaptive, self-* or autonomic systems, but the basic feature is varying degrees of self-governance: The system having autonomous functions, independent of any human permission, enabling the system to be a manager of itself. Following a law of requisite variety (Ashby, 1947) these systems are required to match the complexity of the system they are endeavouring to manage and so contribute a significant layer of complexity to the system itself.

This chapter is intended to introduce the philosophical and computational foundations for knowledge representation in controlling large numbers of interacting entities, which may be conceived of as being contained within a system. This provides a relevant background for clarifying the methodological differences between top-down and bottom-up construction, in such systems as detailed in Chapter 1, with an evaluation based on the evolution and topology of the World Wide Web.

2.1 The Philosophical Standpoints

Through the twentieth century an increasingly recurring theme, in the development of science, was the realisation that the Newtonian paradigm of mechanics was not sufficient

to explain or model the observed complexity of natural systems or phenomena. The reductionist view that every observed phenomenon can be reduced to a set of deterministic natural laws governing the behaviour of the atoms or particles involved became too rigid and sterile to handle the spontaneous creativity inherent in natural systems. Large complex systems appear to be subject to the random occurrence of novel structures and autonomous adaptation to environmental stimuli not apparent in more simple linear systems. This leads to a perceived unpredictability in the functioning of large systems that calls for a review of the methods used to handle large distributed computational systems. Firstly a more collectivist perspective must be taken to reason on the global state of the comprised system. This ought not to replace reductionist ideas completely, as it is necessary to order the system components hierarchically to allow for the bounded autonomy of the low level participants, that precipitates the global situation. Secondly it is essential that the systems in place possess strong deliberative capabilities. It is only through the addition of correct, verifiable cognitive systems that the scale of these complex systems can be addressed.

2.1.1 Reduction versus Collection

The spontaneous emergence of new structure is quite readily observed in the real world. It is generally seen at some point in a system's progress or evolution that may be termed a phase transition. So the formation of ice crystals in liquid water heralds the emergence of symmetric patterns of dense matter in a medium of randomly moving molecules. What characterises this event is self-organisation where structure or patterns appear without any external agent intervening (other than reducing temperature, which is not sufficient to explain the ordering of structure, in the liquid, from a random motion). It seems as if the system arranges itself into a more ordered pattern. This phenomenon contradicts the reductionist, mechanistic worldview; it also does not easily fit with an intuitive understanding of the world, as confirmed by the second law of thermodynamics, which states that a system left to itself will inevitably increase in entropy (disorder). This, apparent paradox, is explained in terms of global entropy: The randomly moving molecules that are fixed in the crystalline ice structure pass on the energy of their movement to the medium in which they are moving. Thus the decrease in entropy of the crystal structure is offset by an increase in entropy of the surrounding environment. The entropy of the whole effectively increases. Thus the observed behaviour is reduced to a comparison of entropies. So a reduction approach gave some explanation of observed

phenomena, yet the outcome for the system as a whole can only be explained in terms of the collective behaviour. In relating this to the observation of a complex distributed computing system it can be seen that the system may be hierarchically disassembled into interacting units so that a reduction of the systems complexity may be achieved through the individual analysis of each unit. Additional events occur within the system, however, as a result of the complex interactions of the units, which can only be ascertained through an appreciation of the collective. Richard Dawkins (Dawkins, 1996) coined the term "Hierarchical Reductionism" to describe a complex system, as a hierarchy of systems each of which is only describable using objects from one layer down in the hierarchy. Thus a complex system can be reduced to component parts but the behaviour of the systems results not only from the actions of the component parts but also from the interactions of those parts.

Therefore the use of the term collective, in this work, is simply meant to convey the concept of the collected system possessing a goal, which is satisfied by the system's participants pursuing their own individual goals. This allows for whole systems' events that are not explainable as the sum of events from the systems' parts. The reduction approach is that complex phenomena can always be explained by reducing the occurrence to the sum of simpler, less complex phenomena. So in less complex systems, with little significant interaction of subsystems, the reduction approach need not be rejected out of hand. As computational systems increase in complexity, however, it becomes necessary to take into account the collectivist perspective encompassing the outcomes from knowledge distributed across the system members. Reduction to subsystems can still be useful but allowances have to be made for the global emergence that ensues from the subsystem interactions.

2.1.2 Formal Verification

The provision of cognitive capabilities to an observing meta-system requires at least a verifiable model of the logic involved to reach decisions on system behaviour. This involves verifying correctness for certain deliberative properties where there is no requirement for these properties to constitute a complete specification. In most usual approaches to formal verification mathematical proofs of correctness are provided to previously written programs. This is a laborious and awkward process for most programs, complex systems' programming notwithstanding, leading to program verification being

more often omitted in favour of software testing giving no means of assurance through unpredictable runtime systems or understanding of the program's structure or function. Program derivation, by contrast, develops proof and program in a synchronized manner. A formal, non-executable, specification is established with a practical application of mathematical rules to deliver executable code. The resulting program is then known to be *correct by construction*. The work presented here uses this approach to design and construct both smaller scale and large-scale systems. The same formal specification is applied to both provide assurance, for system control, for a top-down approach and set the bounds on autonomy for the programmatic specification of low-level interacting entities. This provides for direct specification and control of system components, as required, with monitoring and resultant deliberated actions on the environment and application system in response to the emergent evolutionary features emanating from the autonomous components interactions. Figure 2.1 gives a simple overview.

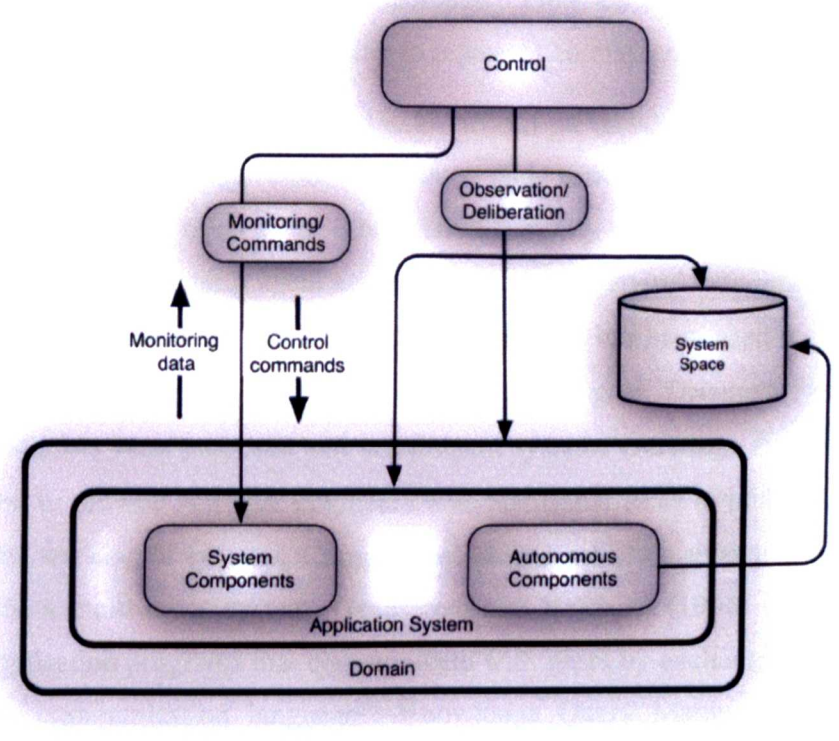

Figure 2.1: Generic Hybrid System Overview

Thus a specification is achieved that gives a programmatic formal verification of boundaries within the system. For instance, assurance of legitimate states is provided by

formally setting the boundaries of behaviour rather than having to explicitly detail all legal or illegal states.

2.2 Agent Approaches to Complex Systems

To enable a high assurance approach to software engineering and endow a system with the deliberative capabilities necessary for autonomous cognitive meta-systems this work has proposed a formal approach to complex system construction and maintenance. Systems are comprised of many interacting components with various degrees of freedom. Some components, for less complex systems or parts of the system, will be completely controlled by the system itself whilst others will be able to act autonomously within given bounds. A major feature of this work is that the specification for the components will not be exhaustive but rather concentrate on merely adjusting the autonomy of the components to specify these bounds. The complexity of the system, and ultimately the necessary techniques to manage the system, will arise from the emergent behaviour engendered by the interactions of these autonomous components. Thus the notion of agent could be construed as central to this thesis in seeking to represent components as agents.

The scientific community has so far failed to arrive at a universally accepted definition of an agent. The problem arises from the many and various characteristics that are associated with the notion of an agent. These are differently emphasised according to the domain of application. The term agent arises from the Latin 'agere' meaning doing, in the sense of something or someone that produces an effect and has been used for upwards of 900 years[1] to denote logical entities. More recently Agent Oriented Programming (Shoham, 1993) has been proposed and multi-agent systems (Jennings and Wooldridge, 1998) adopted to provide software engineers with additional tools to better develop and understand the working of systems. Communication is most often seen as crucial for the interactions in a multi-agent system. Genesereth and Ketchpel (1994) define software agents as "application programs that communicate with peers by exchanging messages in an expressive communication language". Knowledge Query Manipulation Language (KQML) (Labrou and Finin, 1997) and the Federation for Intelligent Physical Agents (FIPA) Agent Communication Language (FIPA, 2003) are widely adopted as frameworks for agent communications.

---

[1] St. Anselm is thought to have set down the earliest recorded modal logic of agency in about 1100AD

It is not the intention here, however, to try and specify what the term agent means or develop any further agent communication language. Rather it will be assumed that the reader has an understanding, in this setting, of what an agent ought to be; a software programmable entity and that communication is achievable through current methods. Thus throughout this work the terms system: component, particle, participant, entity or agent will be most often interchangeable and communication, between them, will be most often portrayed as occurring through a system space, based on distributed tuples. The more appropriate term will be used according to the domain being considered and the associated relevant works in that field.

2.2.1 Agent Based Complex Software Systems

It has been noted that modern software is complex through functionality, management and ubiquity in a distributed setting. This distributed environment means that components will inevitably be required to exhibit more autonomy, as they will be separated, to some greater or lesser extent by some measure, from the rest of the system. In the Logic of Contract Representation (LRC) (Dignum et al, 2003) contracts are proposed to integrate the top down specification of the organisational structures with the autonomy of participating agents. Jennings (2001) comments on the complexity of "industrial-strength" software systems and suggests that a software engineer's role is to provide structures and techniques to cope with complexity. This complexity is not accidental (Brooks, 1995) but is an inherent property of very large-scale systems; adding more resources to a system, although increasing functionality, increases the complexity and the complexity of interaction within the system. Coping with the complexity involves making use of some important generic regularity of properties exhibited by this complexity (Simon, 1996).

- Complexity quite often takes the form of a hierarchical structure with a system composed of subsystems, which are themselves made up of smaller systems. The structure can be assessed at any suitable level of abstraction down to the lowest level atomic subsystem. The nature of the organisational relationships between systems is not necessarily specified but may vary between client/server, peer-to-peer, federated teamwork or swarm behaviour, for examples, at any given time.
- The level of abstraction is a choice based on an observers goal requirements and intention set.

- Hierarchical systems are much more dynamic than non-hierarchical systems of similar size: Complex systems will arise much faster from lower level simpler systems if there are intermediate, identifiable and stable systems through the evolution.
- It is possible to distinguish between subsystem interactions and interactions within subsystems. Interactions between subsystems are more frequent and more predictable. Thus complex systems can be thought of as almost decomposable. Subsystems can be treated almost as if they are independent of the system itself. This however is not quite true as there are interactions between them, some of which are predictable but some that are not.

Thus, to proceed in an agent-oriented way, it is clear that all solutions require a multi-agent approach in keeping with the decentralised theme. Moreover the agents need to interact with each other to achieve personal objectives or to satisfy dependencies and obligations arising from the situated collective environment. The interactions can vary from simple requests (client/server etc.) up to complex social behaviours based on cooperation, coordination or negotiation. This work, while not specifically prescribing an agent framework nevertheless adheres to some general principles that distinguish an agent-oriented approach from other software engineering paradigms. That is, firstly, agent interactions occur through some communicative process based on knowledge representation of system function and state (Mayfield et al, 1995) as against method invocations or function calls, which occur at a purely syntactic level. Secondly, agents can be used as agile situated problem solvers operating in partially observable and unpredictable environments. It is possible to endow agents with the capabilities to make context dependent decisions about the nature and limits of their interactions that could not be foreseen at design time. So adopting an agent-oriented approach to complex software systems mean decomposing the problem into multiple autonomous components that can act, react and interact in flexible ways to achieve the system's functionality. Many agent-based system protocols have been developed that ensure coherent group actions and characterise this macro-behaviour of collectives (Wooldridge and Jennings, 1995) and (Jennings and Wooldridge, 1998).

An agent-oriented approach to software engineering provides many powerful arguments in its favour:

- **Decomposition**: Agent based decomposition effectively partitions the problem space of a complex system. At any given level subsystems work together to provide the functionality of the global system. Furthermore within each subsystem the constituent members work together to achieve the required functionality. Thus the same basic model of interacting components, working towards a particular goal, occurs in all sectors of the system. It is, therefore, entirely natural to modularise the system based on component objectives. To encompass the requirement for distributed control individual agents ought to localise and encapsulate their own control. That is they should be active, maintaining their own thread of control and they should be autonomous, being able to deliberate on their actions and intentions.

- **Interactions**: Complex systems are only nearly decomposable into separate autonomous systems. Interaction is required to fulfil individual and collective objectives. However, because of the system's inherent complexity, it is impossible to know at design time the interactions that may occur. Agents can be deployed with the ability to reason on the nature and range of their interactions at runtime. This strategy of leaving the planning of component/agent interactions until runtime has two major advantages:

    - The usual problems associated with component couplings are in essence removed. Components can be specifically designed to handle unanticipated requests by having adaptability built in through the cognitive facilities. Because an agent framework is knowledge based couplings become a matter of propositional deliberation rather than being subject to the syntactic concerns arising from the types of errors caused by unexpected interactions.

    - The problem of thread control and managing relationships between components is resolved. All agents are always active and synchronisation and coordination is handled from the bottom-up through agents' interactions.

- **Natural Modelling of Complex Systems**: When modelling a complex system the problem is most often characterised by the representation of subsystems, subsystem components, interactions and relationships. Subsystems most

naturally correspond to agent organisations: Constituent components act and interact according to their role within the enclosing system. The interoperation of the subsystems and the constituent components can be most closely modelled through viewing the interactions as social occurrences, where the participants collaborate to achieve some higher-level view (Booch, 1994). Agent systems incorporate the necessary mechanisms for cooperation to achieve a common objective, coordination of actions and negotiation for conflict resolution. The abstraction levels required are available through the agent model, where a collection of components can be viewed as a single conceptual unit. The modelling of collectives is then most naturally handled as a team of components working together to fulfil the collective's goal.

- **Dependencies**: The structure of the agent system is explicit in the representation of relationships dynamically made in the framework. Agent-oriented systems possess the functions to flexibly form, manage and disband collectives. This means that the notion of what constitutes a primitive component can be adjusted according to the granularity desired by the observer. Thus, from one perspective, whole subsystems can be regarded a single components or federations/collectives of agents can be seen as primitive components iteratively decomposing until the system "bottoms out". Additionally the representational structure provides the stable intermediate systems necessary for positive complex system evolution. The availability of such systems allows agents or a collective federation of agents to be developed separately and added to the system incrementally, ensuring a smooth growth in functionality.

2.2.2 Agent–Based Programming Models

Programming models have evolved in response to the needs of the systems being constructed. Distributed computing has established networks of heterogeneous machines of various computational powers. Thus there has been a move away from languages that are dependent on the associated machines' architecture to languages that abstract the problem domain. Agents are a very natural means of problem abstraction: The real world can certainly be viewed as a set of active purposeful agents interacting to achieve objectives. Object-oriented programming (Booch, 1994) and aspect-oriented programming (Kiczales et al, 1997) present similar cases to agent-oriented programming

but abstract the problem domain through different concepts. Object oriented analysis sees the world as composed of objects, which have operations performed on them. Aspect-oriented programming seeks to separate different system concerns into encapsulated aspects with as little functional overlap as possible: The so-called separation of concerns principle. For example, an aspect can alter the behaviour of the base code of a program by applying additional behaviour at various join points (points in a program) specified in a query called a point cut (that detects whether a given join point matches). These approaches, in common with an agent approach, incorporate the principle of information hiding and have first-class mechanisms for interaction. But both objects and aspects are passive in nature; they need to send a message before becoming active entities. In addition objects, although encapsulating state and behaviour realisation, do not encapsulate behaviour modification or action choice. Aspects can perform modifications to the base code but still possess an inflexible structural model, limited to join points to allow runtime action deliberations. Any aspect or object can invoke a public method from another. Once the method is invoked the corresponding actions are performed. In the handling of complex systems it has been found that classes and modules are an essential yet insufficient means of abstraction (Booch, 1994). Objects and aspects are necessarily defined at too fine a granularity for behaviour and method invocation to permit a flexible mechanism of interaction. Design patterns Gamma et al, 1994), application frameworks and componentware (Beneken et al, 2003), although providing additional abstraction techniques, still focus on generic system functions and patterns of interaction that are rigid and pre-determined. Relationships are also difficult to represent without the concept of an agent. In most approaches static inheritance hierarchies define relationships between components, aspects, objects, etc. Much more recent work has sought to address these shortcomings (Miseldine and Taleb-Bendiab, 2006), where, in a similar manner to this work although explicit definitions of agency have been omitted, agent-like entities are used as the principle abstractions. The principle abstraction or agent has its own thread of control so that purpose is localised within the agent and agent selection is encapsulated. The principle of reuse can also be extended to its widest scope. In design patterns and componentware systems, only subsystem components can be reused whilst in application frameworks static, pre-designed interactions are reused. In agent-type programming models, in contrast, whole subsystems, with agent designs and implementations, and flexible interactions, using resource allocation auctions, for

example, can be reused. Additionally legacy systems can be included in the evolutionary and incremental complex agent-system construction. A wrapper can be placed around the legacy code presenting an agent interface to the other software components. Thus, externally, it presents as any other agent. Internally, the wrapper takes requests from other agents and maps them into calls in the legacy code and takes the legacy code's external requests and maps them to appropriate sets of agent commands. Therefore a complex system can grow in an evolutionary manner even maintaining availability, of an existing system, in the process.

In considering complex system participants as agent entities a flexible representation of the domain is established with components capable of autonomous actions and interactions. It is stressed again that there is no attempt here to define an agent any further than as a programmable entity. This programmable entity is capable of autonomous action and interactions within a situated environment. Certain procedures for the agent will be directed and prescribed at design time from the top down. However for an adaptive and flexible complex system many decisions will be delegated to these situated low-level agents/components. The structure and functions of the system will emerge, from the bottom up, driven by the interactions of the situated entities. The top down specification, the rules of interaction (obligations to other agents etc.) and the constraints of the situated environment all contribute to the agents behaviour: Although the behaviour may be autonomous it will still be carried out subject to these rules. The agent's autonomy is always necessarily bounded and subject to normative influence.

2.3 Normative Approaches

A normative structure is required to regulate the behaviour within communities. Thus norms generally arise as a response to the formation of a social situation, to provide rules for a community of more than one individual. For example, collections of software components/agents that may interact have many rules of engagement. Indeed norms can be treated as multi-agent objects (Conte and Castelfranchi, 1995) and classified as one or more of:

- Constraints on behaviour
- Goals
- Obligations

Norms express an idea external to the agent yet the agent is charged with fulfilling the requirement. Thus a norm may take any cognitive form: System norms form the basis of a shared ontology between system participants; behavioural norms form the high level goals in a top down specification and interaction norms result from the social situation in setting parameters for cooperation between participants where an agent might have as a norm an obligation to perform some action requested by another system participant. An agents own intentions can be viewed as a special case of obligation – self-obligation. Normative positions refer to the range of possible normative relations that may exist between two or more system members. Sergot (2001) presents a formal account of such relationships building on Kanger and Lindahl's (Lindahl, 1992) work on normative positions, which combined deontic logic (obligations and permissions) with logics of action/agency to represent complex normative concepts. In general it is these kinds of norms that are of interest in describing and specifying computational systems in general and complex systems in particular. One of the contributions of this work was earlier stated to be a normative bounding of autonomy for system participants. This is one facet of the unifying formalism to specify and reason over complex systems from the top-down and the bottom-up. Norms represent both a top-down specification and a bottom up evolution of role in response to other system member requests/interactions. Deontic logic, as a specification formalism for multi-agent systems, has been advocated as a mechanism for handling norm violations (Boella and van der Torre, 2003). In this work a formalism will be used that specifies system actors (agents) in an abstract manner and verifies and reasons about both system component behaviour and global system outcomes independently of the implementation language used to represent the system participants (agents). Normative agents can reason about norms and obligations and so form a notion of society or federation based on the cognitive concept of obligations giving meaning to the social constructs of cooperation, coordination, trust and reputation. Thus the normative positions of most relevance to this work, in dealing with a top-down specification and bottom-up emergence of function, are respectively permissions and obligations. Permissions are used instead of proscriptions or prescriptions because it is only sought to bound autonomy rather than specify it. This also removes the necessity to be concerned with norm violation, which would again introduce the need for an exhaustive enumeration of the domain. In the usual specifications it is customary to use the deontic operator O that expresses the notion that "it is obligatory". Similarly

permissions are represented by the deontic operator P, which can be stated in terms of O for some statement Ω as $P(\Omega) \equiv \neg O(\neg\Omega)$. Kanger-Lindahl theory (Lindahl, 1992) was defined in terms of *Standard Deontic Logic* (SDL). Sergot's theory of normative positions (Sergot, 2001), however, was not dependent on any particular deontic or action logic. Later in this work the Situation Calculus will be introduced as a highly suitable formalism for the representation of both permissions and obligations in a top-down specification and bottom-up engineering for emergence. Norms most often arise in the consideration of multi-agent systems but also arise with varying connotations in such diverse fields as game theory, logics of obligation and other social concepts, statistics, and distributed artificial intelligence. Various uses and descriptions of the applicability of normative modelling may be found in (Dignum, 1999), (Jones and Sergot, 1993), (Boman, 1999), (Meyer and Wieringa, 1993) or (Moses and Tennenholtz, 1995) for example. In keeping with the proposed propositional account a norm, for the purposes of this work, can be thought of as an implicative sentence where the antecedent is a condition that states the agents position (with respect to some statement) and the consequent is a condition expressing the normative position the agent has regarding some state of the system or environment (Odelstad and Lindahl, 2002). Another feature of a normative position is that of commitment. This generally means an obligation directed from one agent to another. A commitment, c, may be viewed as a 4-tuple c=C(x,y,G,p) denoting a commitment from x to y in the context of G (environment or group) regarding the proposition p (Singh et al, 2000).

This work will show how norms can be more than adequately represented within a formal setting that also allows for system component autonomous interaction and deliberation on non-norm related behaviour. Indeed this is the crux of this work that the introduced formalism specifies both a norm driven system and a system driven by emergent functionality from the actions and interactions of the participating entities. That is a federated collection can be represented and analysed through a propositional account based on the normative positions of the participants with behaviour being entailed by this propositional account. Additionally the actions of the participants and their encounters with each other can be used to engineer emergence and so harness complexity to the function of the system. Thus it is next necessary to consider the process by which teams of components become united into a cohesive federated system.

2.4 Federated Behaviour

The obligation on a subsystem to perform some action amounts to the components, which comprises the subsystem, having a joint intention to see to it that the action is performed. The sum of all the subsystems working at different levels to provide the systems' functions is a major feature of complex systems. Teams of components become hierarchically arranged in dynamic assemblies characterised by a joint intention. The joint action by the team of components, however, is more than a union of simultaneous individual component actions, even if those actions are coordinated (Levesque et al, 1990). A federation is built on the epistemic states of the participants and these states are subject to influence by the group's activities. In earlier works, around this subject, belief-desires and intentions were regarded as paramount in defining the epistemic states of system members (Cohen and Levesque, 1990). Intentions are seen as internal commitments to perform an action while in a particular epistemic state. The commitment is viewed as a goal that persists with time. Thus it is natural to define a joint intention, for a federation, as a joint commitment to perform a collective action whilst all system members are in a shared epistemic state. This is taken as the defining feature of team membership. In addition the individual entities, composing the team, are conceived of: as existing in a situated in a dynamic environment with other entities, as being capable of holding false beliefs and not possessing a complete knowledge of the world, as having goals that are subject to change, having uncertain action outcomes and being subject to external or environmental influence. Thus, in the formation of a coherent team-based solution the notion of joint intention is paramount for the comprised system's function.

2.4.1 Joint Intentions

Joint intentions ought to influence and provide the necessary epistemic structure to set the individual intentions of the system team participants. Typically teams will be involved in joint activities that consist of many separate concerns performed either concurrently or in sequence. Thus a formal setting should show how the joint intentions to perform complex actions give appropriate intentions to the individual components to perform the correct actions. As part of this approach it is necessary to consider the differences between the intentions and actions of an individual and the intentions and actions of a collective. Aggregate agents, for example, will, from an observed perspective possess an individual action and intention set, whilst at a lower level the component members of that aggregate agent will themselves possess individual intentions.

In some ways it is this divergence of epistemic state that provides complex systems with the non-linear reductionist properties previously discussed: If the cases were limited to scenarios where all actions, taken on by the team members, were performed publicly then it would be a relatively simple task to analyse how an agent collective behaves as a single agent because they would share the vast majority of their beliefs. In contrast when agents do not operate in a synchronous manner and in a mutually agreed environment, as most often happens, then there is a tension or added force to the scenario that seeks to preserve some team coherence. The difference between an individual and a group perspective is of an epistemic nature: There must be a concept of mutual: knowledge, belief, intention as well as an individualistic analogue. This then may provide a group cohesion mechanism, roughly stated as:

A and B jointly intend to perform the collective task T if and only if it is mutually known by A and B that T should occur and that each has a mutual belief that they each have an intention to perform their part of the task.

This presents a similar approach to most of the earliest work in this field seeking to represent group dynamics and aggregated behaviour (Tuomela and Miller, 1988, Power, 1984, and Grosz and Sidner, 1990). There are, however, a number of problems with this approach. For instance:

- How can it be specified whether a participant in the task is committed to the task or to the task's parts. It cannot be shown how one participant can be committed to another's actions without stating that the participant intends another participant's action: A clumsy and ill-formed statement. However such commitments are important to enable cooperation and coordination.
- Team structure can be dissolved by participant doubt. Although a persistent intention exists for each participant to perform their part there is no persistence of mutual belief about these intentions.

In order to incorporate these insights into the definition, suppose:

A and B jointly intend to perform the collective task T if and only if it is mutually known by A and B that T should occur and that each has a mutual belief that they each have an intention to perform their part of the task with this mutual belief persisting until the task is completed, found to be unachievable or irrelevant.

Now this definition rules out doubt between the cooperating participants *A* and *B* because each knows the other's position until they reach a mutual understanding that the task is terminated by being completed, unachievable or no longer being relevant. This, however, is now too strong a representation of diverging epistemic states. If A came to believe that it did not possess the capabilities to perform its part of *T* then there is no longer a mutual belief in the activity by all parties. But the defining feature of the aggregated component, *A* and *B,* is the joint intention, kept until believed finished. Under the circumstances of *A* having a private belief of termination, there is no longer a mutual belief that the task is finished. Thus, contrary to an intuitive understanding, there is no joint intention at all. This definition, therefore fails because it demands, from the start, that the participants will mutually believe that they each have their own intentions until it is mutually believed that they do not. It does not allow for the emergence of private beliefs on the status of the task. So when a member of the team come to believe individually that the team goal is no longer achievable it is necessary that the team, as a whole relinquish the goal, even though the epistemic state of the other team members does not support this. Although there is not a mutual belief that the goal is achievable there is not a mutual belief that the goal is unachievable. It cannot follow that a goal is abandoned upon the failure of mutual belief otherwise federations would fail as soon as there was uncertainty about the states of other team members. Alternatively it is required that upon discovering that a goal is completed, unachievable or irrelevant a team member takes on the goal of making this state of affairs mutually believed. Thus before a federation may discharge a commitment there must be a mutual belief that a termination condition holds. Therefore each member of the team must possess a lesser individual goal:

Definition 2.1: *A team member, A, has a lesser individual goal with respect to the remainder of the team and the task T if one of the following hold:*

- *A has the task T as a goal to be completed, yet does not yet believe that T has been completed*
- *A believes that T is completed, cannot be completed or is no longer necessary to complete and has a lesser individual goal to make the status of T believed by the other team members.*

This can be stated, in a simple formalism, with LIG_A being a lesser individual goal for A. p being the proposition made true by the completion of T, $B_A(p)$ being the belief of A

that T has been completed (i.e. A believes p is true). $G_A(p)$ meaning that T's completion is a goal of A. $MB_\tau(p)$ signifying that T's completion (the establishment of the proposition p) is a mutual belief of the federation τ with the usual meaning given to $\Diamond$ (it is possible that) and $\Box$ (it is necessary that). Thus the lesser individual goal, from Definition 2.1 may be stated as:

$$LIG_A(\tau,p) = [\neg B_A(p) \wedge G_A(\Diamond p)] \vee [B_A(p) \wedge G_A(\Diamond MB_\tau(p))] \vee [B_A(\Box \neg p) \wedge G_A(\Diamond MB_\tau(\Box \neg p))]$$

It is also implicit in the definition that $A \in \tau$. Thus a definition of joint persistent goal can be formulated:

Definition 2.2: *A federation, τ, has a joint persistent goal to achieve p if:*

- *There is a current mutual belief that p is false*
- *There is a mutual belief that all members of τ are working for p to become true*
- *It is mutually believed that until there is a mutual belief that p is true, p cannot be true or p is irrelevant all members of τ have p as a lesser individual goal.*

So denoting a joint persistent goal for component A with respect to the federation τ as JPG_A:

$$JPG_A(\tau,p) = [MB_\tau(\neg p) \wedge MG_\tau(p)] \wedge [UNTIL\ [MB_\tau(p) \vee MB_\tau(\Box \neg p)] LIG_A(\tau,p)]$$

This definition has a number of desirable properties that an intuitive account would demand. For instance the case where A is a single agent (a singleton team) reduces to A having a personal goal: If A has a lesser individual goal that persists until the goal is believed to be true or impossible there must also be an ordinary goal that persists. Furthermore:

Theorem 2.1: In a team possessing a joint persistent goal each individual team member also possesses the same as an individual goal.

$JPG_A(\tau, p) \Rightarrow PG(A_1,p) \wedge PG(A_2,p) \wedge \ldots\ldots\ldots \wedge PG(A_n,p)$ Where $A \in \tau = \{A_1, A_2, \ldots\ldots, A_n\}$ and

$$PG(A, p) = B_A(\neg p) \wedge G_A(\Diamond p) \wedge [UNTIL[B_A(p) \vee B_A(\Box \neg p)] G_A(\Diamond p)]$$

Informal Proof: Consider the case of two components/agents $A_1$and A_2. If A_1 has p as a persistent goal suppose at some future time A_1 does not believe that p is true or impossible to achieve. Then there is an absence of mutual belief with A_2 so p must represent a lesser individual goal for A_1. Then from definition 2.1 p must still be a real

goal for A_i and so p persists until A_i believes it is satisfied or impossible to achieve. Similar reasoning applies to all the team members. So if a number of components contract to do something then they individually commit to achieving it.

This account of joint intentions states precisely the dynamics involved in team formation based on a shared commitment to a system state. It predicates that a participant will persist in trying to achieve a goal even if it believes a fellow team member is trying to make known the knowledge that the goal has been abandoned. Although, in terms of optimal local strategies, it is inefficient for participants to persist in behaviour longer than necessary, from another perspective, of global outcome, it is the mutual commitment to this behaviour that provides the federation bindings. The participant always has the belief that team member knowledge of futile actions will always eventually be communicated to it. Additionally if a participant were to quit immediately on suspicion of non-mutual belief then federated behaviour would never occur, as the collective could not be assembled in a robust enough configuration. As previously stated, and illustrated with this treatment of federated behaviour, the dynamics of the system are driven by knowledge: It is the agents/components/participants beliefs and knowledge regarding the properties of themselves and their environment that dictates their goals and actions. Thus it is necessary to consider next how knowledge may be handled within a distributed environment to give rigorous definitions to such concepts as belief.

2.4.2 Modal Logic of Federations and Distributed Knowledge

It has been shown in the previous subsection that federations can be made more robust than a collection of participants in coping with unpredictable occurrences and in failure situations. In situations requiring group cooperation collectives will quickly dissipate through uncertainty. Whereas setting joint persistent goals, as detailed above, defines a team participant and offers a robust configuration to allow team dynamics to happen, even in the face of uncertainty. This means that participants' intentions for enactment are produced not only from their own desires as part of a group but also from a construed team belief. That is a team must have an epistemic structure that supports both individual and group beliefs. The formal methods of investigating the behaviour and performance of a distributed system can be built on a semantic model of knowledge reasoning (Halpern and Moses, 1990). The notion of possible worlds can model the knowledge and beliefs of a component functioning in such a system. More specifically, in this work, it is

intended to consider a multiple agent/component team engaged in providing self-governance functionality, through higher-level cognitive systems, to application level services. The formal setting of the logic required to proceed with this approach is outlined in this subsection. It is based on the *possible-worlds* semantics. This allows reasoning about knowledge. Additionally, in distributed setting, messages passed in the system can be viewed as changing the knowledge state of a system and its individual components. In this way a formal understanding of knowledge may be included in the specification. This is crucial, to this work, as knowledge represents the objects over which deliberation occurs. Hence the dynamics of knowledge acquisition and manipulation, to be used later in this work, need a base theory from which they can be assumed into the proposed specification method.

In dealing with the uncertainty in the knowledge associated with a truly distributed system: At each state of the world, a component believes itself to be in; it has other states that it considers possible: States where all the propositions it believes true are true. So given a statement, φ (say), made up of atomic symbols in a given language, Φ(say), then the agent *knows* φ only if φ is true in all the worlds the agent considers possible. Similarly the agent *believes* φ if there is some world accessible to the agent where φ is true. If conditions are imposed on the possibility relation a number of interesting logics can be captured (Halpern and Moses, 1994). For instance, if no restrictions are placed on the possibility relation, then the normal modal logic of beliefs, K, is obtained. However if it is an axiom that the component can be in situations it considers possible (i.e if S is the set of situations $\forall s \in S$ sRs where R is the possibility relation. That is R is reflexive) then the modal logic T results. Alternatively this axiom may be written $\forall \varphi \in \Phi\ K(\varphi) \rightarrow \varphi$ for an component agent using the modal operator K for *knows* (similarly B is used for believes, which can also be written in terms of the K operator as $B(\varphi) = \neg K(\neg \varphi))$. Likewise if $\forall \varphi \in \Phi\ K(\varphi) \rightarrow K(K(\varphi))$ (positive introspection) then the S4 logic results. This is akin to saying the possibility relation is transitive and reflexive.(ie $\forall s,s',s''$[if sRs' and s'Rs'' then sRs''] and sRs). So for the system participant to know it knows some fact then it must be true in all worlds accessible to the participant and in all worlds accessible from all worlds accessible to it (figure 2.2). Finally another restriction gives the S5 logic. If the possibility relation is symmetric (ie $\forall s$, s' if sRs' then s'Rs), reflexive and transitive then this gives the basic axiom of S5. This can also be stated as:

$\forall \varphi \in \Phi\ B(\varphi) \rightarrow K(B(\varphi))$.

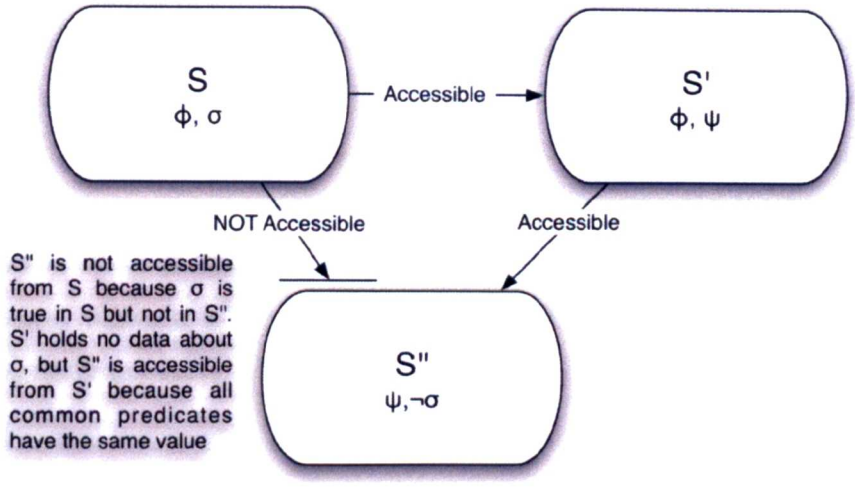

Figure 2.2: Possible Worlds Accessibility Relation

To model the distributed knowledge within a system the syntax used follows the form of most logical representations. So there are:

- A set of primitive propositions,Φ.
- A collection of component agents, n in number, named 1, 2, ..., n
- Modal operators K_1,.....K_n where $K_i(\varphi)$ is read as agent i knows φ.
- $L_n(\Phi)$ is defined to be the smallest set of sentences containing Φ that is closed under conjunction ($\wedge$), negation ($\neg$) and the modal operators.
- The standard logical constructs for OR and IMPLIES are used
    - For two sentences ψ and φ :
        - $\psi \vee \varphi$ means $\neg(\neg\psi \wedge \neg\varphi)$
        - $\psi \rightarrow \varphi$ means $\neg(\psi \wedge \neg\varphi)$
- *true* is shorthand for some valid formula such as $\psi \vee \neg\psi$ and $\neg$*true* is denoted *false*.

Using this syntax it is possible to represent knowledge or belief. The main formal model for *possible-world semantics* is a Kripke structure (Kripke, 1963). For n component agents this consists of a tuple M = (S, π, K_1,.....,K_n), where S is the set of possible

worlds, π is a truth assignment from the primitive propositions and K_i (i=1,.....,n) is a binary relation on the set of possible worlds. So if Φ is the set of primitive propositions then for each $s \in S$ $\pi(s) : \Phi \rightarrow \{true, false\}$ and for agent i $(s', s) \in K_i$ means that in world s agent i considers s' a possible world. The usual order of this relation has been reversed. This is because, in seeking to integrate this approach with the Situation Calculus later in this work, the current situation term is always the final argument and a world is identified with an action history or situation. Also note that each $K_i \subseteq S \times S$. So a binary relation can be defined, between a formula and a pair consisting of a structure and a state (situation) in the structure:

e.g $(M,s) \models \phi$ that is read as ϕ is true at (M,s), for $\phi, \varphi \in \Phi$

$(M,s) \models \phi \Leftrightarrow \pi(s)(\phi) = true$

$(M,s) \models \phi \wedge \varphi \Leftrightarrow (M,s) \models \phi$ and $(M,s) \models \varphi$

$(M,s) \models \neg\phi \Leftrightarrow (M,s) (\neg \models) \phi$

$(M,s) \models K_i(\phi) \Leftrightarrow (M,t) \models \phi$ for all values of t such that $(s,t) \in K_i$

The first three axioms merely correspond to the regular propositional calculus. The final axiom formally introduces the notion of what it is for a component agent i to '*know*' a fact. That is component agent i knows ϕ in situation s of structure M only if ϕ is true in every situation i considers possible when i is in situation s. Now it needs to be asked whether this approach captures what an agent knowing a fact means. That is agent knowledge is defined as the set of valid formulae (those statements that are true in all situations of all structures). So for a structure $M=(S,\pi,K_1,\ldots\ldots,K_n)$ ϕ *is valid in M* ($M \models \phi$) if $(M,s) \models \phi$ for all situations $s \in S$. This leads to results:

if $M \models \phi$ and $M \models \phi \rightarrow \varphi$ then $M \models \varphi$

$M \models (K_i(\phi) \wedge K_i(\phi \rightarrow \varphi)) \rightarrow K_i(\varphi)$

if $M \models \phi$ then $M \models K_i(\phi)$

for formulae $\phi, \varphi \in L_n(\Phi)$ structures M and component agents i=1,.....,n

This constrains the ideas of knowledge that can be modelled by Kripke structures. It can be shown that these constraints are maximal and correspond to a logic K_n that is the multi agent equivalent of the previously mentioned modal logic K with n=1. Similar extensions of single agent constraints, as detailed above, to the multi agent setting lead to logics T_n,

$S4_n$ and $S5_n$, amongst others. The above results are discussed and proven by Halpern and Moses (1994). There is some debate over which, if any, system best captures our intuitive notion of knowledge. It is noted that $S5_n$ captures useful concepts in distributed computing. For example suppose a node in a distributed system (or grid) has received a particular set of messages. There are a number of possible worlds (system states (or situations)) consistent with these messages having been received. Thus the node knows φ if φ is true in all the possible worlds. So the node acquires knowledge but is not required to perform any reasoning and is not necessarily aware of this knowledge. This interpretation exactly matches $S5_n$.

In order to reason about the state of group knowledge it is necessary to integrate the previous subsections team knowledge in terms of the beliefs of the individual agents. The first construct to be applied is an operator to capture the facts (beliefs) that are mutually agreed by the agents (i.e facts that everyone knows or general knowledge). It can be written for a team, τ, of n component agents $A_1,..A_n$ $MB_\tau(\phi) = E(\phi) = K_1(\phi) \wedge K_2(\phi) \wedge ..... \wedge K_n(\phi)$. for $\tau=\{A_i \mid i \leq n \in N\}$. Then the concept of common knowledge of a statement can be modelled by the belief that everyone knows it and that everyone knows that everyone knows it and that everyone knows that everyone knows that everyone knows it.....etc. This is the infinite conjunction: $C(\phi)= E(\phi) \wedge EE(\phi) \wedge EEE(\phi) \wedge .....$, so for $n=1$ $E(\phi)=K(\phi)$. In the language of Kripke structures this can be formally defined as: $E^1(\phi)=E(\phi)$ and $E^{k+1}(\phi)=E(E^k(\phi))$ for $k \geq 1$.

In the case of distributed knowledge it is required to reason about the entire knowledge of the group. So if component agent A_1 believes ϕ and A_2 believes $\phi \rightarrow \psi$ then the belief of statement ψ is distributed between the agents even though no agent individually believesψ. A distributed knowledge operator $D(\psi)$ can be used to specify this fact. So for a Kripke structure $M=(S,\pi,K_1,....K_n)$ the following definitions can be made:

$$(M,s \models E(\phi) \Leftrightarrow (M,s) \models K_i(\phi) \text{ for all } i \leq n$$

$$(M,s) \models C(\phi) \Leftrightarrow (M,s) \models E^k(\phi) \text{ for } k=1,2,....$$

$$(M,s) \models D(\phi) \Leftrightarrow (M,t) \models \phi \text{ for all t such that } (s,t) \in K_1 \cap ..... \cap K_n$$

It can be shown that these axioms provide a complete and consistent specification of these concepts. (Halpern and Moses, 1994). These operators are the general representational forms that will be used later in this work to provide the dynamics of

knowledge, belief and actions, in a suitable formalism, for both the top down specification of behaviour and the reasoning mechanisms for the deliberative cognitive systems providing the analysis and influence of the bottom-up emergent characteristics of the system.

2.4.3 Logical Omniscience

A consequence of the possible worlds approach, as first established by Hintikka (1962), is logical omniscience: A component-agent knows all the logical consequences of its knowledge; if it knows ϕ and φ is a logical consequence of ϕ then it also knows φ.

Logical omniscience is an unrealistic property of a human based knowledge model, as, for instance, it is unreasonable to suggest that all the properties of arithmetic are automatically known from the axioms for constructing Natural numbers (Peano's Axioms). Humans do not know all consequences of their knowledge nor is that knowledge closed under deduction (i.e it is not always the case that $[K_i(\phi) \wedge K_i(\phi \Rightarrow \varphi)] \Rightarrow K_i(\varphi)$). All current formulizations of knowledge, however, have this property with certain techniques to circumvent the problem. This is one of the reasons why, in this work, the observer system is separated from the underlying components of the application system. The observer system only deals in directly observed knowledge and is external to the observed domain being formalised. In particular there is a complete distinction between the observer and the observed components: $K_i(\phi)$ does not mean the observer knows ϕ but rather the observer of the domain can look into the world inhabited by component i and observe that component i knows ϕ. Thus there is no attribution of logical omniscience to the observer, which may indeed culminate in human level understanding of the domain. Rather logical omniscience is minimised through system complexity arising from component-agent interactions that may be hierarchically abstracted to relatively simple models.

The consequences of the observer ascribing logical omniscience to its monitored components can thus be established. The observer is able to formalise a valid sentence about the observed domain and be committed to the logical entailment of this sentence. Nothing in this formalisation, however, implies that the observer is aware of these logical consequences. Rather when one of these consequences is apparent, through the provision of a valid proof, then the observer will assimilate the knowledge. Therefore, taking this standpoint, the observer is not necessarily aware of all the logical consequences of the

system axioms. An approach that stipulates only the direct specification of knowledge (Fagin et al, 1995) allows the specification of the observer system: The observer knows ϕ if and only if ϕ is explicitly represented in the information system or if it can be inferred from existing sentences using a sound but possibly incomplete set of inference rules. Awareness or unawareness of knowledge is used to distinguish between explicit and implicit knowledge. Implicit knowledge is the general account given above and identified as the logic $S5_n$. Explicit knowledge is implicit knowledge with awareness (Fagin and Halpern, 1988). Knowledge of unawareness (Halpern and Rego, 2006) may be represented in a fully formalised logic to accompany the abstraction gained through the use of an observer system.

2.5 Self-Organisation

Federated behaviour represents one way in which a systems function can be delivered. In large-scale complex systems behaviour and function can also emerge, in a non-predictable manner, from the component interactions in the form of emergent self-organisation. This work proposes a unified approach to specify federated behaviour and to monitor for and deliberate on emergent self-organisation. In the previous subsections the definitions and control of behaviour of the collective components, within a system, have been described in a top-down manner. The interactions and behaviour of the participants in the system may be formalised in terms of logical entailment that gives predictability and safety to the system, where all properties may be derived from the specification. It is widely known, however, that participant interaction, within a complex system, can lead to unexpected features and properties of the system, known as emergent behaviour. This can additionally be exhibited as spontaneous organisations becoming apparent within the system architecture or topology. Such phenomena is said to arise from the bottom-up. Functionality within the system results from this self-organisation and from the engineering of this emergence.

As previously stated it is not solely through the aggregated behaviour of the component agents that emergence and self-organisation can be understood. Interactions and the interactions caused by the interactions and so on provide the delicate balance from which emergent behaviour ensues. This interactive behaviour of the low-level components in a system causes the self-organisation to occur at critical thresholds. In many cases the probabilistic models that govern the stochastic actions of the participants will determine

whether a particular form of emergence ensues or not. It is from these interactions that global solutions can be engineered, from the bottom-up, through fine adjustments to the logical model. Additionally, using a cognitive observer system, new emergence can be detected and utilised for future system operation: The action history that engendered a specific global emergence can be captured and reused to engineer the same emergence in future scenarios.

The behavioural rules given to the component agents within a system are the basis of the global emergence of system control, abstracting the role of a system controller. It is the local interactions caused by the: *if (condition) then do (action)* procedure, for each component, which distributes control across the whole system components-members. In turn these rules may be evolvable by a number of methods such as genetic algorithms or adaptation to danger signals. So, for instance, the death of a process may be used to form a receptor that monitors and constrains similar processes to act within safe limits. The properties of complex systems (Holland, 1995) have an impact on the modelling techniques that need to be employed. Aggregation means that a component-agent federation can be considered as a single component-agent. Labelling assigns roles to the various system members. Flow of information and objects is a typical phenomenon of complex systems. Typically energy is used from the environment to increase an object with the resulting entropy being dissipated back into the environment. Nonlinearity is apparent in team functionality being greater than the combined functionality of the participants. Diversity amongst component-agents promotes evolution so that rules of interaction allow modification by other components via labelling and by using genetic algorithms, for instance, to form optimal aggregations. For a complex system to exhibit self-organising behaviour four prerequisites have been rigorously established (Glansdorff and Prigogine, 1978), albeit from a thermodynamic viewpoint:

1. There must be two mutually influencing components within the system (mutually causal).
2. At least one system component must be enhanced by the action of another component (autocatalysis).
3. The system must take in resources from the environment to enhance itself and dissipate the resulting increase in entropy back to the environment (far from equilibrium).

4. At least one of the system components must be accessible to random events from the environment (morphogenetic change).

So in order to achieve the required self-organisation, within the system, to permit system control to emerge, base level architecture and interaction frameworks need to be established. The overall model ought to include both a top-down normative intentional approach to deliberation and a bottom-up distributed control mechanism with methods adapted from pattern recognition, such as Chance Discovery (Ohsawa, 2001) novelty (Magnani, 2005) and danger detection (Aicklen et al, 2003), used to indicate emergence under suitable partial observation conditions.

Self-organisation has recently been considered in a wide variety of fields, stemming from initial exemplars in physics and biology, providing new paradigms of interaction and complex network behaviour through naturally occurring complex systems and large man-made networks of components. A number of works in popular science have resulted in the term becoming quite widely known (Barabasi, 2002 and Strogatz, 2003). Additionally naturally inspired self-organisation is now widely accepted as a field within computer science. This has arisen because, as software systems have become pervasive and ubiquitous throughout everyday environments, the complexity engendered cannot be formalised in advance of the complex event chains and cascading side effects that occur, as the system evolves. Thus, as previously stated, more functionality must be devolved to the participant components, within the system, leading to traditional top-down techniques being less appropriate and more bottom-up functionality emerging from the component interactions.

Naturally occurring self-organising systems are composed of a very large number of interacting entities that cooperate to achieve tasks unrealistic for any one individual component. These systems can scale to any required size, as interaction is local to the participants rather than being staged at any global level. There are a number of self-organisation mechanisms that are exhibited through biological multi-entity (agent) systems. Social insects in engineering self-organisation from their interactions achieve food foraging and nest building feats far in excess of any individuals' capabilities (Camazine et al, 2003). Herding by large mammals, flocking in bird colonies and schooling within large groups of fish are all manifestations of completely decentralised control, with no global directives, which nevertheless results in complex coordinated behaviour being observed for the herd/flock/school as a whole. Typically such systems

use a variety of techniques to engineer the global outcome from the participant interactions:

- **Foraging**: Foraging individuals self-organise their activities to perform a stochastic search through their situated environment (Parunak, 1997). For example, ants lay down a pheromone trail when they are returning to the nest from a discovered food source. The existence of pheromone has an effect on the stochastic decision making process of a perceiving ant; the more pheromone present the more likely the perceiving ant is to follow the trail. The pheromone is volatile and disappears from the environment if not reinforced. This results in old paths to food disappearing when the food source is exhausted. Such techniques can be seen to have immediate applications to computer science related works in network routing, search and optimisation algorithms (Hadeli et al, 2004).
- **Herding**: Typical examples of herding are the flocking or schooling of individual entities/agent/animals into organised regular patterns, in their global environment (Reynolds, 1987). It is self-organising behaviour that may be engineered through the understanding and harnessing of three forces: Collision avoidance, flock centring and speed matching.
    - Collision avoidance occurs where the interacting entities are programmed to be repelled from each other at close distances: They will pull away from each other before crashing.
    - Flock centring relies on the individual participants having a concept of the centre of the flock. Each individual is always trying to move towards this perceived centre without any intervention from a central authority.
    - Speed matching describes the situation where the participants attempt to proceed at the same speed as their neighbours.

  Random pattern formation is a feature of herding, as flocks never attain equilibrium (Toner and Tu, 1999) but may traverse through varying organised structures according to the forces acting. Individuals in herds move less randomly than in foraging scenarios: The rules of interaction prescribe a more tightly bounded autonomy of individuals relative to each other. Applications of these techniques may be seen in motion coordination problems; autonomous nano-spacecraft, for instance, used for unmanned space exploration (Rouff et al, 2004).

- **Nest Building:** Self-organisation can become apparent when swarm individuals cooperatively produce a nest or other complex structure without any centralised master plan. A structure is erected through interaction and simple local rules followed by the participants. These rules typically follow a set pattern: Individuals are in possession of small bricks made from waste matter or debris that is marked with a pheromone. The individuals wander in a random manner but tending towards the strongest local pheromone concentration. At each step a stochastic action is available to the participants: The greater the pheromone concentration the higher the probability that the brick will be deposited. Initially randomly placed bricks appear scattered through the environment. These attract individuals and increase the probability that they will deposit a brick at the same place. The most recent deposits occur at the centre of a pile giving the highest pheromone concentration and so leading to the formation of columns. When the distance between columns diminishes the pheromone scent of each attracts the individuals of the other thus causing the formation of arches. Cooperative robotics is an application within computer science where such group building mechanisms may be of use.

- **Web Weaving:** The weaving of a web is based on the low-level programming task of connecting two locations by a line. The resultant web constructed by this repeated activity, is composed of a network of lines tied to fixed spots. Individuals wander randomly through the environment stochastically connecting points with lines. These individuals however prefer to perform their movement on the lines rather than through or on the environmental medium. Thus it is more likely that where one line ends another begins producing the web effect. The probability of moving on a line has to be finely tuned to facilitate the formation of a web: If the probability is too low then the space is filled with random, unconnected lines giving no web formation. If the probability is too high then participants are trapped on their own lines with no weaving occurring. Once a line has connected two points this becomes the shortest path between these points. Thus this self-organising mechanism has relevance for file retrieval in peer-to-peer systems and network routing. Lines can connect servers, service hosts or resources to allow efficient discovery and retrieval in large-scale computer systems.

- **Moulding:** Moulding occurs most frequently in scarcity situations. It happens when individuals are attracted to specialised food sources or other resources located in the environment (Resnick, 1994). The usual scenario involves individuals seeking food in the environment through random search. As food becomes less frequently found individuals form a cluster that travels through the environment as a single aggregated entity seeking more favourable environmental conditions. When a new favourable environment is discovered the individuals disaggregate and restart the process. Through the moulding phase pheromone diffuses through the individuals neighbourhood attracting other individuals concentrating where the pheromone concentration is highest. These techniques are more generally applied to team coordination and situations where the computational costs of maintaining a collective on a continuous basis is not realistic but a group can be formed in difficult or dangerous conditions. For instance to provide fault tolerance, in a distributed system, when some recognised danger is perceived.

- **Brood Sorting:** Self-organisation, as represented by brood sorting, occurs when objects are sorted into types through some low-level rule set for the participants. This generally occurs in natural systems for items such as eggs. Participating individuals wander randomly picking up and dropping objects according to the number and type of surrounding objects. For example, an individual encountering a large collection of similar objects located with one exhibiting different characteristics will be most likely to pick up the different object and continue moving randomly. It will deposit its object upon encountering an area containing similar objects to the one it is carrying. Such techniques may be used as sorting algorithms, database organisation methods or for virus protection. A swarm of agents can range over system resources, clustering them and detecting outliers.

- **Morphogenesis:** In biology a major subject of research is how the large number of various cells, evident in all life forms, self-organise into a single, coherent, recognisable biological form. Morphogenesis is the emergence of a bodily form from a mass of interacting entities. For example, in many species graduated morphogens are used to determine a cells position and so influence its eventual function. Cells at one end of an embryo emit a morphogen that diffuses along the length of the embryo. Undifferentiated cells ascertain their distance from the

emitting source and so determine their position in the body by establishing the concentration of the morphogen. This technique could be used, with virtual morphogens, to ensure a collective attains some required global shape.

The implications and use of self-organisation and emergence in this work is motivated by two separate yet related considerations: Reuse and detection. Firstly, as outlined above, there are very many naturally occurring models of self-organising behaviour. These can be reused to engineer the required global outcomes from pre-programming the low-level interacting participants. Secondly, through the operation of a large-scale complex distributed system much of the emergence and self-organisation is going to appear in a system dependent manner: The results of the component interactions are not predictable in advance. Thus this work is concerned with formally specifying observer systems, to reason on large-scale system operations and to detect various emergent features that could not be predicted at design time. The system can then be tuned for these emergent patterns and the emergence considered for such factors as recognisability and recurrence. This model of emergence can then be reused as part of the system. This plurality is at the heart of the unified approach presented here. The same formal constructs will be used to provide:

- The top-down command and control structures, where necessary.
- The bottom-up specification to engineer emergence.
- The deliberative capabilities to recognise and influence the system to known self-organising behaviour
- The reasoning faculties to detect and reuse the patterns of actions that engender a certain global outcome for the system to emerge.

Similar principles can be applied to instances of emergent self-organisation exhibited by large-scale complex man-made systems. The best examples being widely distributed computer systems as exemplified by the World Wide Web.

2.6 The World Wide Web

No introduction to complex systems would be complete without a mention of the largest, most pervasive and in many ways the most complex of man-made complex systems: The World Wide Web, as made available via the Internet. Emergent characteristics are apparent in the topological structure of the Internet and World Wide Web. Certainly the

World Wide Web represents the largest, fastest growing system from which topological information can be extracted. In 1999 there were about 1 billion web pages (Lawrence and Giles, 1999). By 2005 this figure had risen to an estimated 11.5 billion pages (Galli and Signorini, 2005) that were indexable. It is becoming increasingly difficult to determine or calculate the exact number of web pages. There are areas of the web that are inaccessible from the outside and there are others sectors that do not have links out yet possess inward links. In April 2007 (the latest figures available) the Netcraft web survey (Netcraft, 2007) found 113,658,468 accessible websites. This, however, only gives a figure for web sites. Neither Yahoo nor Google publicise the size of their index of actual web pages; the last figures available are for August 2005 giving 19.2 billion web pages (Yahoo, 2005). The Netcraft survey for August 2005 found 70,392,567 web sites (Netcraft, 2005). Dividing the 19.2 billion web pages by the 70,392,567 web sites gives the average number of pages per site: 273 (to the nearest whole number). Assuming the average number of pages per site is fairly static; the Netcraft April 2007 figure of 113,658,468 web sites can be multiplied by 273 pages per site average to give an estimate of the size of the World Wide Web. Thus the estimated number of accessible web pages currently on the World Wide Web is 31,028,761,784 (over 31 billion pages). This is, at best, a very rough estimate. There is no centralized control of the Internet or World Wide Web and sites are created and deleted even as counts are being attempted. Thus there are a very great number of web pages interacting with other web pages, via hyperlinks, giving the global outcome of the World Wide Web.

A major observed self-organising behaviour, within the World Wide Web, is the power law degree distribution of the links to web pages (Barabasi and Albert, 1999). Most models of the web, before 1999, were based on random networks topology. The degree distribution of web page links does not conform to such a model. Considering the number of external links to a web page[2]: Not all pages have the same number of links. The spread in the number of links to a page is characterised by a degree distribution $P(k)$, where k is the number of links to a page, giving the probability that a page has k links. If the links between pages were placed randomly, as in a random network, then the majority of pages would have approximately the same number of links, clustered around the mean value of k over all the pages: $<k>$. The degree distribution of a random graph is a Poisson

[2] The same reasoning can be applied to outward links from a web page.

distribution with its peak at $P(<k>)$. Albert et al (1999) reported on the discovery that for a large number of complex systems, including the World Wide Web, the degree distribution is a power law: $P(k) \sim k^{-\gamma}$.

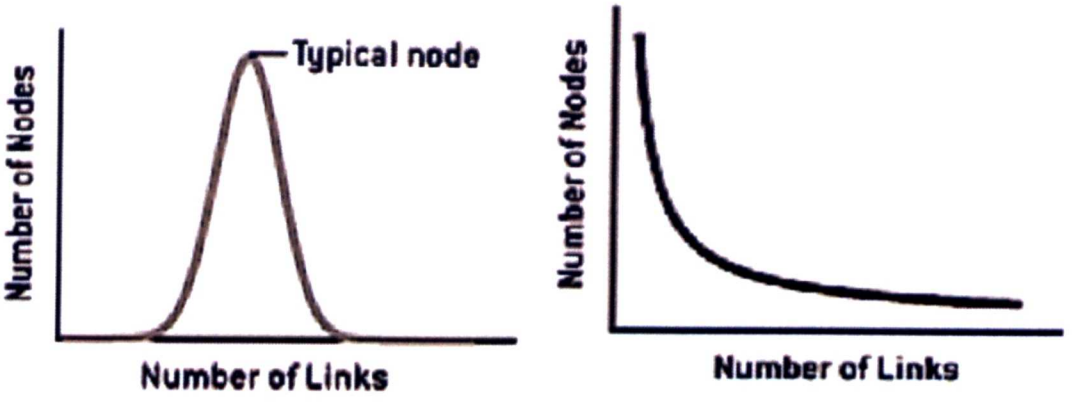

Figure 2.3: The Poisson Distribution (left) and the Power Law Distribution (right) for a Node Degree Distribution

Figure 2.3 shows the very different characteristics of the distributions. The crucial difference is that the power law distribution permits the emergence of a small number of pages with a large number of links. Albert et al (1999) further calculated the value of γ for inward links to be around 2.1, based on a subset of the World Wide Web containing 325,729 pages. This shows an emergent property of the World Wide Web that stems from the low-level interaction of linking between web pages. Behaviour, such as this, is discussed in more detail later in this work; however preferential attachment, where a page is more likely to connect to another page if that page has a higher number of existing links, is a consideration for the interactions that can be engineered to allow power law connectivity distributions to emerge on the web.

This work may also be seen as a general response to the call for a Knowledge Plane for the Internet (Clark et al, 2003). In this a new objective was set to a provide a different kind of network system that can assemble itself, given sufficient high level instructions, reassemble itself, as an adaptive mechanism and automatically detect errors and fix problems as they occur. A knowledge plane is proposed that creates, reconciles and maintains the many aspects of a high level view, which may be specified for the Internet/World Wide Web, and then provides services and data, as required to elements of the system. Much research is underway to address this difficult problem and the Knowledge Plane remains a proposal rather than any technical specification. The work

described herein may be applied to the problem as a formal framework to allow the implementation of such a structure. The knowledge Plane proposal (Clark et al, 2003) includes key attributes essential for the system. These include:

- **Edge Involvement:** It is suggested that much of the knowledge in the system originates and is used by system clients. Thus the knowledge plane is required to be broader than traditional system management techniques to capture the knowledge held beyond the traditional edge of the system. The approach in the work, presented in this thesis, uses an observer system to hierarchically structure and abstract its application level system. Thus any client/consumers of the system are encompassed within the remit of the observer. Furthermore the propositional approach means that knowledge may be inferred from system events rather than always having to be explicitly stated. Techniques will also be introduced later in this work to provide efficient methods of knowledge dissemination and discovery using the perceived topological features of the domain.
- **Global Perspective:** The knowledge plane ought to provide a global perspective including data from outside, inside and from different parts of the system. Again the work presented herein will introduce efficient system management observers that aggregate the knowledge from the specialised regional management necessary for the heterogeneous systems. This will allow different part of the system to be identified as operating different topological models, defining their system boundaries. The knowledge inherent in these sectors will be integrated, at some level in the observer hierarchy.
- **Compositional Structure:** Two knowledge planes ought to be capable of merging their respective activities upon two systems becoming connected. Additionally shared objectives and information cannot be assumed, as some sectors may be required to maintain private data. Later in this work the propositional approach established will be shown to provide the cognitive formalism necessary to allow resolution of conflict between connecting systems whilst the hierarchical nature of the observer systems enables easy integration of system management. The previously detailed account of federated behaviour provides the dynamics of joint intentions. Furthermore the usual methods of information hiding will still be used with purpose localised within each component, encapsulating action selection. The observer is required merely to

reason on the observed outcomes for the global system rather than the informational state of any component.

- **Unified Approach:** The knowledge plane proposal calls for a single unified system with common standards and a common framework for knowledge. Thus it is not enough to provide solutions through distinct mechanisms working bottom-up, which may be loosely tied at the top. Rather, because real knowledge is not partitioned by task a unified approach, although harder to develop, will be more effective in the long term. This is precisely one of the objectives of this work. Later the Stochastic Situation Calculus will be introduced to provide a unified formalism for bottom-up task-based programming and monitoring of global outcome together with a top-down knowledge based approach to influence the system to desirable states and to allow deliberation on perceived global situations.
- **Cognitive Framework:** The knowledge plane needs to:
    - Reason in spite of partial or conflicting data.
        - This work provides methods of partial observation and conflict resolution based on Stochastic Situation Calculus and Partially Observable Markov Decision Problems.
    - Recognise and mediate conflicts in policies and goals.
        - The approach, throughout the work detailed herein, relies on the concept of action histories. Whilst allowing quite a lightweight representation the histories allow recognition of previously observed conditions. The formalism operates on rewarding previously successful policy and goal enactments. Thus a dynamical system is in place to provide cognitive reasoning in conflict situations.
    - Perform optimisations and respond to problems in environments too complex for human intervention
        - Cognitive systems, as formalised later in this work, provide the functions to monitor and respond to whole system outcomes faster and more efficiently than a human operator. As detailed earlier,

the size, complexity and meta-functionality of the latest software systems are beyond the capabilities of human operators to control.

- Automate functions that at present can only be performed by highly skilled technicians.
  - The cognitive techniques developed allow representation, reasoning and learning that permits the meta-systems a level of awareness of their application system and its actions.

It can be seen that the work being presented in this thesis is highly relevant to the World Wide Web and Internet. This is to be expected, as they are examples of highly complex, large-scale systems. Obviously it is not the intention to provide any means of control over these structures but to provide the tools to imbue them and any large-scale complex system with the desirable features as set down in the autonomic blueprints (IBM, 2003) and paradigms of self-*. The method of achieving this cannot be left to simple rule-based structures for all the reasons laid out so far. There must be a cognitive element to provide the systems self-awareness. It is the intention in this work to show how this may be achieved through a propositional approach using logical sentences.

2.7 Summary

This work is seeking to provide a cognitive meta-system to reason and deliberate on large-scale system function. In this Chapter it has been necessary to establish how this relates to complex systems. Firstly the establishment of a suitable standpoint from which to view the dynamics of a complex system was required. A collectivist approach was selected as providing a notion of global outcome from the individual behaviours and interactions of the many components in a complex system. This was further refined to a hierarchical reduction approach, where each new layer in a hierarchy is best expressed in term of the components immediately beneath it in the hierarchy.

Agent technology is a mature and flexible representation and implementation methodology. For this reason this work will use many results from research into agent and multi-agent systems. It is not, however, proposed to investigate agent properties or even rigorously define what an agent means. Throughout this work the term agent will be largely interchangeable with component, participant, team, member, etc. That is an agent is simply a programmable entity that may be (but is not necessarily) capable of autonomous actions. What is evident is that the abstraction of a system to component-

agents is very useful in considering interaction in a system and deliberation on global outcome. The advantage from a programming point of view is that agents are always active entities and agent subsystems with flexible interactions can be reused. Objects, in contrast, can only act upon receipt of a message and reuse pre-designed rigidly constructed modules.

Normative approaches are required for complex systems because all the components, to some extent, are operating within bounds of autonomy. There is a balance to be achieved between setting the level of intervention in the system through governance and allowing local autonomy of action for the system participants. It is necessary that this autonomy be adjustable so that the system can reason on the bounds placed on the actions of its components. These bounds on autonomy, or norms, arise for a number of reasons: They may be set at design time to re-engineer some emergent behaviour, for instance, or they may arise as environmental constraints. It is a vital necessity of any meta-system that the normative position of the system and its component parts is adequately represented.

In view of the logical reasoning approach taken in this work a model of group behaviour is required. The basis for this was set out in subsection 2.4 with joint goals and intentions defined and suitable modal logics considered to represent the system knowledge for the deliberative cognitive systems. An overview of the modal operators was completed and this work will be integrated into the Situation Calculus for a fully formal approach to knowledge, action and intention later in this thesis.

In order to incorporate a bottom-up approach into the work it was necessary to look at how function emerges from interacting systems. The required conditions were considered and a review of some of the naturally occurring systems completed with indications of applications to large computing systems. A general method of formalisation and implementation will be given later in this work that encompasses the specification of these natural systems for: Foraging, herding, nest building, web weaving, moulding, brood sorting and morphogenesis. This permits the engineering of emergence, where required, in computer systems. Additionally the observer system in certain circumstances using the same logic, but for deliberation instead, will be able to reason on novel emergent features and reuse the results.

Finally the complexity of the World Wide Web was noted and certain emergent properties briefly commented upon. These properties will be further developed and

explained later in this thesis. A Knowledge Plane for the Internet was a proposal to apply a cognitive system over the working of the Internet. This work is certainly relevant to the aim s of the knowledge plane proposal and it was shown how this work provides the functions, as detailed in the proposal, that are required for such systems.

The next chapter looks at the meta-systems for giving systems self-awareness capabilities to enable the autonomy to carry on functioning without excessive control governance. The current state of the art is reviewed and the problems of providing such systems are considered in the light of the complexity, large-scale and cognition required for deliberation.

Chapter 3

Self-Governing Systems

The need for efficient methods of specifying and implementing agile meta-systems arises mostly from the complexity of managing large-scale, distributed computing systems. As previously mentioned the functionality, ubiquity and added management complexity contribute to the systems being so unwieldy as to render the monitoring beyond the capabilities of human operators. The original solution, proposed as early as 1968 (Naur and Randell, 1968) was middleware, defined as a layer between application software and system software. This abstracts the complexities of the system, allowing the application developer to devote maximum efforts to the problem being solved without the concern of managing the overall system. In this role middleware is related to providing support for naming (for efficient service calling), service location and service discovery (allowing decoupling of system components and dispensing with hard-coded service references), transport (to permit transparent communication between distributed nodes) and binding (to provide a linking between locally executing code and external code). As distributed systems became more prevalent additional functions such as replication, storage, concurrency control and failure handling of system faults became part of middleware systems. In this way the heterogeneity of the underlying systems and inter-process communication is hidden from the user and application developer. Many realisations of autonomic type functions rely on the adaptation of middleware: A brief review of adaptive middleware is included in Appendix 1. This chapter proceeds with a review of IBM's autonomic initiative leading to what has ensued in state of the art autonomic systems. It is observed that adaptation has so far been restricted to mostly being of a passive nature with some parametric adaptation. Some formal systems for specifying systems' self-governance are given as background to this work and subsequent engineering of self-organisation in middleware is assessed. The need to handle emergent organisation in a specification method for self-governance in large-scale autonomic systems is recognised and the Signal-Grounding problem is identified as presenting the major barrier to accomplishing such systems' self-governance.

3.1 Autonomic and Self-* Systems

Autonomic computing systems are generally thought to exhibit self-: Configuration, Healing, Optimisation and Protection; the so-called self-* properties. The paradigm of Autonomic Computing is based on a natural biological model of system regulation. The autonomic nervous system, in the body, is responsible for providing non-conscious control of vital bodily functions. This allows humans, for instance, to proceed with everyday high-level activities whilst the underlying core functions of the body are handled without conscious effort. So the heart beat rate, the amount of air taken in, determined by the expansion of the lungs, for breathing, responses to hot and cold and many other functions of the body are performed automatically with no human interference or consciousness required. The IBM initiative for autonomic computing originated through a presentation by IBM's senior vice president for research (Horn, 2001). This stated that the major challenges facing the computing industry were complexity and Total Cost of Ownership (TCO). The solution presented was called Autonomic Computing and comprised of eight high level goals for computational systems:

- Possession of System Identity: There ought to be a detailed knowledge of the system and each individual part as it is reduced to component level, including its current status, capacity and connections.
- Self-Configuration and Reconfiguration: System setup and dynamic adjustments to account for environmental changes must occur automatically. Adaptive algorithms are a suggested means to achieve this through the learning of best configurations to produce mandated performance levels.
- Optimisation of Operations: The system will monitor its performance and that of its constituent parts and fine-tune its workflow to achieve predetermined goals. Feedback control mechanisms and control theory are highlighted as possible techniques to respond to internal metrics and control all system parts in a unified manner.
- Recovery from Malfunction: Problems or potential problems must be diagnosed and alternative ways of using resources or reconfiguring the system may be used to maintain a smooth running operation. Initially it is suggested that healing rules

will be predetermined at design time. The goal, however is to allow intelligent systems to discover new rules through runtime.

- Self-Protection: The system must detect, identify and protect itself against various types of attacks and threats to maintain overall system security and integrity. Artificial immune systems are mentioned as providing a detection mechanism for suspicious code that can be isolated and analysed centrally with a cure distributed to the entire system.
- Self-Adaptation: This is identified as self-optimisation turned towards the outside of the system. An autonomic system will find and generate rules on how best to interact with neighbouring systems and the environment. Grid type systems are suggested as a means of connectivity that allows this environment aware ability.
- Independence from Proprietary Solutions: The system must function in a heterogeneous world. It must coexist and interact with different systems through the implementation of open standards. Research into intermediary broker agents to arbitrate resource conflicts is a suggested way forward.
- Complexity Hiding: The computing and software technologies needed to achieve a user's goals should be implemented without the users involvement. The system however must anticipate user requirements and be ready to handle the requests in a similar way to the body being prepared for action by adrenalin without any conscious effort by the individual.

Initial reactions to the autonomic initiative sought to establish the novelty of such an approach: Artificial Intelligence researchers have for many years devoted a large amount of effort to studying systems adaptability, self-awareness and environmental consciousness. Likewise best practice in software engineering ought to be the provision of robust and agile computing systems responsive to user requirements. IBM's Autonomic Computing provides an up to date list of desirable meta-system properties and systems displaying these autonomic characteristics define the current state of the art in meta-systems for self-governance.

3.1.1 State of the Art in Autonomic Systems Research

The establishment of Autonomic Computing as the de facto standard in meta-systems for self-governance has encouraged a lot of initial research efforts to target Autonomic

computing from the various research areas enveloped by the definition. The major construct used is that of a feedback and control loop as shown, very simply, in Figure 3.1.

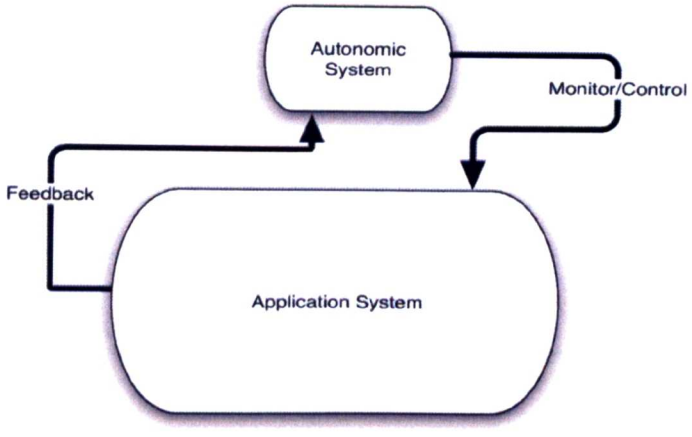

Figure 3.1 Autonomic Feedback/Control Loop

Feedback control is well established as a scientific principle (Laws et al, 2003) and (Kuo and Golnaraghi, 2003). It has been used in many areas of application from robotics to management science to promote a predetermined model of system stability. System stability is achieved through negative feedback in most cases: Some fault state is detected in the system and is relayed to the meta-system, which determines the actions to take to adapt the system so that it returns to its stable state. These control loop adaptations can be classified into three categories (Tianfield and Unland, 2004): Passive adaptation, parametric adaptation and mission oriented adaptation:

- Passive Adaptation arises simply from the addition of a control loop. Simple rules will determine the operation of the loop. A particular metric will be monitored in the application system. When this metric exceeds pre-defined limits then a rule in the meta-system is triggered and the actions dictated by this rule are performed on the system. This, in effect, is the current state of most autonomic implementations relying on simple rule/policy based operations.
- Parametric Adaptation involves a feedback control loop operating on the effect, within the system and environment, of an existing passive feedback control loop as a parametric adaptation loop. This means that the parameters within the passive loop can be adjusted or tuned through the system's runtime. Thus changes in the system's operation or environment result in an optimisation of the parameters in the feedback/control loop. A generic example is shown in Figure

3.2. This represents the current state of the majority of research in autonomic or self-governing systems (Steiner and Hagner, 2007), (Deussen, 2006), (Baresi et al, 2006), (Miseldine and Taleb-Bendiab, 2006) and (Garlan et al, 2004), for example. The tuning of the control parameters is still a very difficult problem to solve through a runtime system. Especially as new parameters and associated feedback/control loops may become apparent to an external intelligent observer that would not be appreciated in a rigid policy based automated system.

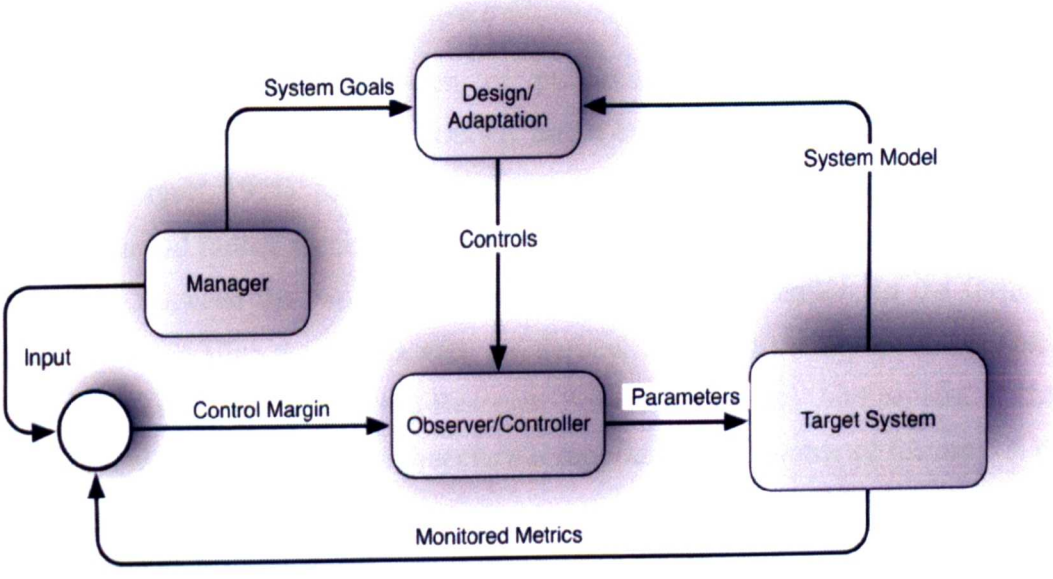

Figure 3.2: The Parametric Adaptation Process

- Mission-Oriented Adaptation can be likened to control loops on control loops on control loops and so on, arranged in a hierarchical structure, culminating in an intelligent observer. The environment of a complex system is most often subject to considerable change over many parameters. The function of the system can be engineered to emerge from the interactions of components/parameters or a function may emerge as a runtime property. In any event the adaptation required to monitor and influence such systems requires the appreciation of the bottom-up nature of the programming model promoting such behaviour. Although the problem has been recognised (Zhu, 2005), there is little support at present to provide a model of self-governance based on mission-oriented adaptation or for reasoning about emergent behaviour generally in multi agent type systems.

3.2 Roles and Normative Positions for Self-Governance Middleware

Components in a programmable agent model can be used to provide functions for self-governance in the middleware for autonomic computing (Randles et al, 2005, Padget, 2003), through parametric adaptation: Agent architectures can be specified to carry out autonomic roles, based on system norms. The usual blurred distinction between component/agent and system architectures will be maintained through this section; although, more precisely a component/agent architecture describes the implementation of individual agents whilst system architectures arise when the component agents are connected through some communicative medium.

The most widely used mechanism of providing components with autonomous capabilities, in such scenarios, is through Belief-Desire-Intention (BDI) type systems (Rao and Georgeff, 1991). A component/agent possesses specified beliefs about the state of its domain and further specified desires regarding what state the domain ought to be in. A mismatch between the two specified notions triggers an intention, in the form of a prepared plan, to align the beliefs with the desires. The most usually cited example of such a system is the Intelligence-Resource-Bounded Machine Architecture (IRMA) (Bratman et al, 1988). Here the architecture is supported by a number of features: A plan library representing a subset of the component/agent's beliefs (Completion of a given set of actions causes a given situation to become true); explicit representation of the beliefs desires and intentions; a reasoner for reasoning over the domain; a means-end analysis process to map intentions to plans; an opportunity analyser to scan the domain for perceptible changes and a filtering process for determining the results of means-end and opportunity analyses that are compatible with existing plans. A number of problems have been identified in taking such an approach, including:

- It is very difficult to ensure that a sufficiently rich model of the domain can be gained by the component/agent: Translating a real domain perspective into a symbolic representation cannot always be guaranteed to give an up to date representation of the actual state of the domain.
- Timely reasoning on the domain state cannot be guaranteed: by the time reasoning occurs the system state is highly likely to have changed in most real world systems.

- There is no provision for intention reconsideration in the light of changeable, dynamic environments.
- There is no cooperation or coordination mechanism for the component/agents: The entities act according to selfish rules.

There have been numerous proposed extensions to the BDI model to include normative behaviour with obligations and thus cooperation and coordination in multi-agent/component systems. For example:

- The Belief-Obligation-Intention-Desire (BOID) model (Broersen et al, 2001) includes a specific specification of obligations with the component/agent type defined by the precedence the component/agent gives to the modal attitudes.
- The Epistemic-Deontic-Axiologic (EDA) model (Filipe and Liu, 2000) is based on organizational semiotics with normative roles for the participants classified as belief, obligation or value system norms.
- The Extensible Belief-Desire-Intention (EBDI) model (Randles et al, 2005) classifies the intentions to promote coordination and cooperation through normative, reward and utility intentions.

The Viable System Model (VSM) (Beer, 1979) has also been proposed as providing a model blueprint to enable a specification of system self-management through middleware, based on BDI (Laws et al, 2003). The VSM considers any viable system to be composed of 5/6 interacting systems, as shown in Table 3.1.

The power of the model arises from the recursive nature of the specification: Embedded instances of a VSM model can occur in the S1 units of a particular system. Thus the operational and management S1 units can be developed as VSM models, making the system structure open ended in both directions. This hierarchy may be followed upwards to more encompassing systems or downwards to smaller more specialized functions. This notion of hierarchical recursive systems will be followed in the description of the Observation System, used later in this work.

Other organizational models have been proposed, for example:

- Gaia (Zambonelli et al, 2003) specifies organizational normative roles with interaction through four attributes: Responsibilities, permissions, activities and protocols. It incorporates an interaction model, through a set of protocol

definitions, which describes the dependencies and relationships between different roles in the system.

- AALAADIN (Ferber and Gutknecht, 1998) is a meta-model to define organizational models. It is based on three core concepts agent, role and group. The latter concept ensuring cooperation and coordination through enforced modularity allowing task decomposition.
- The ROPE project (Becht et al, 1999) similarly defines roles in an integrated environment, termed the Role Oriented Programming Environment, to participate in collaborative task accomplishment.

Table 3.1: Overview of Viable Systems Model (Laws et al, 2003)

Thus it can be seen that there are very many methods through which normative control of systems may be established. Centralised controllers, within the middleware, observe the

system and enforce normative behaviour on the system participant components, either through strict monitoring or by the pre-programmed model adherence of the components. Through all these specification systems, however, there is a complete lack of a general formal logic system for specifying and reasoning about emergent behaviours or self-organisation with the systems. Thus to gain a unified formal specification method for self-governance and autonomic systems it is necessary to further consider what needs to be included to handle emergent self-organisation.

3.3 Self-Organisation and Self-Governance Middleware

Self-Organisation in middleware for Autonomic systems and self-governance systems, in general, is a very important concept given the huge complexity, number of interacting entities and the emergence of properties from the bottom-up: Middleware encapsulates methods for coordinating processes and organising data, yet current approaches cannot scale to large complex systems. This has provided the impetus for bio-inspired design paradigms. These most often rely on stigmergy (Bernon, 2006). Middleware in general has been treated as applicable to a number of application areas (Mamei et al, 2006): Grid computing, Coordination systems, seeking to handle interaction mostly through distributed tuples, caching and replication and pervasive computing are areas of application where middleware is playing an increasingly necessary role in providing the bio-inspired meta-systems required to handle the self-governance of large scale systems. Each of these will be briefly reviewed in the following subsections with relevance to the engineering of self-organisation in the middleware self-governance applications.

3.3.1 Grid Computing

Grid Computing has been seen as a solution to managing large computer systems for a number of years. Condor (Litzkow et al, 1988) was proposed in the late 1980s to usefully employ idle computational time in a network. Grid computing, as a concept, was envisaged as a means to seamlessly integrate heterogeneous systems. Mechanisms to handle resource demand, conflict resolution and allocation are required so that the computational assets are well utilised, data is placed in the locations where it is most likely to be needed and services can be easily found, composed and deployed. This is also required to occur in highly dynamic environments where demand or resource availability cannot be predicted. Many users may log onto the system at the same time, demanding a particular service, or the amount of free computational power may be limited by a lack of

idle nodes. This makes computing resources a commodity in the Grid: Users do not need to know how the system works but rather what it is they need from the system and how much it will cost. In terms of engineering self-organising behaviour to provide functionality in this regard, the natural models, briefly described in Chapter 2, can provide new mechanisms to achieve resource utilisation in Grid systems.

- Foraging may be used for optimising the distribution of services (Andrzejak et al, 2002) in a Grid: Pheromone-like trails can be used to positively reinforce the successful service discovery processes and to ensure the abandonment of obsolete service location data. The placement of resources can also be achieved based on the strength of demand abstracted as a pheromone reading.
- Herding can be seen to have similar applications for service placement (Gilbert et al, 2006). Collision avoidance can ensure replicated or similar services are not located with too much redundancy, Flock centring can optimise the location of services; dynamically adjusting the "centre of gravity" to demand fluctuations. Speed matching can be utilised to balance loads across the Grid.
- Nest Building, similarly, can ensure the provision of adequate resources tailored to demand (Bonabeau et al, 1999).
- Web Weaving is a mechanism that permits new shorter paths to connect resources, nodes, etc (Bourjot et al, 2003). Thus, in Grids, more frequently assembled service compositions may be autonomously organised through a web-weaving algorithm.
- Moulding and Brood-Sorting can be used as an approach to organise grid resources. Moulding can prevent the aggregation of data in a single location on the grid. As the cluster grows towards becoming a bottleneck, in the system, moulding can trigger a differentiation, through brood sorting, to cause the formation of new similar clusters to be distributed across the grid.
- Morphogenesis also provides some powerful tools to engender a contextual concept within the grid and may be utilised, in future work, for self-awareness. This will be dealt with in a little more detail later in this work.

3.3.2 Co-ordination systems

Tuple-based coordination models, such as Linda, provide a useful method of exposing system participant states for use by other system components (Gelernter and Carriero. 1992). A structured set of typed data items form a set of tuples that can be indirectly exchanged between the system participants. This occurs through a shared tuple-space. A tuple can be written to the tuple-space or retrieved via matching queries to data items in a tuple. In this way coordination between system components is handled in an uncoupled manner with actions coordinated on the basis of the tuples present in the tuple-space. Commercial middleware platforms, such as Jini (Arnold et al, 1999) or GigaSpaces (G.T. Ltd., 2002) implementing JavaSpaces, are based on Linda distributed tuples. These however, have not been widely used in large-scale systems. For larger scale systems either a centralised single server is used to accommodate the tuple-space or tuples with common characteristics are grouped together on specific servers or complete copies of the tuple-space are placed on a number of servers or data is stored on a grid of nodes formed by logically intersecting busses. Each node is part of exactly one inbus and one outbus. Tuples are written and replicated on all nodes of the outbus whilst retrieval comes from the inbus. One inbus intersects all outbusses giving a complete view of the tuple-space. SwarmLinda (Menezes and Tolksdorf, 2004) brings principles from known self-organisation models to allow a scaling up of tuple-spaces for large-scale systems. Brood sorting is used to distribute the tuples, dynamically partitioning the tuple-space into clusters of tuples. This ensures wide availability without having to perform computationally costly procedures like replication. Tuples are retrieved by foraging mechanisms: an executing process causes templates to search for matching tuples in the tuple-space. Once a tuple is found it is relayed to the calling process and a pheromone trail left so that subsequent templates seeking a matching tuple may also easily discover the location. This results in the emergence of application specific paths between tuple producing and tuple seeking components. The volatile pheromones disappear if not reinforced meaning that paths dynamically adjust to system changes.

3.3.3 Caching and Replication

Caches minimize latency for network requests. In client/server architectures client side caches act as server proxies receiving data requests. When the data is transferred to the client a copy is held in the cache to be available for a different client making the same request. This relieves load on the servers and minimizes response time. Similarly server

side caches aid load balancing by storing the results of processing in the cache for retrieval obviating the need for further processing upon receiving the same request again. In both these cases cache sizes are obviously limited so the management of the cache's content has important ramifications on systems' performance. It is obviously best to populate the cache with the responses to requests that are most likely to be received. If food foraging techniques were used then the entries in the cache (ants) that received a sufficient number of requests (food) would remain in the cache whilst those entries that received insufficient requests would die out to be replaced by new entries. This can be done adaptively by the cache entities using artificial pheromones to indicate the locations of resources. Newly created entries can follow the trails to predict what resources may be required in the future; creating relevant new cache entries (Floyd, 2007).

3.3.4 Pervasive Computing

The increasing pervasiveness of modern computing systems is one of the identified forces creating the need for adaptive middleware and system self-governance. Applications have to be adaptive as system components can be deployed at any time across multiple domains. In addition the performance of a software system is often dependent on its environment of application. Thus a dynamically enforced awareness and subsequent adaptation to context is required. Current middleware infrastructures, for the reasons previously detailed, do not support these requirements. In addition application component-agents are typically strictly coupled in interactions making dynamic runtime compositions, cooperation and coordination for spontaneous interoperations impossible. This adds to the application and environmental complexity that, together with increasing functionality, provides more driving forces for the provision of adaptive middleware. Field based coordination is the usual method of coordination in open dynamic component application systems. Distributed data structures enable component interaction through the mediation of fields, shaped according to the requirements of the system to permit a selection of coordination patterns to match application goals. This field based coordination displays self-organising attributes such as moulding and morphogenesis. Distributed tuples may be propagated across a field with the tuple values subject to alteration based on actions. For instance, a particular tuple value could increase as it moves through a field. Any component-agent encountering the tuple could ascertain the location of its source (commissioning process) by interpreting the value in the tuple. In such a way component agents can operate by receiving contextual data from the

distributed tuples. Such techniques are used in Co-Fields (Mamei et al, 2004) and TOTA (Mamei and Zambonelli, 2004) middleware.

3.3.5 Autonomic Self-Governance and the Emergence of Self-Organisation

It can be seen that self-organisation applied to middleware applications provides very many useful functions for system self-governance. Indeed the coordination incorporated into the Observer System, to be described later in this work, makes use of the distributed tuple space. Additionally the specification, for engineering such emergence, can also be achieved through the methods outlined in this work, though the adjustment of system parameters and probability; such a specification, based on finding solutions to a Markov Decision Problem, set in a food foraging scenario, is also detailed later in this work. The specification of the self-governance in such cases is nevertheless still ultimately dependent on parametric adjustment/adaptation to engineer the self-organisation. In many cases, as systems scale up and become much more complexly interwoven, features emerge from the system that have no precedence and have no possibility of prior prediction. It is in these cases that mission-oriented adaptation is called for if true self-governance is to be achieved. To handle such occurrences a cognitive observation system is required. The present state of the art in autonomic systems relies on parametric adaptation. Yet, in these cases, the meaning of the observed signals emanating from the system and the engendered actions has no relevance to the automated system controller: Self-governance is provided in such cases by blind adherence to rule sets and policies. To reason adequately for mission-oriented self-governance, to efficiently utilise and adapt to emergent outcome, the cognitive observation system must have an awareness of the intrinsic meaning of the events and actions within the system: Signals observed in the system must be known for the affect they have on the system, a grounded definition of the signals is required.

3.4 The Signal Grounding Problem and Autonomic Systems

Known emergent outcome is extremely effective and useful in engineering macro-scale behaviour from micro-scale interactions. For the reasons outlined in the previous section, however, in the course of a systems runtime operation emergent features will almost certainly arise that cannot be predicted at design time. Some of these will be desirable emergent features that would aid the system in its current or future operation whilst others may need to be proscribed to prevent a current fault recurring in future operations.

As outlined above, current efforts in modelling and implementing autonomic systems and adaptive middleware for behaviour control are achieved using policy and/or norm-based management models. Even in the case of emergence engineering, the component interactions are prescribed or proscribed at design time. In these systems rule-based stimulus-reaction (sensor-actuation) constructs are often programmed as conditional triggers or "hot swapping", in the case of providing facilities for systems' self-healing (Appavoo et al, 2003). This is typified in many of the proposed autonomic systems' blueprints (IBM, 2003). The systems are underpinned by this "sensor-effecter" mechanism that facilitates context awareness, based on input from distributed tuples, for instance, leading to reactive autonomic behaviour (Ganek and Corbi, 2003). The rules, norms or policies are specified at design-time as in (Badr et al, 2004) and encoded into rule-based systems through, for instance, object-oriented components or XML policy documents. Other approaches use methods based on semiotic frameworks (Stamper, 2000) to distribute data through the application domain. Although, as previously stated, these suffer from runtime novelty handling problems; more recent approaches are now adapting machine-learning techniques to support novelty detection and runtime chance discovery (Magnani, 2005). Despite these many varied approaches meaningful introspection and accessibility of evolving regulatory models, of adjustable autonomic systems, are theoretically difficult to achieve as they quickly come up against the Signal or Symbol Grounding Problem (Harnad, 1990), a problem more usually encountered in psychology. Here the limits of a purely symbolic model of cognitive function have been reached leading to a more connectionist approach to cognitive modelling. The semantic interpretation of a formal symbol system must be intrinsic in the system rather than merely being assigned a meaning by some conscious process. That is an interpretation of the meaning of sensing results as signals, or their associated semantics as symbols, in the specification a truly agile cognitive observation system for self-governance, need to be programmed in and adapted throughout their lifetime, to achieve system the self-governance with an absolute minimum of human intervention. Many current autonomic systems implementations, for self-governance, whilst aiming at self-adaptation, only provide for the restructuring of the software models with any adaptation or evolution, of their control model specification and encoding, most often achieved by expert human operators. However, although autonomic systems, on their own, do not need to be capable of self-adaptation they do need to be adaptable. For instance in the administration of the

stasis between sympathetic and parasympathetic responses threshold adjustments are required (Miseldine and Taleb-Bendiab, 2005a). Knowledge of the meaning of a perceived signal or observed symbol, encountered in a specific context grounds that symbol/signal to provide an inferential mapping from sensor to actuation. In applying more recent psychological models of cognitive process not based entirely on input/input and input/output associations (Turkkan, 1989) to the cognitive meta-systems required; it can be observed that it is impossible to provide an adaptable autonomic response system without the system itself ascribing meaning to the symbols and signals in the system. It is for these reasons that signal grounding is vital to systems engineered for the autonomy that autonomic functions provide (Baillee, 2004)].

3.4.1 The Signal Grounding Problem Explained

Symbol grounding, whilst being an open question in information theory (Floridi, 2004) is vital in order to provide meaning to the symbols and signals (system events) that emerge in systems operations. An original formulation was given in (Searle, 1980) with "The Chinese Room Argument" as a problem of intrinsic meaning or "intentionality". In this a person is imagined locked in a sealed room with the facility to receive Chinese symbols, which may be manipulated, according to a set of rules, based on the symbol's shape. Chinese symbols are then output from the room. The person receiving the symbols does not understand Chinese but performs actions based on the symbols' shape not meaning. However the symbols are systemically interpretable as having meaning but this meaning is not intrinsic to the symbol system itself. Hence the meaning of symbols is outside the symbol system. So this does not form a viable model for symbol recognition in a cognitive system. That is cognition cannot be solely symbol manipulation. Signals require grounding within the context of a cognitive system so that the symbols have meaning within the cognition of the system. This means that the system must have a zero semantical commitment condition (Taddeo and Floridi, 2005) so that: *There is no innate semantic resources supposed and there is no access to external semantic resources.* The system ought to use its own computational resources to perceive (through sensors and instrumentation) and perform operations (through procedures and effectors) to ground its symbol set to interpret system signals. For autonomic systems this requires an associated observing cognitive system to assess the triggers for autonomic responses for both adaptation and discovery. In this way the meaning of symbols and signals is grounded through cognitive systems.

3.5 Summary

The current state of research, in providing systems' self-governance through middleware, is largely based in autonomic computing. Autonomic computing ought to provide a holistic approach to computer systems development that copes with complexity and adds a new level of automation and dependability to systems through self-healing, self-configuring, self-optimisation and self-protection functions. Additionally it should encompass the automation of system adaptations for computing systems. This is why autonomic computing and adaptive middleware have become almost synonymous. Yet current realisations of the paradigm merely add complexity and scale to an already complex and usually large system. This is because adaptation is generally only conceived of at a passive level. Although passive adaptation is a powerful tool in the establishment of a control loop, all transitions of the system have to be allowed for at design time and the system rendered in a top down fashion. In contrast parametric adaptation allows for system tuning at runtime by the adjustment of parameters. This represents the current leading research in this field, including the engineering of emergence for pre-defined outcomes by the adjustment of probabilities in the low-level component interactions. This is still predominantly a top down approach, for the adjustment of thresholds, although pre-conceived bottom up functionality may arise from engineered emergence. It is only through mission-oriented adaptation that truly self-governing systems can emerge. Autonomic computing takes elements from a number of disciplines to provide solutions for self-governance; feedback control, adaptive control and artificial intelligence, for instance. Computational systems however differ substantially from conventional feedback/control systems: Conventional feedback/control methods are generally applied to physical systems compliant with physical laws. Computational systems, in contrast, are artificial societies not generally governed by physical laws but rather by goals, norms and policies set by humans, social imperatives or environmental constraints. As a result it is a little artificial to directly apply conventional feedback/control loops, based on continuous parameters, to autonomic systems where evolution occurs through discrete events. Specification of roles and norms can go some way towards achieving self-governance, and is indeed a vital element to be included in any specification method. What is additionally required, however, is an associated cognitive system to harness the frameworks and methodologies of feedback and adaptive control rather than the rigid algorithmic procedures of physical feedback/control systems.

In mission-oriented adaptation the system is guided through self-awareness towards its high level goals. This includes detecting, reasoning about and acting on newly discovered emergent global system features relevant to the mission of the system. This can only be achieved through signal grounding requiring higher-level cognitive reasoning by the meta-systems. This work seeks to present a formal semantics for the event-situation-condition-action sequence, via a suitable formalism utilising many concepts from various disciplines, to formalise the "adjustable" governance models for autonomic software behaviour and system evolution, to ground the signals or symbols emitted from a run-time system and its environment. This requires the establishment of cognitive observation systems. These cognitive systems will be arranged hierarchically ultimately culminating in human or user level intervention. An observer/controller monitors a running system that may be an aggregation of further observed running systems. In this way control is passed up through the system to a level with the appropriate knowledge to ground the perceived signal. Final responsibility resides with the human level participants. Thus a holistic approach to systems self-governance and the formal specification of adaptive middleware, including autonomic computing functions, will be provided through a proposed single formulation. This formulation will provide the specification for smaller scale (component) systems through norm-based deliberation as well as the formal models for large-scale complex systems through a cognitive observer model. The next chapter considers the features of the various classes of formalisms necessary to construct an appropriate specification; reasoning and evolutionary language to provide a holistic unified scaleable solution to systems' self-governance.

Chapter 4

The Specification of Complex Systems

The representation and reasoning about computational systems can take many forms. One aim of this thesis is to provide a method to allow the robust design of meta-systems. This includes both the more usual top down approaches for rule-based or norm-governed behaviour through specific policies and the less easily achievable bottom up approaches, for more complex systems, of engineering emergence and responding to unpredictable system events. This chapter reviews the classes of representation and reasoning formalisms available to provide the attributes that are most desirable in modelling the complex adaptive middleware meta-systems required. This leads to recognition that the Stochastic Situation Calculus provides a unique combination of relevant properties, such as support for counterfactual reasoning and no prior state space enumeration, in a single formal specification method. Thus the chapter concludes with a detailed overview of the Situation Calculus approach to modelling dynamic systems with stochastic actions.

In any event it is necessary that the formalisms provide an openness to the specification as it is a fundamental requirement for this work that a holistic unifying formalism be provided that can permit a shared understanding of knowledge state, autonomy of action and conflicting actions and the emergence of new entities into the system. For this reason this chapter commences with a brief overview of open systems.

4.1 Open Systems

It has been noted that the distributed systems evident in the real world display characteristics of openness and are exemplified by the following features: (Hewitt, 1990):

- There is communication between independently developed separate systems.
- Concurrent and asynchronous contact occurs through decentralised control based on interactions.
- Local inconsistency occurs amongst the system participants: The participants themselves hold consistent belief sets, but these may be at variance to the global or other participant perspective.

- System participants are bounded entities in terms of autonomy, knowledge and influence.

It is a fundamental feature of the application of distributed artificial intelligence that systems are large-scale open systems, which are subject to unpredictable outcomes and the emergence of new sources and items of knowledge, as described in Open Information System Semantics (OISS) (Hewitt, 1991). Thus the interactions of open systems can be characterised by features such as:

- Trials of Strength (Competition): When participants are working against each other through conflicting goals.
- Global Commitment (formalised earlier as joint intentions): When participants in the system carry out a joint course of action.
- Cooperation: When system participants have mutually dependent roles in the enactment of global commitments.
- Negotiation: A trial of strength conducted through the communication of shared beliefs.

Despite some opposition to OISS (Gasser, 1991), which suggests these notions are too vaguely defined and an encompassing computational theory is required, these concepts nevertheless represent a quintessential definition of an open system as it relates to this work in supplying the necessary attributes of an open system that any formalism must be able to accommodate. The formalism is further required to permit the runtime accomplishment of these features through adaptive middleware. Therefore the requisite properties of an open system, based on OISS, are:

- System participants are heterogeneous and are mostly capable of autonomous actions.
- The complete behaviour of the system participants or global outcome cannot be predicted entirely and in contrast to (Hewitt, 1991)
- Participants may enter and leave the system at any time.

This means that the meta-systems considered will feature trials of strength (competition), global commitments (joint intentions), cooperation and negotiation. Thus the formalisms and deliberation mechanisms must be able to accommodate these properties. This means that the states a system may enter throughout its runtime cannot be totally defined at

design time. The heterogeneity of component agents, and their interactions, makes it impossible to predict, at design time, both the sequence of state transitions and the states that will be entered. New previously undefined components or assemblies (federations/ teams/swarms) of component agents will arise through the runtime operations, resulting from the adaptation to competitive conditions, the provision of resources for cooperation, the outcomes of negotiation and the formation of joint intentions. Thus the formalism for the cognitive facilities an observer system must possess include the ability; to handle, analyse and recognise novelty and to be able to reason on the outcomes of actions: The formalism required for this top-down and bottom-up approach must necessarily allow the emergence of previously unknown states and permit hypothetical reasoning. There are two approaches that can be taken to provide such formalism: An algebraic approach or a logical predicate/propositional approach. The remainder of this chapter is devoted to assessing the formal systems available before defining the precise formal mechanism that will be used to define and implement the meta-systems with the previously stated required attributes.

4.2 State-Based Approaches

A state in a computing program or machine is a particular configuration of the data held at each step of a process. Typically the state can be represented by a set of variables with each state assigning different values to the variables. The variables are given initial values at the beginning of the process and change value according to system events. Thus it is possible to specify simple systems by a sequence of states or a set of transitions: So if in one state a particular event occurs then the system/machine moves into another specified state. This is a purely reactive system akin to the passive adaptation most often seen in current autonomic system implementations. A state stores the information regarding what has happened so far to the system (the changes in the value of the system's variables). A transition indicates a change of state and is described by a condition or action that would need to be fulfilled or occur, in that state, to enable the transition. An action is a description of the activity that is to be performed at a given moment. A very simple example is shown in Figure 4.1: In state 1 a light is on, when in this state an action occurs that is pressing the off switch then the state changes to state 2 where the light is off. Conversely when in state 2 and the light is off pressing the on switch causes the light to illume and the state changes back to state 1 LightOn.

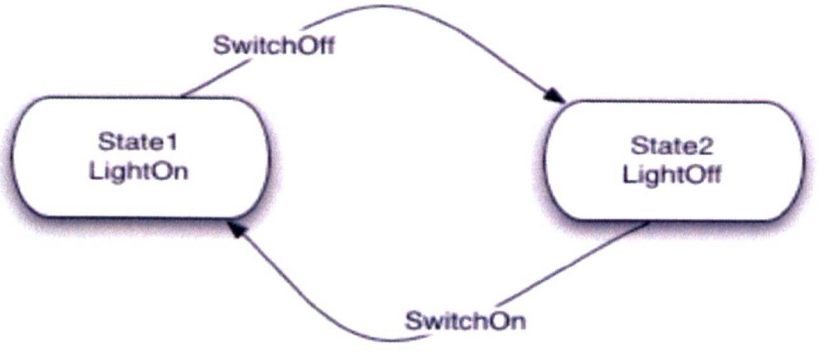

Figure 4.1 Simple State transitions

The simplest model of behaviour for a computer program is as an input/output function: A value is input at the beginning of the process and at some point there is a value as outcome or output. It is this idea that is the basis of Finite State Automata or Machines (FSA or FSM): A process is modelled as an automaton, which has a number of states and a number of transitions from state to state. The basic unit of the behaviour in such a system is an action, the execution of which is denoted by a transition. There are also initial states and a number of final states. A Behaviour of the process is a run from a permissible initial state to a final state. Two automata are considered equal if there is "language equivalence" between them determined by behaviour: Behaviour is characterised by the set of executions from initial state to final state. So language equivalence occurs between two automata if and only if they produce the same output for every identical input into each automaton.

It can be seen that the behaviour of system components can be represented in this way. However there is no provision to model the interactions between these components. Through an execution from initial state to final state a system may interact with another system. This is a requirement of modelling parallel or distributed systems with concurrency theory: The theory of interacting distributed components. The term Process Algebra has been applied to a class of formalisms considered to be an approach to concurrency, based on parallel composition as the basic operator.

4.2.1 Process Algebras

In Process Algebra, *process* refers to the behaviour of a system: A system is anything exhibiting behaviour, for instance the execution sequence of a software system, the actions of a machine or human actions. Behaviour may be defined as a permutation of all the actions or events the system can perform, the order in which the actions/events are

performed and any constraints such as timings or probabilities. Throughout this it is not possible to capture all behaviour but rather an abstracted version of behaviour, which captures the main function of the system. Behaviour is observed and the unit of observation is the action. These actions are further idealised to be discrete occurrences; performed at some time point with different actions happening at different times: A discrete event system.

Algebra in Process Algebra arises out of taking an axiomatic approach. In regular algebra elements represented by symbols are combined with operators and results may be calculated according to the rules or axioms of operations. Process algebra is a mathematical structure satisfying some axioms given for basic operations: A process is an element of the structure, which can be operated on or combined with other elements (processes) to perform calculations based on the axioms. In this way alternative composition (choice) can be reasoned on together with sequential and parallel composition as basic operations on processes. Thus it is possible to state some structural laws for operations in process algebras, similar to regular algebra: If the choice operation is denoted by +, sequential composition by * and parallel composition by ||. Then the basic laws, for processes X, Y and Z can be:

- X + Y = Y + X (Choice is commutative)
- X + (Y + Z) = (X + Y) + Z (Choice is associative)
- X + X = X (Choice is idempotent)
- (X + Y)*Z = X*Z + Y*Z (Choice is right distributive over sequential composition)
- (X*Y)*Z = X*(Y*Z) (Sequential composition is associative)
- X||Y = Y||X (Parallel composition is commutative)
- (X||Y)||Z = X||(Y||Z) (Parallel composition is associative)

So rules are established for manipulating the processes and defining structure. A null process may also be defined as a completely inactive process with no interactions. Depending on the domain the set of processes may form an abelian group under sequential composition, for instance. These laws however do not relate parallel composition to sequential composition or choice. The axioms only represent properties of the operators without any explicit representation of dynamics in a system: There is no

mention of action execution. To represent interleaving concurrency a theorem is required to express parallel composition in terms of the other two operators. The algebras that provide such a mechanism are termed process algebras of interleaved concurrency, in contrast to simple concurrency (Baeten, 1993). To adequately model the complex systems under consideration it is necessary to be able to represent both true concurrency, where two or more actions start and finish at the same time, and interleaved concurrency, where one action is due to occur after another; typically when the outcome is independent of the order of execution.

Prior to the formulation of process algebras in the 1970s the only theory that existed specifically to deal with concurrent processes was Petri Nets (Petri, 1962). Automata theory can model programs but it is not sufficiently powerful to deal with process interaction: The interactions of a process with other processes, between input and output, affect the outcome of the procedure making behaviour difficult to analyse. It is for this reason that the more powerful process algebras possess an expansion theorem that links the dynamic system properties of parallel composition and process communication with the more static system properties of sequential composition and choice. The usual way this is achieved is through the reduction semantic: linking parallel composition, communication and sequential composition. So, for instance, if $\bar{x}$ (a) represents the sending of message a along channel x and x(b) is the process that expects to receive data on channel x, where b is a place holder to be substituted by the arriving data, then for processes W and Z:

$$\bar{x}(a) * W \parallel x(b) * Z = W \parallel Z\left[{}^{a}/_{b}\right]$$

forms a general reduction semantic. This means that the process $\bar{x}$(a)*W sends a message a along channel x. The process x(b)*Z receives this message on channel x. When the message has been sent the process $\bar{x}$(a)*W then acts as process W whilst the process x(b)*Z acts as the process Z[a/b]: The process Z with a placeholder b replaced by the data received on channel x, namely a. It is the formulation of this particular property and the representation of communication that distinguishes the major process algebras. Additionally, to completely model computational systems, allowances must be made for non-terminating behaviour. Recursion and replication are methods of providing finite representations of infinite behaviour. Recursion may be adequately represented through sequential composition X*Y where Y = X*Y, for instance. Replication (!) may

likewise be represented by an infinitely countable number of parallel compositions of the same process: !X = X || !X.

The first process algebra with a complete theory was the Calculus of Communicating Systems (CSS) (Milner, 1980). This was quickly followed by similar formulations, Communicating Sequential Processes and the Algebra of Communicating Processes that emphasise slightly different aspects of the theory; it is from these that most modern process algebras may be derived.

The Communicating Sequential Processes (CSP) (Hoare, 1985) was originally conceived as a concurrent programming language (Hoare, 1978), without mathematically defined semantics or an ability to represent unbounded time delays in message passing (unbounded non-determinism). CSP, with influences from Milner's CSS, was further developed into the process algebra as presented in (Hoare, 1985). It places less importance on an equational structure, whilst the other early calculus, the Algebra of Communicating Processes (ACP) (Bergstra and Klop, 1984), prioritizes the algebraic structure and has a much more general abstraction of communication than the distinguishing combination of message passing and interleaving in CSP or the abstraction of communication by process hiding in CSS. Process hiding (in CSS and ACP) is denoted by τ: the passing from one state to another without any (observable) action occurring. All three share the static laws, as stated earlier. CSS denotes sequential composition by "*", with the same concept represented by ";" in CSP and "." in ACP. Differences become apparent in the choice operator, although the laws, as stated still hold for all representations: CSS and ACP denote choice as "?" and "+" respectively, whereas CSP distinguishes between deterministic choice (determined by the environment) denoted by "□" and non-deterministic choice (a simple indication that choice is available) denoted by "⊓". Parallel composition and communication are represented quite differently across the formalisms: CSP has different operators for communication and interleaving: "|[{a}]|" for communication across parallel composition where both the processes running in parallel must be able to perform action *a* before event *a* can occur: A synchronisation point; "|||" is used for interleaving to represent totally independent concurrent activity. After some initial variation CSS uses the same operator, "|", for communication and interleaving. Whereas, in CSP communication between processes occurs if both processes offer the action *a*, in CSS communication takes place if one process offers the action *a* and the other its complementary action $\bar{a}$ (interpreted as input

and output respectively). The communication between a and $\bar{a}$ results in a τ step: the communication provides synchronisation but the result is not visible. ACP offers parallel composition through the merge operator, "||": the parallel composition of two processes with interleaved actions. Communication is denoted by the binary operator "|".

Thus the main differences between the three major process algebras can be seen together with representational qualities they possess. Various formalisms extending and adding to the original process algebras have been proposed:

- Ambient Calculus (Cardelli and Gordon, 1998) is applied to concurrent computation where mobility, in terms of mobile devices and code location, is an issue. The basic primitive is an ambient: a bounded computational space that may be physical or virtual, a laptop computer or an address space, for instance, with a definite boundary. There are three basic operations on ambients, where computation is interpreted as the crossing of the ambient boundaries: to enter another ambient, exit an ambient or open an ambient, by dissolving its boundary. The reduction semantic defines the effects of these operations on ambients.
- π-Calculus (Milner et al, 1992) is also applied to concurrent computational scenarios in a mobile setting. It represents a continuation of the work on CSS and is mostly arranged around the same formalism. The difference is that the content of the message is a channel name, which may subsequently be used by the receiving process. Mobility is handled through network change based on the interactions of processes.
- Performance Evaluation Process Algebra (PEPA) (Gilmore and Hillston, 1994) extends CSS and CSP with performance evaluating primitives. This involves associating a random variable with each activity to represent the expected rate at which the task can be performed. The models formed can be used to assess quantitative properties of the system such as: latency, demand or throughput. It utilises the usual operator primitives of choice, sequential composition, co-operation (parallel composition) and hiding (transition through an unobservable event).
- Weighted Synchronous Calculus of Communicating Systems (WSCCS) (Tofts, 1993) is derived from a dialect of CSS: SCSS (Milner, 1983). It is based on an abelian group of action symbols with a set of weights. In the process algebras

discussed so far the issue of non-determinism has been handled by leaving the biases in making choices unspecified. In WSCCS the weights are used to add a probabilistic quantification for non-determinism. Such a probabilistic account provides a good formulation for specifying the bottom-up function emerging from many natural systems. It is in the adjustment of the probability that self-organising behaviour becomes apparent. An agent-based investigation of honey-bee colonies was used to demonstrate the value of this approach (Sumpter, 2000). Subsequently, more recently, this has led to applications for robotic swarm spacecraft (Rouff et al, 2005).

These process algebras are powerful yet primitive representation and reasoning formalisms. The elegance stems from the restricted number of operators: they are mathematical formalisms for describing and analysing the properties of concurrent computational systems yet do not include atomic elements such as; numbers, Booleans, data structures variables, functions or control elements such as if/then statements or do/while loops. This gives process algebras their apparent simplicity. However this simplicity comes at a price: from these basic elements all features of the system have to be constructed making representative formalisms more difficult to follow. In addition the language equivalence of finite state automata cannot be easily translated to interacting systems of automata. Bisimulation is the technique used to assess the equivalence of systems in process algebras. Informally this means that two systems are the same if their actions and transitions are indistinguishable. Thus the definition of bisimulation will be specific to the domain.

To utilise process algebras, in reality, requires the additional development of appropriate modelling and verification methodologies that have to be placed above the primitive constructs of the formalisms. This is typical of the requirement for modelling and proof infrastructure when dealing with higher order logics (Gordon and Melham, 1993). Additionally it can be noted that all states and transitions in a process algebra specification must be pre-defined before implementation. This represents a major drawback in providing the required adaptability and self-awareness necessary for systems self-governance. For these reasons it is proposed to investigate the use of predicate (first order) logics to achieve a similar simplicity of approach whilst gaining the advantages cognitive reasoning with built in fully formal verification and correctness mechanisms.

4.3 Propositional Formalisms

In seeking to gain the same simplicity of approach, with a minimum number of primitives, as process algebra and provide more flexible reasoning mechanisms it is necessary to address a basic problem in applied artificial intelligence (AI): What methodologies provide the best, most efficient mechanisms for converting the results of sensing actions into actions on the environment? Typically most AI programs use a simple set of assertions derived from data gathered by humans with no notion of recognition, spatial awareness, noisy sensors, etc (Brooks, 1991). This was concisely termed the Signal-Grounding problem in Chapter 3. It is the focus of this work that, based on the Knowledge Representation Hypothesis (Smith, 1982), as stated in Section 1.3, with the consequences, as listed in Section 1.4, the manipulation of logical sentences provides the most natural way to design "intelligent" processes.

A logical setting for AI, where knowledge is represented by logical sentences and intelligence by proving properties of these sentences, dates back to the 1950s (McCarthy, 1959). Green presented the classical approach to planning, using general purpose reasoning about actions in first order logic, to synthesise algorithms by theorem proving (Green, 1969): Given a description of the effects of actions on a domain Σ, an initial state S_0 and a goal Γ find a sequence of actions σ such that Σ logically entails Γ after σ is executed from S_0. The logical approach was further applied to robotics in the seventies as exemplified by the Shakey project (Nilsson, 1984) with the STRIPS planning system (Fikes and Nilsson, 1971), where states are represented by logical conjunctions of free ground literals. In the early nineties Levesque and Reiter took the early version of Situation Calculus (McCarthy, 1963), designed for logically specifying dynamical systems, and extended it to include time, concurrency, procedures, probability, etc., in a way that gave efficient implementations for cognitive robotics (Reiter, 1991). A contrasting logic approach to planning, for cognitive robotics, is presented by interleaved planning, sensing and execution (Shanahan, 1996) based on the Event Calculus (Kowalski and Sergot, 1986): the planning process is subject to frequent disruption while actions are performed and sensing data assimilated. In this given a goal Γ it is necessary to find a sequence of actions Δ_N such that:

$$\Sigma_B \wedge \Sigma_E \wedge \Delta_N \wedge \Delta_M \models \Gamma$$

Σ_B is a background theory comprised of the axioms for the domain.

Σ_E is a theory mapping the objects and actions in the domain to the sensor data.

Δ_M represents the many explanations (sequences of actions) to account for the sensing data. Thus signals within the system are grounded (Randles et al, 2006b) by interaction with the domain.

Δ_N may thus be taken to be the logical description of the actions occurring naturally in the environment or performed by a particular agent component.

Thus it is clear to see that there are a number of advantages to be gained with a logic-based approach to AI:

- If a component agent's design is logic-based then a rigorous mathematical account can be given regarding success or failure in meeting its goals.
- The component agent's knowledge and goals are specified in a universal declarative language, making the component easy to modify, adapt or maintain.
- It is a reasonably easy task to incorporate high-level cognitive facilities in a logic-based component: including the ability to reason on the actions of other components, its own knowledge or to plan based on sensed data.

Thus it would seem that techniques previously restricted to applications in cognitive robotics might provide rich formalisms to endow modern autonomic type systems, for self-governance, with the necessary cognitive properties to reason on system features both in terms of component (norm-based) autonomy and the observable global outcome of component interaction. In this way self-governance and autonomic functionality can be applied to much larger and more complex systems than is currently the case. This gives rise to a number of methodological options: Firstly, at one extreme, the basic unit of representation is the logical sentence and the unit of computation is a step in a proof. This gives a very short path from specification to implementation that simply involves the application of a general-purpose theorem prover over the axioms of the domain (previously termed Σ_B) and the sensed data (previously termed Σ_E). The same logical sentences and the same theorem prover can provide necessary conditions to explain the sensing data (abduction) for planning as well as deducing and inferring other results based on Σ_B and Σ_E. Unfortunately it is not likely, in the near future, that any such powerful theorem prover will become available. Secondly, at the other extreme, algorithms for planning, sensing and action can be handcrafted and proved correct with

respect to the logically specified account. This approach, however, gives no systematic process by which the implementation may be derived from the specification and makes consistent maintenance and adaptation difficult to achieve. Finally the best approach, between these two extremes, is centred on a form of logic programming: The logical specification is preserved into the programming language, making the specification computationally realisable. This is achieved by taking the clausal fragments, described in a suitable formalism and rendering them into a language that makes use of the logic. System properties, behaviour, correctness and all logical consequences of the specification can be easily derived given a mainly first order logical formalism based on Horn clauses.

4.3.1 Logical Calculi: Events and Situations

Logical theories of action and change most usually involve reasoning about the truth of propositions and the occurrence of events in terms of causation and timing; either relative to some ordering or absolute for scheduled events. The main logical calculi, considered to be appropriate for reasoning about dynamic systems are Event Calculus (Kowalski and Sergot, 1986) and Situation Calculus (McCarthy, 1963). The two formalisms were originally conceived with very different ontologies to satisfy a requirement to specify a domain from a particular perspective. The Event Calculus, as currently formulated (Miller and Shanahan, 2002), may be utilised to provide a reasoning tool to analyse the "commonsense" description of domains, including cognitive states (Mueller, 2006). In Event Calculus the central theme is the notion of action occurrences, or events, happening at specific time points. These events provide the start and end points for time intervals during which certain fluents hold. These time points are considered to occur along a single real-valued time line: Fluents can hold or not at a certain time point. The formula *time(T)* denotes that *T* is a time point. The formula *holds(A,T)* represents the fact that A holds at time T. An event is the occurrence of an action at a certain time point, denoted by: *happens(E,T);* event E happens at time T and *act(E,a)* denotes that event E consists of an occurrence of action *a*. The final primitives in Event Calculus are *initiates(E,A)* and *terminates(E,A)* meaning that the event E initiates or terminates, respectively, the fluent A. Thus a frame axiom may be stated in Event Calculus as:

$$holds(A,T) \Leftrightarrow happens(E_1,T_1) \wedge (T_1<T) \wedge initiates(E_1,A) \wedge \neg(happens(E_2,T_2) \wedge (T_1<T_2) \wedge (T_2<T) \wedge terminates(E_2,A))$$

There is also a constraint on timing: The order on time points must be a linear order. This means that the following must hold for all T_1, T_2 and T_3:

- $\neg((T_1<T_2)\wedge(T_2<T_1))$
- $((T_1<T_2)\wedge(T_2<T_3)) \Rightarrow (T_1<T_3)$
- $(time(T_1)\wedge time(T_2)) \Rightarrow ((T_1<T_2)\vee(T_2<T_1)\vee(T_1=T_2))$
- $(T_1<T_2) \Rightarrow (time(T_1)\wedge time(T_2))$ – meaning that operator "<" can only relate time points.

The linear ordering of time in the Event Calculus is necessary because partial orderings makes it possible to build system models that do not conform to time constraints (Denecker and De Schreye, 1995). For instance, without linear ordering, related events cannot be scheduled to occur in the required ordering: A terminating event may occur before an initiating event for a particular fluent. This problem with the Event Calculus also precludes any reasoning involving branching time structures, making reasoning in possible worlds models impossible. Likewise conterfactual reasoning cannot be supported; Statements of the form, 'if action A had happened then proposition B would have held' can only be represented where it is possible to conceive of several possible evolutions of the world (Belleghem et al, 1997). Extensions to the Event Calculus have been proposed to accommodate hypothetical reasoning: The linear ordering of time $[(T_1<T_2)\vee(T_2<T_1)\vee(T_1=T_2))]$ can be replaced by a branching time axiom defining a situation, initiated at time point T_1, as a set of time points, T_2, such that no events occur between T_1 and T_2 (Belleghem et al, 1997). This formalism, however, does not allow for concurrent events. Another proposal (Lévy and Quantz, 1998) endows each Event Calculus predicate with a situation argument with situations being related by equality through certain time intervals. A Branching Discrete Event Calculus (BDEC) (Mueller, 2007) is proposed for discrete event systems, shown to be equivalent to continuous event systems with integer time, using a restricted Situation Calculus. This, however, can lead to two identical sequences of events resulting in two distinct (non-equivalent) situations.

While seeking formalisms to adequately represent the cognitive functions, necessary to reason on large-scale systems self-governance, it is desirable, if not necessary, to be able to reason on the various outcomes of actions. It is clear that representing such reasoning in the Event Calculus is not a straightforward task. The most often used methods to endow Event Calculus with hypothetical reasoning procedures always involve imbuing

Event Calculus with a restricted Situation Calculus type functionality. Thus it may be more practical to actually use a pure form of the Situation Calculus from the first instance to gain the reasoning capabilities necessary for systems self-governance.

In Situation Calculus the basic concepts are actions and situations. It provides a quite natural way to represent commonsense type formulations. A situation provides a snapshot of the world that is changed by an action occurrence: Actions are the cause of situation transitions. In line with the recent applications of Situation Calculus to cognitive robotics, where a fully formal description of the Situation Calculus has been provided (Levesque et al, 1998), this work will consider a situation to be a sequence of actions or action history, forming the major objects in the calculus. Each of these situations will have a set of fluent values dictated by the initial situation, termed S_0 and the action history. There is a primitive binary operation *do*; *do(a,s)* denoting the successor situation to *s* resulting from performing the action *a*. Actions are generally denoted by functions and situations (action histories) are first order terms. In seeking to represent a domain actions must have preconditions; necessary conditions that must hold before the action can be performed. The predicate *poss* is used with *poss(a,s)* meaning that it is possible to perform the action *a* in a world resulting from the execution of the sequence of actions *s*. In order to address the frame problem, effect and frame axioms are combined into one Successor State Axiom (Reiter, 1991): A successor state axiom for a fluent is TRUE in the next situation if and only if an action occurred to make it TRUE or it is TRUE in the current situation and no action occurred to make it FALSE, with the precondition axiom *poss*(a, s) meaning it is possible to perform action a in the situation s. These form the simple, yet highly expressive, primitives of the Situation Calculus. There have been many attempts, as outlined above, to give the Event Calculus the expressive power of the Situation Calculus. The example of counterfactual reasoning shows that there are problems handled in the Situation Calculus that cannot be handled in the Event Calculus. Most attempts to reconcile the two, however, include an equivalence result, showing the two formalisms to be essentially the same (Kowalski and Sadri, 1997). This contradiction is resolved because all the equivalences are between the Event Calculus and a restricted version of the Situation Calculus. For instance in (Kowalski and Sadri, 1997) frame axioms, in the Situation Calculus are augmented with *happens(A,S)* primitives: This makes frame axioms applicable only to actual asserted sequences of actions and additionally ensures that only one sequence of actions can exist, just like the Event Calculus. Likewise BDEC

(Mueller, 2007) restricts reasoning to actual asserted action histories with the requirement that no actions can occur between adjacent situations.

Rather than seeking to provide Event Calculus with the ubiquity of representation exhibited by the Situation Calculus, it would seem more sensible to use the Situation Calculus to provide the primary representation and reasoning system. The conciseness of the language with the greater expressiveness, allowing hypothetical reasoning, gives a very powerful initial formalism for defining and implementing system self-governance meta-systems. The next section summarises and defines the main results in Situation Calculus augmented, where necessary, for application in this work.

4.4 The Situation Calculus

Henceforth this work will take a logical approach to modelling the dynamic systems, necessary to provide cognitive functions in adaptive middleware meta-systems for self-governance in large–scale complex systems, based on a first order logic language, Situation Calculus, first proposed by John McCarthy in 1963 (McCarthy, 1963). Since that time much has happened to and through the Situation Calculus. The problem of not only specifying what changes occur in a domain but also the many instances of no change in a system leads to the unavoidable consequence of a large number of frame axioms (McCarthy and Hayes, 1968), termed the representational frame problem. Reiter's solution (Reiter, 1991) provided a very efficient and compact representational mechanism to mostly resolve the frame problem. This provides two main benefits: Modularity, because as new actions and fluents are added to, or become apparent in, the domain, only new effect axioms need to be included as the frame axioms are automatically supplied or updated from these; Accuracy, because the frame axioms are given as part of the effect specification there is no possibility of accidental omission.

In addition methods to address further problems, previously identified as presenting, difficulties in logically specifying a domain may be circumvented. The position taken in this work, to these problems, is as follows:

- The Qualification Problem, concerning the reliability of actions in the real world. Given an action it is difficult to define circumstances under which the action is guaranteed to succeed. That is there may be an indefinite number of minor qualifications as action preconditions. The usual way to accommodate such a problem is by choosing to ignore all the minor qualifications; concentrating on the

necessary and sufficient conditions that define when an action may be performed (Reiter, 2001).

- The Ramification Problem arises because of the implicit consequences of actions. It involves the additional consequences of actions because of state constraints (Finger, 1986), also termed integrity constraints in database scenarios. For instance the movement of a process also entails the movement of any processes contained within it. The constraints/relationships may be explicitly represented and the effect of actions taken into account accordingly (Schubert, 1990).

This more commonsense approach to reasoning and more modular representational style has led to applications of the Situation Calculus to many real problems. The original appeal was in applications to cognitive robotics (Lesperance et al, 1994) with agent programming (Lesperance et al, 1997) and database updates (Lin and Reiter, 1994) naturally following. More recently works have appeared on modelling ubiquitous information services (Dong et al, 2004), representing agents in e-business (Albrecht et al, 2003), solving logistics problems with Markov Decision Problems (Sanner and Boutilier, 2006) and reasoning with incomplete knowledge (Vassos and Levesque, 2007). There are also works progressing on the meta-theory of Situation Calculus improving representational and reasoning techniques. For instance the ramification problem and complex temporal phenomena are addressed in the Inductive Situation Calculus (Denecker and Ternovska, 2007).

The main difference, in using Situation Calculus, from alternative approaches is the concentration on cognition as the driver for component-agent behaviour and observer system function. This is needed because the state and evolution of large-scale systems cannot be foreseen at design time; thus cognitive functions are required at runtime, by the meta-system for self-governance and participant component-agents, to reason on the perceived state of the system and take appropriate actions. Additionally component-agent behaviour can be specified to engineer emergence or condition behaviour: modelling component-agent beliefs and their consequences. Beliefs encompass statements about what is true in the component-agent's environment, what actions it and its fellow components, including the environment as a whole, can perform, what effects these actions have on the domain, the conditions under which actions may be performed, the resources available for cooperative task accomplishment and the outcomes of sensing procedures. These beliefs condition behaviour and it is the purpose of this work to give a

theoretical and computational account of the process of deliberation that leads to action based on the Knowledge Representation Hypothesis (Smith, 1982) using logical sentences as the fundamental mathematical representation for large-scale system control and analysis; thus answering the question posed at the start of Section 4.3: What methodologies provide the best, most efficient mechanisms for converting the results of sensing actions into actions on the environment? The following subsections summarise the features of the Situation Calculus that will be required to progress with this work. Although a fairly technical treatment of the calculus will be presented it is not the intention to give a fully formal account of the Situation Calculus in this work; the descriptions given are based on or derived, where necessary, from (Levesque et al, 1998).

4.4.1 The Situation Calculus Language: Foundational Axioms

The Situation Calculus is a language of three disjoint sorts: Actions, situations and objects (anything that isn't an action or situation, depending on the domain). Generally *s* and *a*, with suitable subscripts, will be used for situations and actions respectively. Additionally the language consists of:

- Two function symbols of the sort situation;
  - The constant symbol S_0 denoting the initial situation
  - A binary function *do:action×situation→situation.* Situations are sequences of actions; *do(a,s)* is the result of adding the action *a* to the end of the sequence *s*.
- A binary predicate symbol *poss:action×situation. poss(a,s)* meaning it is possible to perform action *a* in situation *s*.
- Predicate symbols:
  - Of the sort $(action \cup object)^n$ for each $n \geq 0$ denoting situation independent relations such as $moveAction(move(service,l_1,l_2))$
  - Of the sort $(action \cup object)^n \times situation$ *for each* $n \geq 0$ denoting relational fluents: situation dependent relations such as *heavyLoad(service,s).*
- Function symbols:

- Of the sort $(action \cup object)^n \rightarrow object$ for each $n \geq 0$ denoting situation independent functions such as *location(move(service))*
- Of the sort $(action \cup object)^n \times situation \rightarrow object \cup action$ for each $n \geq 0$ denoting functional fluents: Situation dependent functions such as *location(service,s)*.

- Action functions to denote actions of the form $(action \cup object)^n \rightarrow action$ for each $n \geq 0$ such as move(l_1,l_2).
- An ordering relation $\subset$: $situation \times situation$ denoting a subsequence of actions: $s_1 \subset s_2$ means s_1 is a proper subsequence of s_2. Alternatively the situation s_1 occurs before s_2.

The functional and relational fluents take one argument of the sort situation; by convention this is usually the final argument. There are four immediate, domain independent, axioms of the Situation Calculus, which detail the fundamental properties of situations in any domain specific representation of actions and fluents:

1. $do(a_1,s_1) = do(a_2,s_2) \Leftrightarrow (a_1 = a_2) \wedge (s_1 = s_2)$
2. $(\forall P)[P(S_0) \wedge \forall (a,s)(P(s) \Rightarrow P(do(a,s)))] \Rightarrow \forall s\, P(s))$
3. $(\neg \exists s)(s \subset S_0)$
4. $s_1 \subset do(a,s_2) \equiv (s_1 \subset s_2) \vee (s_1 = s_2)$

Some logical consequences of the foundational axioms follow immediately, for instance:

- $S_0 \neq do(a,s)$ for any s
- $do(a,s) \neq s$ for any a or s
- $(s = S_0) \vee (\exists (a,s_1)\ s=do(a,s_1))$. i.e. $S_0 \subset s$ for all s

This version of the Situation Calculus differs from McCarthy's (McCarthy, 1963) original formulation in which situations were identified with states. The state-based approach was evident in early works (McCarty and Hayes, 1968) where it was stated: "A situation, s, is the complete state of the universe at an instant of time." A different approach is to combine states and situations with an action theory as occurs in the Fluent Calculus (Thielscher, 1998) with a translation available at implementation level (Schiffel and Thielscher, 2006).

4.4.2 Composite Actions and Procedures

Sometimes there will be a requirement to move between situations that are separated by a sequence of actions rather than a single event or action. This will be denoted by $DO(\alpha,s_i,s_j)$, which, informally means that it is possible to reach situation s_j from situation s_i by executing the sequence of actions, $\alpha = [a_1,a_2,.....,a_n]$, where each a_i is a single action. To relate a Situation Calculus representation to more familiar programming constructs *DO* may be defined inductively by its action on the sequence comprising of its initial argument.

- If α is just a primitive action *a* then $DO(a,s,s^*) \equiv poss(a\langle s\rangle,s) \wedge s^* = do(a\langle s\rangle,s)$. It is necessary to put situational values into the actions, the meaning of $a\langle s\rangle$, as the action may rely on the value of a fluent in that situation. For example, the action expression *moveTo(location(service1))*, in a scenario for mobile services, where *service1* is a particular service is a functional fluent that is situation dependent as service1 may move. Thus the action needs to state *moveTo(location(service1,s))*.
- If α is a test condition denoted by "?" then $DO(a?,s,s^*) \equiv a\langle s\rangle \wedge s=s^*$
- If α is the sequence of actions split into $[\alpha_1, \alpha_2]$ denoted by ";" to separate the sequence terms (which may themselves be sequences of actions) then $DO(\alpha_1; \alpha_2,s,s^*) \equiv (\exists s')[DO(\alpha_1,s,s') \wedge DO(\alpha_2,s',s^*)]$
- If α is a non-deterministic choice of actions from $\{\alpha_1\ \alpha_2\}$, denoted by "|", then $DO(\alpha_1|\alpha_2,s,s^*) \equiv DO(\alpha_1,s,s^*) \vee DO(\alpha_2,s,s^*)$
- If α is an iteration executed 0 or more times, denoted by "*" and $T \subseteq situations \times situations$ is the set of ordered pairs of accessible action histories, where the second is accessible from the first term, then $DO(\alpha^*,s,s^*) \equiv$ $(\forall T)[(\forall s_1)\ (s_1,s_1) \in T \wedge (\forall s_1,s_2,s_3)[DO(\alpha,s_1,s_2) \wedge (s_2,s_3) \in T \Leftrightarrow (s_1,s_3) \in T]]$ $\Leftrightarrow (s,s^*) \in T$

This states that doing the action(s) α zero or more times takes the situation from s to s^* if and only if (s,s^*) is in every set T where (s_1,s_1) is in T for all s_1 and whenever doing α in situation s_1 brings about situation s_2 and (s_2,s_3) is in T then (s_1,s_3) is in T. Using these constructs it is possible to start to relate Situation Calculus to a programming language (Lesperance et al, 1997). For instance:

$$if\ a\ then\ \alpha_1\ else\ \alpha_2 \equiv a?\ ;\ \alpha_1 \mid \neg a?\ ;\ \alpha_2$$

$$while\ a\ do\ \alpha \equiv (a?\ ;\ \alpha)*\ ;\ \neg a?$$

4.4.3 Time and Concurrency

In order to represent actions that occur in time, for a certain duration or at the same time it is necessary to consider the ways in which time may be represented within the axioms of the Situation Calculus, as stated up to this point. The formulisation described so far only conceives of actions occurring sequentially and without timing constraints. As happened in previous discussions regarding process algebras there are different scenarios under which concurrent actions can occur: Actions may occur together and have the same duration; the duration of one may completely envelope the duration of the other or their durations may just overlap. The representational device used within the Situation Calculus to address these problems is to consider instantaneous actions initiating and terminating action durations with a relational fluent representing the extent of the action (Reiter, 1996). For instance instead of the monolithic action to move a process, A, from location l_1 to location l_2: *move*(A,l_1,l_2) the instantaneous actions *startMove* and *endMove* may be used and the procedure of moving represented by the relational fluent *moving*(A,l_1,l_2,s)*:* The *startMove* action causing the *moving* fluent to become true with the *endMove* action making it false. Similarly the *communicate* action can be represented by the pair of instantaneous actions *startCommunicate* and *endCommunicate* with the relational fluent *communicating(s).* It is then quite simple to represent these actions and fluents in the Situation Calculus, as defined:

$$poss(startMove(A,l_1,l_2),s) \Leftrightarrow \neg\exists(l_3,l_4)moving(A,\ l_3,l_4,s) \wedge location(A,s)=l_1$$

$$poss(endMove(A,l_1,l_2),s) \Leftrightarrow moving(A,\ l_1,l_2,s)$$

$$moving(A,\ l_1,l_2,do(a,s)) \Leftrightarrow a=startMove(A,l_1,l_2) \vee [moving(A,\ l_1,l_2,s) \wedge a \neq endMove(A,l_1,l_2)]$$

$$location(A,do(a,s))=l_2 \Leftrightarrow \exists\ l_1\ a=endMove(A,l_1,l_2) \vee [location(A,s)=l_2 \wedge \neg(\exists\ l,l')\ \ a \neq endMove(A,l,l')]$$

With this representation complex concurrency can be handled. For example for a particular process:

$$\{startMove(l_1,l_2),\ startBroadcast\},\{endBroadcast,\ startReplication(l_3)\},\ \{endMove(l_1,l_2)\}$$

is the sequence of actions commencing with simultaneously starting to move from l_1 to l_2 and broadcast, followed by simultaneously ending the broadcast and starting to replicate at location l_3, followed by ending the move at l_2 whilst the replication is still proceeding.

This gives a particularly neat representation for interleaved concurrency: Two actions are interleaved if one is the next action to occur after the other. Thus an interleaved concurrent representation can be given for moving and broadcasting, for

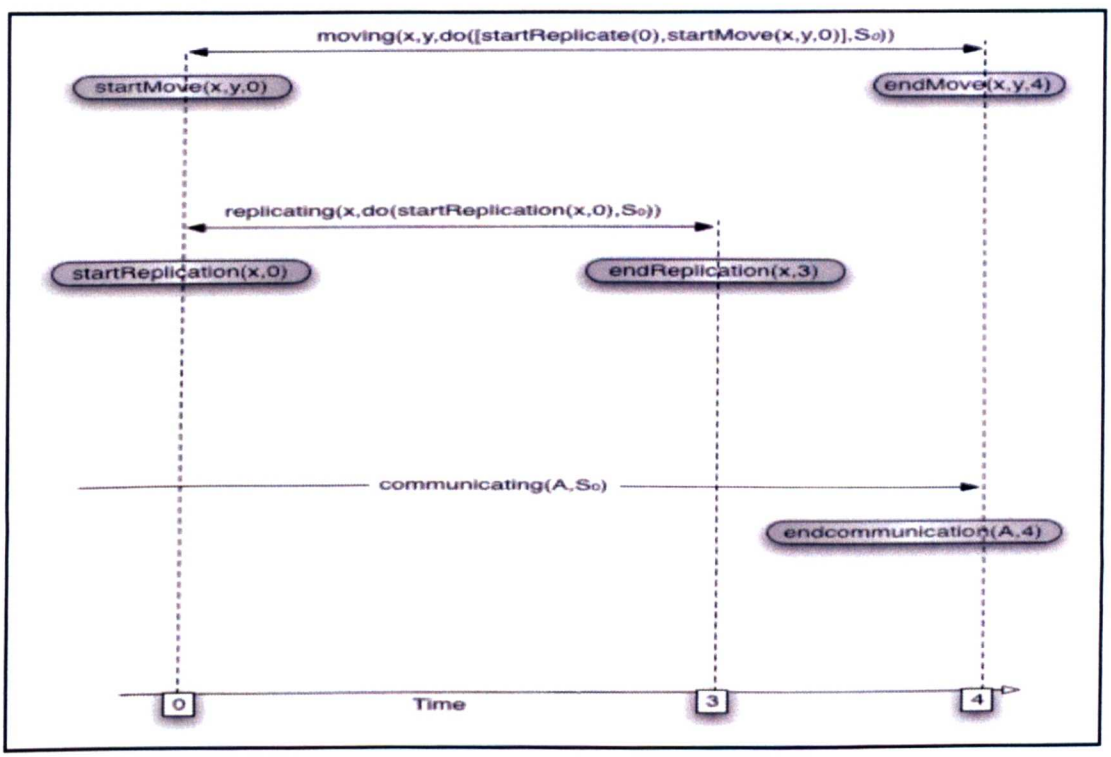

Figure 4.2 Interleaved Time Processes in Situation Calculus

instance: do([*startMove*(l_1,l_2),*startBroadcast*,*endBroadcast*, *endMove*(l_1,l_2)],S_0) where broadcasting is initiated after a move is started and terminated before the end of the move. Alternatively it might be the case that:

do([*startBroadcast*, *startMove*(l_1,l_2), *endBroadcast*, *endMove*(l_1,l_2)],S_0)

Thus any overlapping occurrences of moving and broadcasting, except for exact co-occurrences of the initiating or terminating actions, may be realised in the formalism. This is achieved without having to extend the formalism in any way.

To incorporate time into the Situation Calculus does involve extending the foundational axioms with one new axiom and introducing two new functions. The representational device for denoting time, in the Situation Calculus, is simply to add a temporal argument to actions. Thus startBroadcast(t) is the instantaneous action of the broadcast starting at

time t. The first new function is devoted to extracting this time argument from the representation: It is a function *time:actions*$\rightarrow\Re$ *.time(a)* where $\Re$ is the set of real numbers. So, for example *time(startBroadcast(t))=t.* The second new function is *start:situation*$\rightarrow\Re$*. start(s),* which denotes the start time of situation s. This gives rise to the new foundational axiom, which may be succinctly stated as; *start(do(a,s))=time(a).* This facility to express time allows for the representation of any complex interleaving processes. In Figure 4.2 a time line is illustrated where there are three overlapping processes: The movement of a process (say) from location *x* to location *y* (*moving(x,y,s)*); the replication of the process at location *x* (*replicating(x,s)*) and the communication with another process A (*communicating(A,s)*). The *moving* and *replicating* procedures begin together at t=0 with the *replicating* finishing at t=3 and the *moving* at t=4. The *communicating* procedure is already occurring in the initial situation (t=0) and finishes at the same time as the *moving* procedure (t=4).

Finally, in dealing with modelling temporal aspects of dynamic systems, it is possible to handle true concurrency; when one or more actions happen at the same time. This is achieved by simply considering concurrent actions as sets of simple actions: The notation $a\in A$ denotes that the simple action *a* is one of the actions of the concurrent action *A*. The *do* function symbol can be extended to accept concurrent actions as arguments: $do(\{startMove(x,y),openFile(f)\}, S_0)$, for example. The foundational axioms, as previously stated, are then composed exactly as for simple actions; a concurrent action set replaces the simple action in the axioms. The action precondition axioms then become: endMove

$$poss(a,s)\Rightarrow poss(\{a\},s) \text{ and}$$

$$poss(A,s)\Rightarrow(\exists a)a\in A\wedge(\forall a)[a\in A\Rightarrow poss(a,s)]$$

The successor state axioms can also be adapted to take account of concurrent actions, for instance:

$$moving(x,y,do(A,s))\Leftrightarrow startMove(x,y)\in A\vee moving(x,y,s)\wedge endMove(x,y)\notin A$$

There are certain problems with true concurrency: Precondition interactions or conflicts may be present. This does not occur with interleaved concurrency: This work will mostly make appeal to the account of interleaved concurrency when modelling system actions.

4.4.4 Sensing and Knowledge

The representation of knowledge and beliefs in the Situation Calculus is achieved by seeing the world states as action histories or situations with the concept of accessible situations (Moore, 1985). So if s_1 and s_2 are situations then $(s_1, s_2) \in K_i$ means that in situation s_2 agent i considers s_1 a possible situation with K_i an accessibility relation for agent i.

That is all fluents known to hold in situation s_2 also hold in s_1. So an accessibility fluent may be specified: $K_i(s_1,s_2)$ meaning in situation s_2 agent i thinks s_1 could be the actual situation.

So knowledge for agent i ($knows_i$) can be formulated in a situation as:

$knows_i(\phi, s) \equiv \forall s1(K_i(s_1, s) \rightarrow \phi(s_1))$ [alternatively $\forall s_1 (\neg K_i(s_1, s) \vee \phi(s_1))$]

This gives rise to a fluent to represent knowledge dynamics in the Situation Calculus that still satisfies the constraints of the solution to the frame problem (Reiter, 1991).

However to make any axiom complete it is necessary to establish whether a sensing action has taken place (Scherl and Levesque, 1993). That is if the action that occurred, to change the situation to its successor, was the perception of the value of a fluent. So the change was a change in the epistemic state of the agent. Thus it is necessary to distinguish sensing actions by writing $SR(sense_\phi,s)$ to denote that the action produced a result for ϕ.

$SR(sense_\phi, s) = r$ = value of ϕ in s

Thus a successor state axiom for K can be stated:

$K(s_2, do(a, s)) \Leftrightarrow \exists s_1(s_2=do(a, s_1) \wedge K(s_1, s) \wedge poss(a, s_1) \wedge SR(a, s)= SR(a, s_1))$

4.4.5 Probability: Stochastic Situation Calculus

In order to account for the uncertainty present in the domain it is necessary to consider a stochastic formulation of the Situation Calculus. To achieve this, as part of a first or second order logic representation, the approach is to decompose stochastic actions into deterministic options (Boutilier et al, 2001). So there are two classes of actions:

- Stochastic Actions that are under the control of a system component but have an uncertain outcome.

- Environmental Choices that form the choice of deterministic actions for any component chosen stochastic action.

There will be a finite number of choices for each action and each will have an associated probability, *prob(a,b,s)* meaning the probability that a will be the outcome of stochastic action *b* with *a* a possible action in s. It can be defined:

$$prob(a,b,s)=p \equiv choice(b, a) \wedge poss(a,s) \wedge p=prob_0(a,b,s) \vee [\neg choice(b,a) \vee \neg poss(a,s)] \wedge p=0.$$

Here *choice(b,a)* is the decomposition of the stochastic action *b* into its deterministic primitives and $prob_0(a,b,s)$ is the associated probability for each deterministic action *a*. This gives a modular construction to provide axioms when uncertain outcomes are observed in the domain. For example if the stochastic action is the flipping of a coin then: $choice(flipCoin,a) \equiv a=showHead \vee a=showTail$ with

$prob_0(showHead,flipCoin,s) = 0.5$ and $prob_0(showTail,flipCoin,s) = 0.5$

In reality most stochastic actions revolve around the success or failure of action occurrences. For example if there are two actions of adding a process to a system and making a link from process X to process Y, denoted by *addProcess* and *addLink(X,Y)*. The *addLink* action is a stochastic action, in that the action may succeed or fail so can be decomposed into two deterministic actions: *s_addLink(X, Y)* and *f_addLink(X,Y)* (meaning the action succeeds or fails respectively). The action *addLink* is assumed to be under some cognitive systems autonomous control. If the system elects to perform the action then non-determinism arises and exactly one of either *s_addLink(X, Y)* or *f_addLink(X,Y)* is enacted with associated probabilities. Thus, as above:

$$choice(addLink(X,Y),a) \equiv a=s\_addLink(X,Y) \vee f\_addLink(X, Y)$$

with

$$prob_0(s\_addLink(X,Y),addLink(X,Y),s) = p$$

$$prob_0(f\_addLink(X,Y),addLink(X,Y),s) = 1-p$$

where p is a probability distribution for the action occurrence.

The associated successor state axioms for the domain and the action precondition axioms can then be stated in the usual manner. To provide a decision theoretic position into the account, to in effect assess the importance of actions, a real valued reward function,

reward(a,s), may be introduced. This denotes the reward a component may gain by performing action *a* in situation *s*. For instance:

$$reward(addProcess,s) = 40$$

$$reward(s\_addLink(X,Y),s)=r \equiv [(server(X) \vee server(Y)) \wedge r=80] \vee$$
$$[(process(X) \wedge process(Y)) \wedge \neg(server(X) \vee server(Y)) \wedge r=30]$$

$$reward(f\_addLink(X,Y),s)=r \equiv [(server(X) \vee server(Y)) \wedge r=-60] \vee$$
$$[(process(X) \wedge process(Y)) \wedge \neg(server(X) \vee server(Y)) \wedge r=-10]$$

There may also be costs involved with performing actions. For instance it is costly to move a process from its present location *loc(s)* to a new location *l*:

$cost(move(l),s) = 0.2*|(loc(s),l)|$ where $|(x,y)|$ is some domain measure of a distance between x and y. It is possible to eliminate the cost function by incorporating it into the reward function. The domain usually determines the approach taken: If resources are consumed in the performance of actions it is usually more intuitive to retain the cost function. In this way Markov Decision Processes (MDP) may be specified in the Situation Calculus: The representation, in Situation Calculus of a dynamic probabilistic domain. In general an MDP consists of a state (situation) space, a set of actions, a transition function over the space and a reward function that maps a state (situation) and action to a real value. The goal is to find a policy that maximises the value function. In the simplest case, for a Situation Calculus account of an MDP, a cost function, *cost(a,s)*, and a reward function, *reward(a,s)*, are defined for each deterministic outcome *a* of a stochastic action and the value function becomes a linear combination of these:

$$value(do(a,s)) = value(s) + reward(a,s) - cost(a,s)$$

The application and solution of such problems is very applicable to engineering the self organising processes observed to occur in natural systems, as outlined in Chapter 2, such as: Foraging, herding, nest building moulding, web weaving and brood sorting. It is the adjustment of probabilities that determines whether behaviour emerges or not in these scenarios. This will be further investigated later in this thesis.

4.5 Summary

This chapter has provided the justification for the choice of Situation Calculus as the unifying formalism for providing the modelling and reasoning necessary to handle the

top-down normative approach to software engineering, the bottom-up engineering for emergence and the cognitive facilities for deliberating on systems' evolution and the emergence of novel function. The modelling formalisms under consideration need to exhibit properties of openness to handle the uncertainty within their operating environments. State based approaches rely on process algebra, mathematical formalisms for describing systems of interacting Finite State Machines (FSMs). It is a flat modelling formalism: There are two sorts, states and transitions, together with axioms for composing the FSMs, usually with no hierarchical structure; all the states appear in the same layer and transitions may connect any pair of states leading to a modelling formalism that is difficult to scale and maintain. In addition each realisation of a model is a case specific solution: A new application usually requires a completely new design and subsequent changes in the model cause difficulty with maintenance issues because of the tight coupling between states and transitions. However in relation to the requirements, in this work; to design, specify and analyse deliberative meta-systems for the automatic run-time maintenance, tuning and security of large-scale, complex, distributed computing systems, the major failing of a process algebra approach is the necessary enumeration of all possible states in advance. Such enumeration is impossible in this work as the evolution of large-scale complex systems is, in general, not predictable (Bullock and Cliff, 2004). Even if enumeration were possible changes in state compositions would result in an exponential increase in the number of system states. Furthermore any practical utilisation of a process algebra, including the facility for deliberation on the system by itself, requires the additional provision of modelling and verification methods to sit above the primitive constructs of the formalism (Cerone and Milne, 2005). Thus the need for a formalism giving a "propositional account" (Smith, 1982) is established. This naturally leads to the adoption of mathematical logic: Instead of enumerating states and their transition functions, sentences describing what is true in the system and its environment and the causal laws in effect for that environment are favoured. This means that: System behaviour is determined by the logical consequences of the systems description, through entailment; a non-procedural specification is provided where system properties may be verified and logical deduction used to establish correctness properties and a system specification is obtained that is executable, giving a formal system simulator. Throughout this work the formalism used is required not only to model the systems under consideration but also to provide the deliberative functions of a cognitive

system. This promotes the use of a single formalism to model the system, for correct function through normative positions or emergence engineering and permits the reasoning over system state, for analysing and grounding emergent self-organisation. It is for this reason that the Situation Calculus was chosen to provide such formalism: The other predominant suitable propositional representation system, Event Calculus, necessitates reasoning with the linear concept of a single time line, thus making hypothetical reasoning unfeasible. Attempts to incorporate hypothetical or counterfactual reasoning into Event Calculus tend to transform it into an equivalent Situation Calculus with additional primitive constructs, giving a less elegant representational formalism. Table 4.1 summarises the features required of a formalism for this work in comparison to the calculi previously reviewed in this chapter.

Table 4.1: Comparison of Formal Representation System Features

	NO STATE ENUMERATION	CONSTITUENT VERIFICATION/ CORRECTNESS	STOCHASTIC SYSTEM MODELLING	QOS/QOP	HYPOTHETICAL REASONING
CSS	✖	✖	✖	✖	✖
CSP	✖	✖	✖	✖	✖
ACP	✖	✖	✖	✖	✖
AMBIENT CALCULUS	✖	✖	✔	✖	✖
Π-CALCULUS	✖	✖	✔	✖	✖
PEPA	✖	✖	✔	✔	✖
WSCCS	✖	✖	✔	✖	✖
EVENT CALCULUS	✔	✔	✖	✔	✖
SITUATION CALCULUS	✔	✔	✔	✔	✔

The Situation Calculus is widely regarded as a formal modelling system for dynamic worlds and as such it is highly suitable for application towards systems' self-governance. Systems are regarded as being composed of infinitely many successive actions (transitions) between infinitely many situations (system snapshots). A situation is regarded, in the modern incarnation of the calculus (Levesque et al, 1998), as a sequence of actions or action history, whereas originally (McCarthy, 1963) situations were identified with states. Transitions are implied by relevant preconditions to the action and

changes to situation dependent relations and functions (fluents). The basic idea of Situation Calculus is similar to that of FSMs. Situation Calculus exceeds the capability of an FSM by removing the requirement for the design time enumeration of states, prior to runtime. Situations are generated dynamically at runtime taking infinitely many instances, giving Situation Calculus many desirable properties in modelling and reasoning over dynamic, open worlds. Situation Calculus also gives a rich descriptive formalism with precondition axioms to ensure correct action execution, successor state axioms to represent the execution effects and complex action constructs to express elements of programming languages such as sequential, conditional and recursive execution series. Methods to address the management of situation data and the ever-increasing length of the action-sequence-based situations are proposed later in this work. The solution to the frame problem (Reiter, 1991) and methods of handling complex actions, temporal concerns, sensing actions, epistemic representation and reasoning on stochastic actions make Situation Calculus very relevant to this thesis.

The Stochastic Situation Calculus provides all the necessary specification techniques to proceed with this work. In subsequent chapters the observer system will be defined and methods will be shown, specified for use by the observer system. This includes the specification of system norms for the participating components, the specification of federated behaviour that is assessed through the observer system, the specification for engineering emergence and the specification of deliberation techniques. Additionally various properties of large-scale systems will be derived for use by the Observation System. This will be illustrated through implemented simulations/applications that translates the Stochastic Situation Calculus specification into fully coded operational examples. Finally the specification technique is shown in application to two case studies.

Chapter 5

The Observer Model Signal Grounding and Engineering Emergence

The major objective of this work is to provide a formalism to permit the modelling and reasoning necessary for the meta-systems of large-scale complex systems. Much work already exists on the top-down approach of policy and model based systems, for instance. Some works are also in progress to address bottom up methods of engineering known emergence. These facets are encompassed by the work in this thesis and will be demonstrated through the formal setting of the Situation Calculus in the remaining subsequent chapters. Additionally, however, this formal setting may be used to provide deliberative facilities, to the meta-systems, for reasoning over the observed state of the system. This is done entirely through the logical formalism allowing deduction and other logical procedures to be used automatically to predict future system performance/operation and detect component interactions that lead to a particular state. This leads to three distinct areas of work:

- Firstly, when a particular state is observed to occur the sequence of actions that preceded that particular state/situation can be analysed, through the formal setting, thus grounding the occurrence of useful/interesting states for the runtime system.
- Secondly, the low-level interactions that preceded instances of self-organising behaviour can be replicated at design time for engineering emergence.
- Thirdly, when a particular occurrence of a recognisable emergent or self-organising behaviour is observed, which has not necessarily been engineered for, then the properties associated with the detected behaviour can be utilised by the meta-system.

The operative function through these processes is observation. This observation and the methods of achieving reliable observation are crucial to this work. Thus, based on the collectivist approach of hierarchical reductionism, an observer system will be proposed that allows the system to be seen as component parts, of a single parent level, with interactions that cause the whole system behaviour not to follow in a linear manner, but rather emerge from the interactions of the components. This will then be used firstly to

address the Signal-Grounding problem, as outlined above. Secondly, it will be shown how a known emergent behaviour may be implemented, through the Observer System, as a Markov Decision Process through the Stochastic Situation Calculus. Thirdly, a particularly relevant class of self-organising behaviour, Scale-Free systems, will be assessed that has been observed to have a very close association to large-scale man made and natural systems. This scale-free behaviour has many interesting and useful properties that may be utilised by a deliberative meta-system.

5.1 The Observer System

The proposed Observer System is built around the deployment of appropriate monitoring and sensing modules, with guards to bound component autonomy and ensure legitimate operation, for the system components. These components may be further reduced to component level with appropriate monitoring and guard facilities. The behaviour of the components is thus viewed at a reductionist level for the purposes of modularisation but the interactions can be assessed at the global (containing) component level. This gives the hierarchical reductionism of the collectivist case where the global behaviour of the containing system is not necessarily a linear combination of the comprised component behaviour. Figure 5.1 gives a view of two levels, at a particular point, in a system.

Each component, at the illustrated level of the system, has its own domain specific observing module that represents the intrinsic knowledge of the component. This is relayed and stored in a distributed tuplespace where the observer of the containing system can reason on the state of individual components and more importantly the behaviour that emerges from the interactions of the n components, in this case. Also norms can be passed down the system, through the observers, to bound the autonomy of the individual components The distributed tuplespace is an application of adaptive middleware for co-ordination, as discussed in Section 3.3.2. Through this representation the formal account can be expressed from any reasonable perspective: The norms for individual components can be stated as well as the deliberative mechanisms for the observers, for instance.

The flexibility of this Observer Model lies in its self-similar structure at each hierarchical level of the system. Each, separated out, observer monitors a set of components, which may themselves consist of components with separate Observer Systems. Thus the system is open-ended in either direction through the hierarchy and may be followed upwards,

where more high-level goals may be set, or downwards to ever-smaller components, where more low level functional goals will be satisfied.

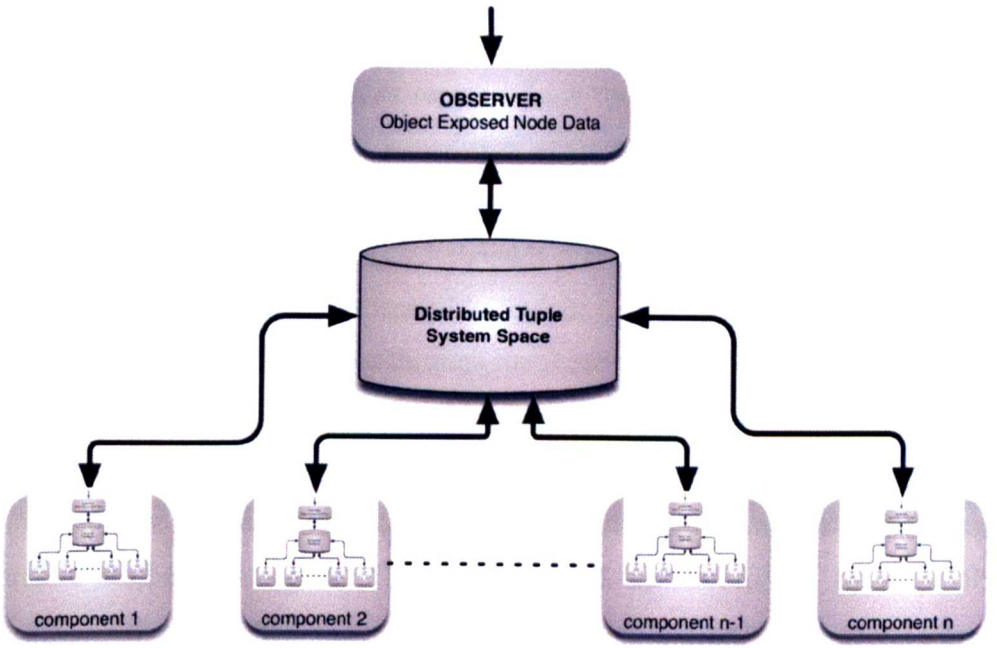

Figure 5.1 The Observer System

At each level in the system the Observer System takes the same structure although the detail would differ according to context with the deployment of the most appropriate probes for the part of the system being monitored. In this way data emanating from a low level component can be passed up the observer hierarchy until a level is reached where the data has intrinsic meaning for that particular observer. This may even result in human intervention, as any observer system is necessarily bound to culminate in human level assessment. This allows each component level, in the hierarchy, autonomy of operation bounded by the purpose of the system as defined and enforced by the associated Observer System. Furthermore cooperation and coordination is achieved between components at the same or different levels through the abstract communication facilities provided by the distributed tuplespace. Such a system is akin to the recursively defined functions in the Viable Systems Model (Beer, 1979), briefly reviewed in Section 3.2. The components in the Observer Model represent the S1 units of the Viable System Model with the varying levels of management (S2, S3, S3*, S4 and S5) abstracted and distributed over appropriate observers in the Observer System.

An Observer System conceived of in this way addresses many of the issues identified as necessary attributes for a Knowledge Plane over the Internet (Clark et al, 2003), as discussed in Section 2.6. *Edge Involvement* is achieved by the pervasive nature of the observation permeating through to appropriate monitors in the system. The separation of concerns, treating the observation as separate from the application, allows the formation of a *Global Perspective*, through knowledge aggregation at the observers, identifying heterogeneity and deploying the appropriate monitors and guards accordingly. *Compositional Structure* is maintained through the rigorous approach of mathematical logic and deliberation allowing federated behaviour between non-homogeneous systems. The *Unified Approach* is developed through the Stochastic Situation Calculus providing a rich representational medium and a mechanism to permit deliberation through logical entailment. This leads into the provision of a *Cognitive Framework* where reasoning can occur in the face of partial or conflicting data with resolution realised through the deliberative stochastic nature of the formalism and the reward/cost based MDP type analysis respectively. In this way system optimisations and responses to environmental problems can be handled through the already established means of data handling but with the additional benefit of meaningful deliberation based on a systemic comprehension of the intrinsic significance of sensing data. Thus functions presently undertaken only by skilled operatives may be achieved through automated functions.

5.2 Addressing the Signal Grounding Problem

The Observer System, as described in the previous section, provides a means to work with the Signal-Grounding problem, as outlined in Chapter 3. The relevant data will travel up the hierarchical Observer System until a module is reached where the data has an intrinsic meaning for a particular observer: Rather than simply providing a stimulus to action the piece of system knowledge will have a precedent and likely consequence to the system that may be grounded at this appropriate observer module. Conversely norms grounded at a certain level in the system can be relayed down to the individual components for which the norm has no intrinsic meaning but is simply a prescription to be followed. Using the grounded signals within the Situation Calculus representation allows the evolution of a dynamic self: The circumstances that preceded a situation may be extracted giving a prediction for actions in a specific context. Reasoning, using deduction, abduction, induction or inference, can then be performed on the logical

representation to supply receptors for perceived signals. In this way new interactions that cause no harm, and may be beneficial, are allowed.

For example the action history represented by:

$$do(a,do(a_l,do(a,s))) \text{ with } SR(a,s)\neq SR(a, do(a_l,do(a,s)))$$

where SR(a,s) and a=sense$_f$ for some fluent f are as defined in Section 4.4.4 and a_l is some deterministic action can be used to provide a new prediction for the results of action a_l where the values of other fluents in situation s form the action precondition axioms for a_l as a context. In this way, action a_l, executing in the context of situation s, grounds the signal for f. So a signal may be grounded by the system and a parameter for autonomic type response can be adapted at runtime. For example, using the previously defined sensing and knowledge constructs from the Situation Calculus, suppose a service usage is monitored via a CPU load sensor within the Observer System: A fluent *heavyLoad(s)* is true if the CPU is working at over 60% capacity:

$$cpuload(do(a, s)) = n \Leftrightarrow [cpuload(s) = n \wedge a \neq sense_{CPULOAD}] \vee [a = sense_{CPULOAD} \wedge SR(sense_{CPULOAD}, s) = n]$$

$$heavyLoad(do(a, s)) \Leftrightarrow [heavyLoad(s) \wedge ((a \neq sense_{CPULOAD}) \vee \neg(a = sense_{CPULOAD} \wedge SR(sense_{CPULOAD}, s) < 60))] \vee [a = sense_{CPULOAD} \wedge SR(sense_{CPULOAD}, s) > 60]$$

Thus an action a_l may be assigned a predicted outcome via the construct:

$$do(a,do(a_l,do(a,s))) \text{ with } SR(a,s)\neq SR(a, do(a_l,do(a,s)))$$

with $a=sense_{CPULOAD}$ to deduce:

$knows(heavyLoad,s)$ and $knows(\neg heavyLoad,do(a_l,s))$

In this way the action a_l can form the action that returns a system to a required predicted state, based on a grounded signal for heavy CPU load.

This illustrates the idea of how grounded signals may be extracted or adapted, within a running system, to be used as autonomic type responses. In relaying deliberation to higher cognitive entities the Symbol/Signal Grounding problem is circumvented, in some cases, by the evaluating authority being a human mind. In this case the intrinsic meaning of the signals is apparent to the expert operator. However it has been shown here that these same signals may be manipulated by the system's cognitive facilities to adapt the grounding of the signal to the practical system's actual operating environment.

Additionally new signals can be ascertained and assessed by using the usual reasoning methods of mathematical logic as outlined.

Additionally a mechanism needs to be included to determine the relevance of the response. In this way correct responses are reinforced whilst poor results lead to the perceived relevance of the signal diminishing, eventually leading to the signal response losing all significance to the system, in cases where the grounded signal was detected in error. An example simple reward system may consist of, for instance:

$$reward(regenerateService(S),s)=r \Leftrightarrow S=service_1 \wedge r=80 \vee S= service_2 \wedge r=60 \vee service(S) \wedge S \neq service_1 \wedge S \neq service_2 \wedge r=40$$

$$reward(sense_{CPULOAD}, s)=10$$

$$cost(allocateMemory(n, S),s)=0.1*size(n)$$

The action *regenerateService(S)* represents a class of actions depending on the stimulus that caused the action to be considered for enactment. For instance, in the example from the previous section, it may be

regenerateService(S).heavyLoad or regenerateService(S).unresponsive

according to whether the action was undertaken in response to heavy CPU load or system non-responsiveness.

Additionally the rewarding of the action needs to be moved to the successor state, as it is only here that the success of the prediction can be determined. Thus, for example it may be stated using a *reward* for the action performance to cumulatively influence a *fitness* function for the grounded response:

$$reward(regenerateService(S).heavyLoad, do(a,s))=r \Leftrightarrow a=regenerateService(S) \wedge [(r=10 \wedge \neg heavyLoad(do(a,s))) \vee (r=-10 \wedge heavyLoad(do(a,s)))]$$

with

$$fitness(regenerateService(S).heavyLoad, do(a,s))= reward(regenerateService(S).heavyLoad,do(a,s))+ fitness(regenerateService(S).heavyLoad, s)$$

In this formulation the more successful a prediction of response to stimulus is the higher the fitness value of the successor state axiom. Thus axioms that fall below a certain

threshold can be discarded whilst those with a higher fitness can be treated as first class axioms with a corresponding increase in their reward value compared to an axiom of equal merit but lower fitness. Thus the Signal-Grounding problem is addressed by providing preinstalled, human level, grounding of some initial signal set. The current state of research means that cognitive systems, culminating at human level, are required to provide grounding for the signals emanating from the system, in order to evolve responses for self-governance. However, in this work, grounded signals and appropriate responses are also extractable automatically from the runtime system. The formalism permits the critical assessment and requisite adaptation, through the use of the decision theoretic techniques involving utility, fitness and value.

5.3 Emergent Behaviour in Large Scale Complex Systems

The previous subsection indicated the methods by which an Observer System may detect new signals and seek to assign intrinsic meaning to the data emanating from a system. The second important facet of emergent behaviour identified is the interactions that led to a known behaviour occurring. In such cases the complexly interwoven yet simple low-level interactions that precipitated some global emergent phenomenon may be isolated and reused to bring about future instances of the associated emergent global event: Signal grounding is necessary at runtime to take advantage of newly discovered behaviour whereas previously detected emergent behaviour may be used at design time to engineer emergence. Thus the self-organising behaviour such as foraging, herding, nest building moulding, web weaving and brood sorting, as discussed in preceding chapters, may be used as the basis of models to achieve the observed desired outcomes: A particular emergence may be engineered. For instance, the emergent outcomes observed in foraging have been termed as stigmergy (Grassé, 1959). Although Grassé introduced the term stigmergy to explain the behaviour of termite societies, the same term has been used to describe indirect communication mediated by modification of the environment that can also be observed in other social insects. More recently this approach has been adopted for manufacturing systems (Hadeli et al, 2005). A stochastic Situation Calculus approach is especially useful in this regard as treating the scenarios as Markov Decision Processes, without the requirement for a state space enumeration, allows the parameters to be assessed and adjusted optimally through the formal setting, whilst preserving the benefits of self-organisational emergence.

5.3.1 Markov Decision Processes and Natural Self-Organising Systems

A Markov Decision Process (MDP) is a model M = < *S,A,T,R*> where

- *S* is a set of environmental states
- *A* is a set of actions
- *T* is a transition function *T*: $S \times A \times S \rightarrow [0,1]$ where $T(s_1,a,s_2) = p(s_2|s_1,a)$: The transition function gives the probability of entering a state s_2 when action a is executed in state s_1.
- *R* is a reward function, $R: S \times A \rightarrow \Re$: This gives a real valued reward, *R*(s,a) for performing action a in state s.

A policy, in an MDP, is a function π: $S \rightarrow A$ that forms a plan in which an action is chosen according to the current state. For each policy, there is an associated expected cumulative reward, $V^{\pi}: S \rightarrow \Re$, which is satisfied by the Bellman Equation:

$$V^{\pi}(s) = R(s, \pi(s)) + \gamma \Sigma_{s'} T(s, \pi(s), s') V^{\pi}(s')$$

where $0 \leq \gamma < 1$ is a discount on the value of future rewards. The goal is to find an optimal policy π^* with $V^{\pi^*}(s) \geq V^{\pi}(s)$ $\forall s \in S$ and π, denoted $V^*(s)$. The most usual method of solution is by value iteration (Bellman, 1957): A series of Q-functions are recursively defined by setting $V^0 = R$ with V^n termed an n-stages to go value function and

$$Q^n(s,a) := R(s, a) + \gamma \Sigma_{s'} T(s, a, s') V^{n-1}(s') \text{ for any a}$$

For every state s, a Q-function $Q^n(s,a)$ is computed by using V^{n-1} for any action a, and $V^n(s)$ equals the maximal value of Q^n according to a: $V^n(s) := max_a Q^n(s,a)$

The Q-function $Q^n(s,a)$ denotes the expected value of performing action *a* at state *s* with *n* stages to go and acting optimally thereafter. The sequence of value functions V^n produced by value iteration converges linearly to V^*. It is clear that at each derivation of V^n it is necessary to search the entire state space. To proceed with this approach the state and action spaces need to be explicitly enumerated. In contrast the merits of using a propositional account have already been established. There are a number of proposals for representing MDPs in a propositional account. For instance it is possible to provide a probabilistic variation of STRIPS (Boutilier et al, 1999) or use stochastic actions in a first order representation (Poole, 1997). However following the formal setting of stochastic Situation Calculus it is possible to use Symbolic Dynamic Programming to look at

solutions for MDPs (Boutilier et al, 2001) using the representation described in Section 4.4.5. The functions to represent reward, probability and value are best described in a case notation so, for instance,

$$V(s) = case[\Phi_1(s), v_1, \ldots, \Phi_n(s), v_n]$$

means that the result of the value function in situation s is v_i if the formula $\Phi_i(s)$ (for $0<i\leq n$) consisting of domain fluents, is true. Then the Q-functions are given by:

$$Q^n(a(x),s) = R(s) + \gamma \Sigma_i \, prob(n_i(x),a(x),s) \; V^{n-1}(do(n_i(x),s)) \qquad [A]$$

The value function value may then be logically stated as:

$$V^n(s) = (\exists a)(\forall b) \; Q^n(a,s) \geq Q^n(b,s)$$

Now it is possible to consider previously described naturally occurring self-organising behaviour in terms of the stochastic Situation Calculus. For example consider the foraging behaviour where ants transfer food particles from food sources to their nest. This is a relatively simple domain consisting of locations, ants and food particles; ants can pick up and drop food particles and move between locations. Accordingly, following the procedure for stochastic actions outlined in Section 4.4.5, it is necessary to specify the deterministic outcomes for each stochastic action. For example the *pickUp* action can succeed or fail, denoted *s_pickUp* and *f_pickUp* as appropriate:

$$choice(pickUp(ant, f), a) \equiv a = s\_pickUp(ant,f) \vee a = f\_pickUp(ant,f)$$

Similarly the stochastic actions move and drop can be decomposed in the same manner into successful or unsuccessful alternatives. The probability, $prob(n_i(x),a(x),s)$, can be specified for each of the choices $n_i(x)$ associated to the action $a(x)$ executed in s:

$prob(s\_pickUp(ant,f), pickUp(ant,f),s) = 0.95$

$prob(f\_pickUp(ant,f), pickUp(ant,f),s) = 0.05$

$prob(s\_drop(ant,f), drop(ant,f),s)=p \equiv wet(s) \wedge p=0.6 \vee \neg wet(s) \wedge p=0.9$

In the above-defined case notation this can be written:

$prob(s\_drop(ant,f), drop(ant,f),s)=case[wet(s),0.6;\neg wet(s),0.9]$

Similarly for the opposite deterministic primitive of the drop action:

$prob(f\_drop(ant,f), drop(ant,f),s)=1- prob(s\_drop(ant,f), drop(ant,f),s)$ is equivalent to:

$prob(f\_drop(ant,f), drop(ant,f),s)=case[wet(s),0.4;\neg wet(s),0.1]$

$prob(s\_move(ant,l)\ move(ant,l),s)=p \equiv pheromoneTo(l,s) \wedge p=0.99 \vee$

$\neg pheromoneTo(l,s) \wedge p=0.8$

$prob(f\_move(ant,l)\ move(ant,l),s)=1-prob(s\_move(ant,l)\ move(ant,l),s)$

Thus it can be seen that dropping a particle of food is more likely to succeed in dry conditions and a move to a new location is more likely to succeed if there is an existing pheromone trail to that position.

The described deterministic actions will have action precondition axioms:

$poss(s\_pickUp(ant,f),s) \equiv (\exists l) foodAt(f,l,s) \wedge antAt(ant,l,s)$

$poss(f\_pickUp(ant,f),s) \equiv (\exists l) foodAt(f,l,s) \wedge antAt(ant,l,s)$

$poss(s\_drop(ant,f),s) \equiv carrying(f,ant,s)$

$poss(f\_drop(ant,f),s) \equiv carrying(f,ant,s)$

$poss(s\_move(ant,l),s) \equiv true$

$poss(f\_move(ant,l),s) \equiv true$

The successor state axioms are:

$foodAt(f,l,do(a,s)) \Leftrightarrow \exists ant[antAt(ant,l,s) \wedge a=s\_drop(ant,f)] \vee$

$[foodAt(f,l,s) \wedge \neg(\exists ant) a=s\_pickUp(ant,f)]$

$antAt(ant,l,do(a,s)) \Leftrightarrow a=s\_move(ant,l) \vee [antAt(ant,l,s) \wedge \neg(\exists l') a=s\_move(ant,l')]$

$carrying(f,ant,do(a,s)) \Leftrightarrow a=s\_pickUp(ant,f) \vee [carrying(f,ant,s) \wedge a \neq s\_drop(ant,f)]$

$wet(do(a,s)) \Leftrightarrow wet(s)$

$pheromoneTo(l,do(a,s)) \Leftrightarrow pheromoneTo(l,s)$

This gives a stochastic Situation Calculus account of the domain. In addition pheromone levels could be adjusted through ants moving and depositing pheromone whilst carrying a food particle with a suitable decaying function for the level at a particular location.

To move now towards representing this scenario as an MDP a reward function can be specified: $R(s)=case[(\exists f) foodAt(f,nest,s),100;\ \neg(\exists f) foodAt(f,nest,s),0]$

The Q-functions, $Q^v(a(x), s)$, can be characterised by a single case statement, using equation [A] and the fact that the reward, probability and value functions can all be represented as case statements. The problem, however with equation [A] is the presence

of a successor situation in the term $V^{n-1}(do(n_i(x),s))$. This can be handled using classical regression techniques (Reiter, 1991). This is possible because the actions, although stochastic in nature have been decomposed into deterministic primitives. Thus applying regression gives the properties of the pre-action situation that provide the properties of the post-action state that effect the value function. So $Q^V(a(x),s)$ can be derived as a series of case statements such as: $case[\alpha_i(x,s),q_i]$ that gives the Q function for the action $a(x)$ with respect to V. At this point allowance has been made for action precondition axioms; it is not clear whether any of the $n_i(x)$, as choices for the stochastic action $a(x)$ are actually possible. Thus the Q- value forms a relation $Q^V(a(x),q,s)$ meaning that $a(x)$'s Q value in s is q. This relationship is undefined unless one of the $n_i(x)$ are possible in s.

$$Q^V(a(x),q,s) \equiv [\bigvee_i (poss(n_i(x),s))] \wedge q = case[\alpha_i(x,s),q_i]$$

For example consider the value function V^0 regressed through the action $drop(ant,f)$ with reward $R=V^0$, discount value $\gamma = 0.8$ and:

$V(s) = case[(\exists f)foodAt(f,nest,s),100; \neg(\exists f)foodAt(f,nest,s),0]$

That is if some food is in the nest then the value is 100 otherwise it is 0.

The resulting value for $Q^1(drop(ant,f),q,s)$ is a case statement with four elements, after removing inconsistent and identical cases:

$\alpha_1(ant,f,s) \equiv \exists f' foodAt(f',nest,s)$

$\alpha_2(ant,f,s) \equiv wet(s) \wedge antAt(ant,nest,s) \wedge carrying(f,ant,s) \wedge \neg \exists f' foodAt(f',nest,s)$

$\alpha_3(ant,f,s) \equiv \neg wet(s) \wedge antAt(ant,nest,s) \wedge carrying(f,ant,s) \wedge \neg \exists f' foodAt(f',nest,s)$

$\alpha_4(ant,f,s) \equiv (\neg antAt(ant,nest,s) \vee \neg carrying(f,ant,s)) \wedge \neg \exists f' foodAt(f',nest,s)$

The associated values for the q_i (i=1-4) are: $q_1=180$, $q_2=48$, $q_3=72$ and $q_4=0$

The representational power of Situation Calculus may thus be observed in this example of engineering emergence. The model permits the tuning of parameters, which is utilisable by the Observer System, through a MDP, to engineer various desirable properties. The example given here is quite a simple scenario. Additional parameters, however, may be specified to better capture the dynamics present in the natural domain: For example a measure of the quantity of food in the nest can be added or the Q-value of the *move* action can depend on the strength of pheromone present in that situation with the value term in the case statement dependent on the pheromone strength.

5.4 Summary

This chapter has presented a proposed Observer System based on a collectivist behaviour model that is hierarchically reducible solely at each individual component level. The logical specification and assembled metrics allows deliberation to proceed on the observed operation of the system. The Observer System, together with the Stochastic Situation Calculus, gives a means of specifying self-governance to varying scales of system. To illustrate this two techniques have been proposed for use by the Observation system with its associated cognitive functions. Firstly the specification was shown of how the cognitive Observer System would address the Signal-Grounding problem. In this way newly evolving behavioural models could be incorporated into the specification at runtime. Secondly it was shown how the Observer System would achieve a controlled engineering of emergence for the governance of its associated application system: Parameters are adjusted according to reasoning based on solutions to a Markov Decision Problem. The next chapter details the manner in which large-scale systems evolve and the properties associated with the most prevalent form of emergent organisation. Thus the Observer System may then be endowed with facilities to harness the features of these systems when their emergence is detected.

Chapter 6

Properties of Large-Scale Networks

In the previous chapter the bottom-up perspective on systems design has been considered through the application of a separate Observer System. Firstly an approach was illustrated so that the Observer System could extract useful action sequences that gave rise to some previously unpredictable occurrence. Secondly it was shown how the Observer System could influence, through a formal model, the evolution of a system providing for the engineering of a particular known emergent behaviour. Next, in this chapter, generic properties of complex networks and particular classes of emergent organisation will be considered that has been observed to be prevalent in large-scale, complex systems. Thus the Observer System upon detecting some class of global system behaviour will be able to utilise the properties of the system to either infer some monitoring data, thus scaling down its sensing operations, target resources for better robustness based on the topology of the system or influence the system to assemble in a particular desirable way.

6.1 Properties of Large-Scale Complex Systems: Scale-Free Systems

The observed topological structure of the World Wide Web and Internet was briefly discussed at the end of Chapter 2. This feature of power law connectivity is one aspect being increasingly observed as a property in many man-made and natural complex systems. Thus the interactions that result in such global phenomenon and the architectural properties of the systems exhibiting this behaviour are of importance to the work presented here. Thus the recognition and properties of such behaviour requires further investigation. It is through the use of Statistical Mechanics that the development and features of these systems are explored.

Complex networks have usually been described through the use of graph theory. Simple designed networks, such as electrical circuits, can be completely described by regular graphs; large-scale complex systems, however, are too interwoven and complicated for any design pattern to be apparent: Observed events and evolution appear to occur in a totally random manner. Thus modelling the systems as random graphs with the same number of edges and vertices as the system under investigation would seem to be a reasonable approach: The properties of the random graph ought to correspond to properties in the actual system. Random graphs, in relation to large-scale networks were

first represented by the Erdös-Rényi (ER) model (Erdös and Rényi, 1959), where there are N nodes connected to each other with a probability p. Thus the maximum number of connections, when all nodes are connected is $N(N-1)/2$ resulting in the expected number of connections being $pN(N-1)/2$, distributed randomly. This model has been dominant in the study of complex networks until very recently when organizational structure was observed to be necessary for the functioning of large-scale complex systems, such as the World Wide Web. If complex systems deviate from random topologies then metrics and measures need to be developed to capture the underlying organizational principles. Whilst many metrics and measures have been proposed for complex networks, three main concepts have emerged as the major factors in the analysis and characterisation of these complex systems:

1. The *Small World* concept characterises the property that no matter what size a complex network might grow to be there always seems to be, in most cases, a relatively short path between any two nodes. The distance between two nodes is defined to be the number of connections on the shortest path between the two nodes. Thus the distance between two nodes does not depend on the usual Euclidean distance but on the interconnections between them. This was noted in the "six degrees of separation" principle (Travers and Milgram, 1969) where it was shown there is a very short path of acquaintances between any two people. The property seems to persist through many large-scale systems. The small world concept however is not evidence of some organising principle. Indeed random graphs are examples of small worlds because if the connection probability $p > ln(N)/N$ then a random graph will be connected (Erdös and Rényi, 1961). This means that a path can be found between any two nodes in the system. The typical distance, on a connected graph, between two nodes scales as the logarithm of the number of nodes and thus tends to be quite small, making even random graphs examples of small world phenomena.

2. *Clustering* tends to occur in social networks where circles of friends or acquaintances occur where every member knows every other member, forming a clique. This tendency in large-scale systems may be represented by a clustering coefficient (Watts and Strogatz, 1998). The clustering coefficient for a particular node i is given by the ratio of the number of edges between the nodes that i is connected to and the maximum number of edges that could occur between the nodes i is connected to. Thus if i has k_i edges connecting it to k_i other nodes (termed i's associates) then the maximum number of connections that could occur between i and all its associates is

$k_i(k_i-1)/2$, for undirected links. If the actual number of edges that occur between all i's associates is E_I then the clustering coefficient for i is given by: $C_i = \frac{E_i}{k_i(k_i-1)/2} = \frac{2E_i}{k_i(k_i-1)}$. The clustering coefficient for the whole network is then given by the average over all the nodes: $C = \frac{1}{N}\sum_i C_i = \frac{1}{N}\sum_{i=1}^{N} \frac{2E_i}{k_i(k_i-1)}$. Thus the clustering coefficient of random graphs is quite small: The probability that the associates of a node are connected, on a random graph, is the same as the probability that any two nodes are connected. Thus the clustering coefficient for any random graph is p. In the vast majority of real world networks however it may be observed that the clustering coefficient is very much greater than in a random graph with the same number of edges and vertices (Watts and Strogatz, 1998). This represented the initial indication that random graphs could not account for all the behaviours in large-scale complex networks. Furthermore empirical studies indicate that the clustering coefficient appears independent of network size in many real world systems (Albert and Baraási, 2002). This is the case for ordered lattices where the clustering coefficient depends only on the coordination number (the number of adjacent nodes each node is connected to). For instance Figure 6.2 shows a regular lattice ring of 20 nodes with coordination number 4.

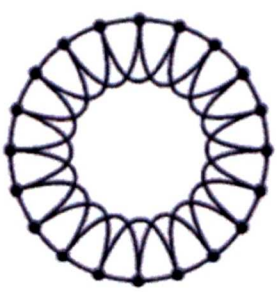

Figure 6.1: Regular ring lattice where each node is connected to its 4 nearest neighbours

If each of the lattice nodes is connected to the K nodes closest to it then the clustering coefficient is $C = \frac{3(K-2)}{4(K-1)}$ which converges to ¾ for large K. Regular lattices, however do not in general display short path lengths between nodes, as the nodes furthest away from a neighbourhood cluster, on the opposite side of the ring, for

instance, will require there to be many nodes on the shortest path between them. The Watts-Strogatz model was the first attempt to model real world systems by providing a method of generating graphs with both high clustering and short path lengths across the system (Watts and Strogatz, 1998). The model is generated by commencing with a regular ring lattice of N nodes on which every node is connected to its K nearest neighbours ($K/2$ on either side). Then each edge of the lattice is randomly rewired with probability p omitting duplicate edges and self-connections. This introduces $pNK/2$ long-range edges that connect nodes that would otherwise be part of different neighbourhoods. The variation of the p parameter interpolates between a regular lattice and a random graph, as shown in Figure 6.3. This is for a ring lattice with $N=20$ and $K=4$. A node is chosen and the edge that connects to a clockwise neighbour. With probability p this edge is connected to a randomly selected node. This process is carried on until all nodes have been considered once. The figure indicates the results for different values of p: For $p=0$ (total order) the original lattice remain unchanged, as p increases the network becomes increasingly disordered until at $p=1$ (randomness) all the edges are randomly rewired. So at values of p a high clustering coefficient is obtained together with the small world property of short path lengths.

Figure 6.2: Random rewiring in the Watts-Strogatz model (Watts and Strogatz, 1998)

3. *The degree distribution* for a network recognises the fact that not all nodes in the network possess the same number of connections. There is a range in the number of edges associated with a node, termed its degree. This range is given by a probability distribution $P(k)$, the likelihood that a node has degree k: The probability that a

randomly selected node has exactly k edges. In random graphs, because every node is equivalent, all nodes have a degree very close to the average degree for the network, $<k>$. Thus the degree distribution is a Poisson distribution with a peak at $P(<k>)$. As outlined in Section 2.6 this differs significantly from the observed distribution for the World Wide Web (Albert et al, 1999), which, in common with the Internet (router level graphs) (Faloutsos et al, 1999) and cellular networks (Jeong et al, 2000) amongst many others, display a degree distribution with a power law tail: Instead of the Poisson distribution that would be expected in the random graph representation $P(k) = e^{-z}z^k/k!$ (Bollobás, 1985) where $z=<k>$, the average degree, the distribution is given by $P(k) = ck^{-\lambda}$ for some λ with $k=m,....M$ where m and M are the lower and upper bound respectively of the nodes' connectivity and c is a constant normalisation factor. This distribution of node degrees suggests that there will be many nodes possessing a low connectivity but, crucially in the tail of the distribution, a small number of nodes with a very high degree are expected. Thus there is a lack of scaling around the mean connectivity leading to such systems being termed *scale-free*. This means that systems attaining or passing certain thresholds exhibit simplified or self-organising behaviour despite their complex nature. It may be observed that the scale-free type network will be highly resistant to the random removal of nodes, as most nodes have low connectivity, but may be susceptible to specifically targeted attacks, against the few highly connected nodes. This has prompted such networks to be labelled as robust yet vulnerable. Such a system is illustrated in Figure 6.4 where the large square nodes can be observed to form a hub connection backbone for the system. The removal of one of these larger hubs may seriously compromise the integrity of the system, whereas removal of one of the smaller (less connected) nodes would hardly be noticed. These properties may be utilised by a cognitive observer system to identify and protect vulnerabilities whilst benefiting from the underlying robustness and fault tolerance of the system.

These discoveries of the concepts and properties of small path length, clustering and power law degree distributions have changed the manner in which large-scale-complex systems have been studied over the past few years. There are three distinct models for large-scale network systems: Firstly random graphs are still applicable to many classes of systems and act as a benchmark on which to base the observed characteristics of complex network systems. Secondly small world models interpolate between random graphs and

highly clustered regular lattices. Thirdly power law distributions for large system connectivity has introduced many models of scale-free behaviour that attempt to explain and promote the power law tails, in the distribution, through specific low-level interactions amongst the system participant components.

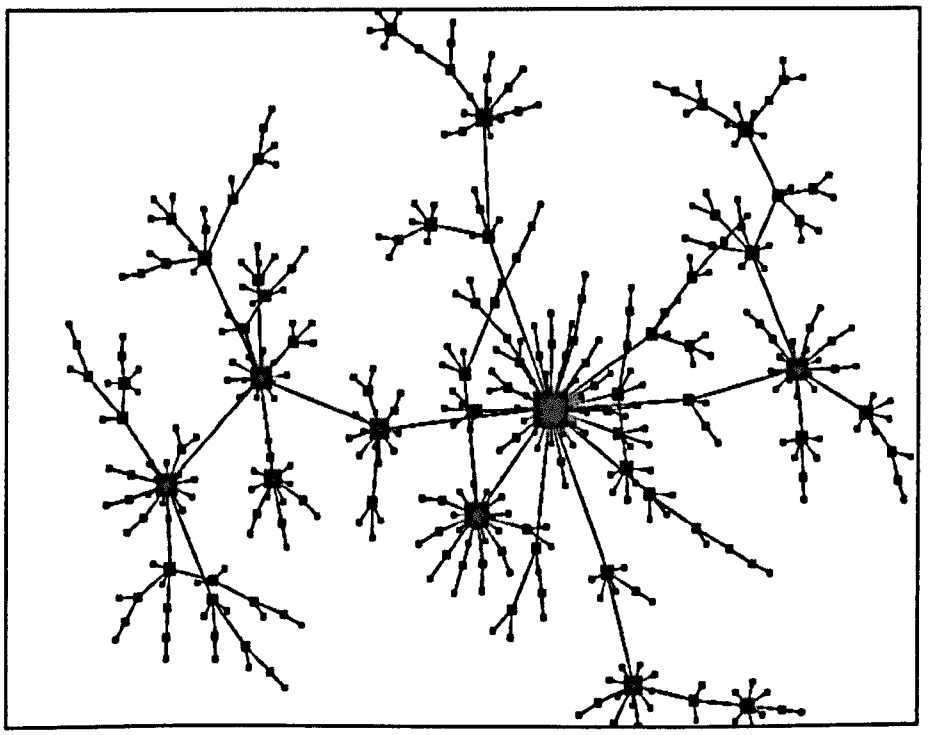

Figure 6.3: Typical topology of a scale-free system

In contrast to scale free systems, small world networks are most usually homogeneous in their degree, with most nodes possessing a degree very close to the average, and cannot exhibit system growth as the node number is fixed. Many real world networks exhibit very strong growth and, as previously noted, are not homogeneous in node degree. It is for this reason that the metrics and measures used through this work will be presented through a scale free perspective: Scale-free behaviour has been widely observed to be a very common feature in large-scale complex systems. Thus by harnessing the properties of this complexity it ought to be possible to promote more scaleable methods of achieving system self-governance by the detection and utilisation of scale-free behaviour.

6.2 Scale-Free Properties and Metrics

Scale-free topologies have been observed to occur as a property in many different forms of large-scale complex systems. The Internet, as routers (vertices) and links (edges), and the World Wide Web, as Web pages (vertices) and Hyperlinks (edges), has already been

noted as primary examples of this phenomenon in man-made systems (Albert et al, 1999 and Faloutsos et al, 1999). Additionally, as more data on the topological structure of large networks become available, it is apparent that this organisational behaviour is extremely widespread in complex systems: Actor collaborations (Barabási and Albert, 1999), scientist collaborations (Newman, 2001), metabolic networks (Jeong et al, 2000), ecological systems (Montoya and Solé, 2000), citation networks (Redner, 1998), telephone call patterns (Aiello et al, 2000), flare patterns on the sun (Paczuski and Hughes, 2004), civilian war casualties (Alvarez-Ramirez et al, 2007) and even the goal scoring propensity of Brazilian soccer players (Onody and de Castro, 2004), amongst many more examples, have all been shown to organise in a scale-free manner. These examples are summarised in Table 6.1 for degree distributions given by $P(k) \sim k^{-\lambda}$.

Neither random graphs nor small world networks reproduce this significant organisational property observed so widely in the real world (Albert and Barabási, 2002). The mechanisms for producing such features and the metrics and measures associated with the component interactions and global system outcome are vital for efficient deliberation by an observer system on a large-scale network system. There are two generic aspects of real systems that are absent from random graphs and small world networks; namely growth and preferential attachment. The ER model assumes a fixed number of nodes that connect with uniform probability, whilst small world networks likewise assume a fixed node set that reconnects according to distance. In contrast most real systems are open, the number of nodes varies with time, and most commonly the network continuously expands by the addition of new nodes that connect to nodes already present in the system. Additionally most real networks do not display random dynamics for connecting up nodes rather they exhibit tendencies to preferential attachment, such as the likelihood of connecting to another node is dependent on that nodes degree. For example a newly created Web page is more likely to have hyperlinks to already well-known popular pages with an existing high connectivity.

The Barabási-Albert method for constructing scale-free systems combines these properties through growth and preferential attachment into two steps (Barabási and Albert, 1999):

- *Growth:* Starting with a number of nodes, m_0 add a new node, at each point in a series of time steps, which connects to m ($\leq m_0$) existing nodes with m edges.

- *Preferential Attachment:* When selecting the nodes to attach to it is assumed that the probability P that a new node connects to node i is dependent on the connectivity k_i of that node, so that:

$$P(k_i) = \frac{k_i}{\sum_j k_j}$$

Table 6.1: Scale-free systems with $P(k) \sim k^{-\lambda}$

System	λ	Reference
World Wide Web	2.1	Albert et al, 1999
Internet	2.2	Faloutsos et al, 1999
Actor Collaborations	2.3	Barabási and Albert, 1999
Scientist Collaborations	1.2	Newman, 2001
Metabolic Networks	2.2	Jeong et al, 2000
Ecological System	1.05	Montoya and Solé, 2000
Citation Networks	3	Redner, 1998
Telephone Calling	2.1	Aiello et al, 2000
Solar Flare Patterns	2	Paczuski and Hughes, 2004
War Casualties	2.5	Alvarez-Ramirez et al, 2007
Brazilian Soccer Goal Scorers	2.44	Onody and de Castro, 2004

This account gives a network where the probability distribution for the connectivity of the nodes is a power law with an exponent of approximately 3 (Barabási and Albert, 1999). This, however, is not a universal network model but rather a minimal expression of the underlying mechanisms in the emergence of scale-free topologies. It predicts a power law degree distribution with a fixed exponent of 3 whereas Table 6.1 shows many different exponents scattered in the interval [1,3].

Extensions to the scale-free model, such as rewiring (Sayeed-allaei et al, 2005) and evolution (Albert and Barabási, 2002) have provided a consistent model of real world scenarios to account for scaling exponents and connectivity cut-offs, for instance. The

scale free model however neglects to take into account an aspect of competitive systems. That is the nodes are not homogeneously attractive irrespective of preferential attachment. Some nodes are better than others at attracting new links. The model predicts that node i's degree k_i at time t is given by:

$$k_i(t) = \sqrt{\frac{t}{t_i}}$$

where t_i is the time at which node i was added to the network. Thus the oldest existing nodes have the highest degree as they have the longest time to acquire them. If this were the case for all systems any new component in a system would have no chance of acquiring links. This is obviously not the case, as, in any real world example system, new successful nodes or components can emerge. This is because some system constituents have a higher fitness value that makes them more adept at competing for links at the expense of less fit components. Thus varying fitness provides extra scaling in that the time dependent connectivity of the node depends on the node's fitness. As detailed above, for preferential attachment, the probability that a new node will connect to node i is given by:

$$P(k_i) = \frac{k_i}{\sum_j k_j}$$

To model the ability of nodes to pick up connections, irrespective of solely its existing connectivity a fitness parameter f_i is assigned to each node. Initially it is assumed that this is unchanged with time. However in real systems fitness will vary with time and this may be handled via the reward and cost functions to produce a fitness value based on selected actions. So now it is assumed that the probability P_i that a new node connects to an existing node, i, depends on the connectivity of i, k_i and the fitness of i f_i where f is chosen from a distribution such that:

$$P_i = \frac{f_i k_i}{\sum_j f_j k_j}$$

This provides the simplest formulation whereby fitness and preferential attachment can be taken into account when new nodes "decide" which existing nodes to link with. This means that even a recently added node with few links can acquire connections at a high rate if it has a sufficient fitness level. If, as in the originally stated Barabási-Albert model,

m links are added per time step then nodes introduced at a later time can also gather a high number of links: A node, i, collects links at a rate:

$$m\frac{f_i k_i}{\sum_j f_j k_j}$$

If the nodes' fitness is homogeneously distributed, that is each node has the same fitness, then the regular scale-free model is exhibited. So $k_i(t) \sim (k)^{1/2}$.

In order to solve these equations it is assumed that as k_i emerges over time it still follows a power law distribution but now there is also a new scaling in the system because of the fitness function.

As such networks such evolve, a similarity to physical systems can be noted. The dynamical properties of the evolving system go through three distinct phases.

- **A Scale-Free Phase** This is when all the nodes have fitness values tightly clustered around a mean-value and the model reduces to the general scale-free model. In this the first mover wins because the older nodes gather the most links with connectivity $t^{1/2}$ so that old nodes with low t_i have a high k_i. However the oldest, most connected nodes are not the outright winners as its proportion of links $k_{max}(t)/(mt)$ tends to 0 in the limit. Thus, as described in (Barabási and Albert, 1999) a connected hierarchy of large hubs coexist with degree distribution given by the expression $P(k) \sim k^{-3}$.
- **A Fit Get Rich Phase** This phase emerges where the nodes have varying fitness's. Nodes increase their connectivity with time but the dynamic exponent is larger for more fit nodes.
- **A Winner Takes All Phase** In this stage the competition for links entails the node with the greatest fitness emerging as the clear winner. There is an emergence of new competing nodes but the fittest always acquire a finite proportion of links

This gives the unique response to node failure, exhibited by scale-free networks, as manifested in the fraction of nodes that can arbitrarily be removed from a network without affecting the system connection integrity. Random networks disintegrate; into isolated clusters after a critical number of nodes have been removed. This critical threshold all but disappears for scale-free networks with degree exponent less than or

equal to 3 (Cohen et al, 2002). These networks break apart only after all the nodes have been removed. In practice this means hardly ever.

As previously stated the Barabási-Albert model provides an indication of the dynamics promoting the observed scale-free behaviour. The addition of a fitness function to represent the nodes ability to gain links further improves the model in relation to real world systems. For instance in citation networks the number of citations a paper receives tends to decrease with age. In the plain Barabási-Albert model, where the rate of attachment is proportional to the degree, the opposite is true: The older the node the more links it will gain. Indeed if the more mature nodes, in the Barabási-Albert model, are disregarded then scale free behaviour ceases to be evident. Scale-free behaviour, however, is extremely prevalent in observed large-scale systems. In order to account for this further models have been proposed to take account of the impact of age on the nodes ability to acquire new links. This amounts to specifying a fitness function on a node that determines its ability to acquire new links.

The Klemm-Eguíluz model (Klemm and Eguíluz, 2002) addresses this problem by seeking the production of a highly clustered scale-free network based on the low-level nodes degree-dependant deactivation dynamics. This model suggests a more directed approach to facilitate a node being able to link to other nodes but not necessarily being able to accept links. A node can be inactive or active: Active nodes can receive links from other nodes whilst nodes are deactivated with probability P and can then no longer receive links. The probability P decreases as the node degree increases. Thus the more links a node possesses the less likely it is to be deactivated: If k is the degree and a is a constant bias then $P \propto 1/k+a$. The dynamics are as follows:

1. Add a new node i to the network with no connections so $k_i=0$
2. Attach m outgoing links from i to m active nodes, such that each of the m nodes receives one extra link.
3. Activate i
4. Deactivate one of the active nodes. The probability that an active node j will be deactivated is given by: $P(k_j) = \frac{1}{(a+k_j)\sum_{l \in A} \frac{1}{a+k_l}}$ The summation in the denominator runs over the set A of the active nodes.

5. Goto 1

This is shown to produce a network structure with higher clustering coefficient of 5/6, larger than in a regular lattice, and scale-free, power law connectivity with exponent $2+\frac{a}{m}$ as a global outcome (Klemm and Eguíluz, 2002). The scale-free networks produced through this method have a resemblance to a regular lattice in the parameters observed. It seems to follow that a sequence of phase transitions ought to exist interpolating between regular lattices and scale-free systems. An attempt to reproduce a scale-free model with preferential rewiring on a regular lattice combining the Watts-Strogatz and Barabási-Albert models fails because, although the more connected nodes continue to become more connected, the overall acquisition of nodes increases uniformly across the system (Kawachi et al, 2004). Thus there are no corresponding sparsely connected nodes: In the construction of a scale-free model the rich get richer but the poor must also become relatively poorer. Thus a procedure which showed a continuous interpolation between regular lattices, small world networks, random networks and scale-free systems would be of benefit to the Observer System in deploying appropriate monitoring probes and assigning various properties to different levels or sectors of the system based on its derived topological structure. A fixed cardinal structure transition is defined (Kawachi et al, 2004), where:

1. Each node *i* is selected in turn together with an edge that connects it to a second node *j*.
2. With probability *p* the link *i-j* is deleted and the highest degree node of *i* and *j* is rewired to connect to a third node *l*: If $k_i > k_j$ then *i* connects to *l* otherwise when $k_j > k_i$ then *j* connects to *l*.
3. Node *l* is chosen according to the probability: $\Pi(k_l) = \frac{k_l + 1}{\sum_m (k_m + 1)}$
4. Repeat procedure for every link of every node then repeat.

It is observed that $p=0$ gives the regular lattice. $p=10^{-2}$ gives a small world type network with scaling around the mean degree, a large clustering coefficient and small path lengths between nodes. $p=10^{-0.5}$ gives a random network with scaling around the mean degree, low clustering and small path lengths between nodes. Finally $p=1$ reduces to the

Barabási-Albert model with power law degree distribution but low clustering and small path lengths between nodes. This is illustrated in Figure 6.4.

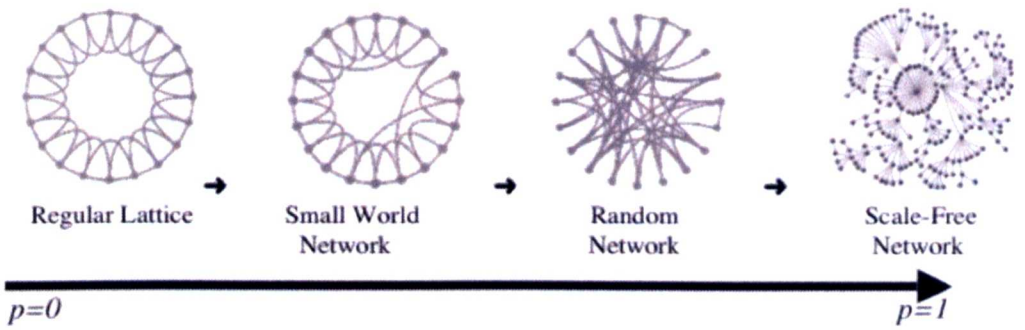

Figure 6.4: A structural phase transition

Thus it is possible to account for most types of behaviour within large-scale complex systems by the adjustment of parameters and interpolation is possible from a regular lattice, by nature displaying a high clustering with long path lengths to a small world, with high clustering and short path lengths to a random network, with low clustering and short path lengths to either a scale-free (Barabási-Albert) system with low clustering and short path lengths or a scale-free (Klemm- Eguíluz) system with high clustering and short path lengths. The recognition and topology determination is thus of major interest for Observer System deliberation. The algorithms influencing connectivity are summarised in Table 6.2.

Table 6.2: Comparison of network construction algorithms

	Regular	Small World	Random	Scale-Free	Clustering	Distribution
Barabási-Albert	✘	✘	✘	✔	Low	Power Law
Barabási-Albert With Fitness Functions	✘	✔/✘ *depending on function*	✘	✔	Varies with function: Generally High	Power Law
Watts-Strogatz	✔	✔	✔	✘	High	Normal
Klemm-Equíluz	✘	✔	✘	✔	High	Power Law
Kawachi et al	✔	✔	✔	✔	High	Normal/ Power Law

6.2.1 The Hub Connection Density Measure

It has been noted, through this chapter, that one of the most noticeable features of scale-free topology is the formation of high degree hub nodes. In view of the defining role of hubs and hub-to-hub connectivity in the exhibition of the "robust-yet yet-fragile" nature of self-organisation in scale-free systems, the most obvious measure to assess whether a system is exhibiting scale-free organisation is to gauge the extent to which hubs are forming and connecting in the system. High hub density can be characterised as high degree nodes (or hubs) connecting to other high degree nodes. The rearrangement inequality (Wu and Liu, 1995) states that:

If $a_1 \geq a_2 \geq \ldots \geq a_n$ and $b_1 \geq b_2 \geq \ldots \geq b_n$ then for any permutation $(a_1', a_2' \ldots a_n')$ of $(a_1, a_2, \ldots a_n)$

$$a_1b_1+a_2b_2+\ldots a_nb_n \geq a_1'b_1+a_2'b_2+\ldots+a_n'b_n \geq a_nb_1+\ldots\ldots\ldots\ldots+a_1b_n$$

Thus if the product of connected node degrees is assessed then a measure will be maximised when high degree nodes are connected to other high degree nodes (Li et al, 2005). So if the set of links for the system is C, meaning that if node i is connected to node j then (i,j)∈C for any nodes i and j, then $\sum_{(i,j)\in C} k_i k_j$ is maximised when high degree nodes are connected to high degree nodes. The hub connection density is defined for a graph consisting of a set of vertices, V, with |V| = N and a set of edges E linking the vertices, as: $\frac{1}{N}\sum_{(i,j)\in E} k_i k_j$ for any vertices i and j with degrees k_i and k_j respectively. This then gives a notion of hub connectedness across the nodes in a system modelled as its analogous graph. In a similar way to (Li at al, 2005) this measure attempts to quantify the nature of hubs in the system that is mathematically rigorous and allows the detection of scale-free organisation. Furthermore, in the next chapter, the measure will be empirically evaluated to provide the limits of phase transitions from regular lattice to small world network through random network to scale free system. Thus giving a measure to assess the actual topology of the systems evolution and so inform the Observer System of the most appropriate monitoring for the system and its components.

6.3 Acquaintance Monitoring

To adequately monitor a large-scale complex system, for autonomic response, for instance, a large fraction of the nodes have to be observed to form a reasonable knowledge of the global system's state. In scale-free systems, however, system state

awareness is centred at the hubs, which make up a small proportion of the whole network. The distributed data regarding sensing results, from within the complex system, including inter-node monitoring tends to accumulate at these hubs. Thus extracting the most relevant data for system self-governance is dependent on the detection and identification of hubs to the Observers System modules. In practice the identification of hubs requires global information, so to form a representative sample of the data held at the computing nodes involves using random walk algorithm methods, selecting those nodes with the highest degree and monitoring these nodes. However this approach gives no assurance of successfully detecting the major hubs and is subject to local optimisation.

In this work the acquaintance immunisation strategy (Cohen et al, 2003) is adapted to randomly traverse the graph towards the hubs along their acquaintance arcs, for a quantifiable fraction of randomly chosen hubs. This is mathematically and experimentally proven (later in this chapter and the next) to give a high accuracy hub detection method for a monitoring selection algorithm based on a reduced set restricted to the system knowledge hubs. It is widely known that to be successful in protecting a network by randomly immunising members requires a very large proportion to be immunised and so arrest epidemic conditions. Percolation on scale-free networks, where the percolation threshold tends to 1, shows a large proportion of the nodes have to be removed before the network is compromised (Cohen et al, 2003). In a similar way a large proportion of the nodes have to be immunised otherwise the network remains contagious. Knowledge propagation over the system can be thought of in a similar manner. In order to maintain a reasonable knowledge of the system state a large proportion of the nodes need to be monitored.

Considering the "robust yet fragile" nature of the scale-free networks; the fragile refers to the fact that the most highly connected nodes or hubs in the network can be targeted for attack and their removal can compromise the network. Thus the best possible course of action for observation ought to be targeting the hubs for monitoring. In this way any threat to the hub structure can be dealt with at the earliest possible detection, but also the majority of the system knowledge is contained through the hub structure of the system. In effect the hub connectivity system forms a self-organising network overlay. This type of topology is observed through systems of peer-to-peer networks based on JXTA (Traversat et al, 2002). In JXTA each hub maintains a list of all the other known hubs in the system. At periodic time intervals the hubs pass this list to all the other identified

hubs. Over time it is expected that all the hubs in the system will be discovered. The identification of hubs, however, is a problem, for large complex networks, as previously stated, because information regarding the degree of every node in the network is not necessarily available. That is there is a lack of global data. Schelling's model (Schelling, 1971) has been used in a pure peer-to-peer setting to create an adaptive overlay network (Singh and Haahr, 2006). This approach, whilst useful in a peer-to-peer system, assumes a static homogeneous system of nodes and may still precipitate local optimisation, due to unknown node data and the lack of global information, resulting in network partition.

Acquaintance Monitoring Selection (AMS) removes the need for global data. For AMS, at any point in the systems evolution, a random proportion r of the N nodes is chosen and a random acquaintance that they are connected to is searched for. The acquaintances rather than the original nodes are monitored. The nature of scale-free networks means that in most cases the nodes chosen for monitoring will be of higher degree than the node that nominated them. The fraction r may be more than 1 as a node may be queried more than once. However the fraction monitored, f, will be always less than or equal to 1. So to monitor the network it is necessary to calculate the critical fractions for the network. That is the critical fraction of the nodes that need to be chosen r_c and the resultant critical proportion that are monitored to gain all the knowledge in the system located through the virtual hub overlay f_c can be calculated. Thus for networks displaying the characteristics of scale-free topology a simple value can be calculated and the algorithm invoked to effectively monitor the whole network by monitoring only the key nodes.

The probability that a node with k connections in a system of N nodes is selected for monitoring is:

$$\frac{kP(k)}{N\sum_{n} nP(n)}$$

where $P(k)$ is the power law distribution for connectivity. This ratio quantifies what is observable within scale-free scenarios: randomly chosen acquaintances possess more links than randomly selected nodes. This is extremely useful in statistically obtaining a high assurance that tends system hubs to be identified.

To analyse and determine the methods required producing the critical fractions it is necessary to look at the nodes that randomly selected nodes connect to. That is following

a branch starting from a random link, on a spanning cluster, there will be a level, l, where there will be $n_l(k)$ nodes of degree k. At the next level $l+1$ each of these nodes has $k-1$ neighbours (excluding the nominating acquaintance node). Let the event that a node with degree k remains unmonitored be R_k. To establish the number of nodes of degree k' that are unmonitored at level $l+1$, $n_{l+1}(k')$; the number of links emanating from the lth layer, is multiplied by the probability that a node of degree k' is nominated by an unmonitored node at level l: $p(k' | k, R_k)$. Then it is necessary to multiply this by the probability that this node is unmonitored given this nodes and its neighbours degrees and the fact that the neighbour is unmonitored: $p(R_{k'} | k', k, R_k)$. So:

$$n_{l+1}(k') = \sum_d n_l(k)(k-1)p(k'|k,R_k)p(R_{k'}|k',k,R_k) \qquad [1]$$

and using Bayes Theorem:

$$p(k'|k,R_k) = \frac{p(R_k|k,k')p(k'|k)}{p(R_k|k)}$$

At each time step a random site (with degree k, say) is chosen with probability $1/N$. The probability of being reassigned to a specific acquaintance is $1/k$. So the probability that an acquaintance is not selected at one attempt, and thus remains unmonitored, is: $1-1/Nk$. Now the number of nodes multiplied by the fraction of nodes that observation is being attempted for gives the number of monitoring attempts: rN. Thus for all attempts the probability that a node is not chosen is given by:

$$v_r(k) = (1-\frac{1}{Nk})^{rN}$$

This quantity is approximates to $e^{\frac{-r}{k}}$

If the neighbour's degree is not known then the probability that an acquaintance is not chosen is the average of the v_r taken over k. It is also apparent if a neighbour of known degree is monitored this does not provide any additional data about a nodes probability of observation. Thus

$$p(R_k|k,k') = p(R_k|k,k',R_{d'})$$

$$p(k'|k,R_k) = \frac{Np(k'|k)e^{\frac{-r}{k'}}}{\sum e^{\frac{-r}{k}}}$$

Substituting in [1] gives

$$n_{l+1}(k') = v_r^{k'-2} p(k'|k) e^{-r/k'} \sum_k n_l(k)(k-1)e^{-r/k}$$

Since the summation does not depend on k' then $n_l(k)$ is some constant factor of $n_{l+1}(k')$, so that

$$n_{l+1}(k') = n_l(k') \sum_k p(k|k')(k-1)v_r^{k-2} e^{-2r/k}$$

This then gives a critical phase measure: If the sum is larger than 1 the branching will continue indefinitely. If it is smaller than 1 a monitored node is encountered. An expression for the critical proportion of nodes needed to be selected, r_c is given by:

$$\sum_k \frac{P(k)k(k-1)}{k} v_{r_c}^{k-2} e^{-2r_c/k} = 1$$

The fraction of monitored nodes can be easily obtained from this expression. These results show that strategies such as random walk for Internet type networks (exponent in power law connectivity is between 2 and 3.5) selection require f_c=1, whereas using the acquaintance monitoring algorithm requires a monitoring threshold of $f_c \approx 0.25$ (Cohen et al, 2003) This proves the value of this approach for when applied, as a monitoring strategy, to a system exhibiting self-organisation through scale-free connectivity.

6.4 Monitoring Strategies and Memory Management

The Signal Grounding problem was investigated in this work and possible ways around it proposed earlier in this chapter. The solution, however, required the Observer System to maintain a large number of action histories in order to achieve the grounding of some system signals. Following the investigation of Acquaintance Immunisation and monitoring on scale-free networks it may be possible to trim the situation space via the application of these principles to a Situation (action history) space. To attempt to achieve such a reduction in required memory space for action histories the representation of the situation space is required to conform to scale-free principles. Thus it is proposed that within the situation space two nodes are linked if the sense-action-sense procedure leads

to the expected or grounded response for a particular fluent. In the terminology and formalism used previously: Two situations (action histories) S_i and S_j are connected if there is some fluent F (assumed to be a relational fluent, for now) that changes value in situations S_{i+1} and S_{j+1} for actions a_1 and a_2 respectively. That is there is grounding for F. So there is a link between two situations if:

$knows(\neg F,S_i)$ and $knows(F,do(a_1, S_i))$ and

$knows(\neg F,S_j)$ and $knows(F,do(a_2, S_j))$ deduced from

$do(a,do(a_1,do(a,s)))$ with $SR(a,s) \neq SR(a,do(a_1,do(a,s)))$

$do(a,do(a_2,do(a,s)))$ with $SR(a,s) \neq SR(a,do(a_2,do(a,s)))$

where $a=sense_F$

In this way situations are linked according to preferential attachment, with the number of connections the fluent grounding already possesses determining the likelihood of further links with fitness varying according to the success or failure of the actions in changing the fluent values. Additionally some situations will naturally drop out as the fitness for grounding a fluent diminishes through the learning procedure previously outlined. In this case the links to that "lost" situation will be transferred, via the existing mechanism, to another situation. This means that different situations (action histories) are linked via a signal having the same ontological relevance to those situations. So connected actions histories, although seemingly different, can be thought of as representing equivalence between situations with a scale-free connectivity based on preferential attachment.

6.5 Summary

At each application level in a system an observer may be placed to provide cognitive monitoring for use at that location and throughout the wider system. To this end three topics have been identified and addressed as highly relevant to the operation of large scale complex systems under the influence of a deliberative cognitive observer system:

a) The establishment of the conditions that resulted in new unforeseen emergent behaviour (Signal Grounding).

b) The engineering of emergence: Reproducing a known emergent outcome through the programming of the low-level component interactions.

c) The provision of metrics and measures that promote or assess the emergence of a wide class of behaviour observed to occur in large-scale complex systems, scale-free connectivity.

Thus the previous chapter progressed with a treatment of signal grounding based on deliberation over action histories in the Situation Calculus. The subsequent grounding could then be assessed by future operation and maintained or dropped from the cognitive systems' consideration accordingly. The engineering of emergence was assessed through a previously defined known model of the emergent behaviour resulting from food foraging observed in many natural domains such as ant colonies. The behaviour was assessed by means of a Stochastic Situation Calculus representation of the domain as a Markov Decision Problem. Through this model the Observer System would be able to analyse and influence the engineering of very particular emergent outcomes through the adjustment of the domain parameters.

Now, in this chapter, the emergence and prevalence of Scale-Free systems is described. The power law distribution for connectivity has been observed to occur in very many large-scale manmade and natural complex systems from the Internet to solar flare patterns. In order for the Observer System to utilise this property it is necessary to understand the phase transitions that may occur in the evolution of these systems. The particular observed occurrences involved power law connectivity with high clustering. High clustering was shown to be a feature of small worlds that reduced to random graphs on sufficient applications of the small world algorithm on a regular lattice system. Preferential attachment, growth and fitness were identified as key features in promoting scale-free behaviour as observed in large-scale complex systems. In order to assess when a system was behaving to scale-free principles the hub connection density measure was proposed based on the connectivity of the system hubs. Additionally a light monitoring strategy especially suited to scale-free networks was proposed based on acquaintance immunisation. The chapter concluded by extending this strategy to deal with the explosion in the number of action histories requiring to be stored in the accomplishment of the signal grounding techniques outlined earlier. These measures and properties of scale-free and large –scale systems will be evaluated experimentally in the next chapter.

Chapter 7

Executing the Specification

The previous chapters have presented a theoretical and deliberative framework for the representation and analysis of distributed computing systems of any scale. The ubiquity of the (Stochastic) Situation Calculus has been shown to

a) Allow the specification of top-down approaches for representing normative positions for centralised control.

b) Permit the deliberation necessary for bottom-up functionality to be utilised in three ways:

   - In the detection of some new emergent features (through signal grounding).
   - To engineer particular instances of previously realised emergence (through parametric adjustment)
   - To detect various instances of a particular generic network topology with associated metrics that enable the harnessing of the systems complexity providing, for instance, more efficient monitoring strategies.

This chapter presents example implementations arising out of the (Stochastic) Situation Calculus Specification. It commences by describing the implementation of a system based on an extensible BDI used by a centralised system controller in autonomic middleware, using a single observer. Then it continues, after a brief introduction to the Netlogo (Wilensky, 2007) simulation environment, by describing the topologies of large complex systems through an analysis of the types of system structures that occur and an implementation of the algorithms for the interaction dynamics presented in the preceding chapter. This involves implementing the algorithms to produce and compare the global topological structure and properties of these systems. The simulation continues with an implementation and evaluation of Acquaintance Monitoring Selection procedure. Finally the knowledge acquired through one application of the AMS is assessed for a variety of systems. Additionally, in Appendix 2, a simple scenario is implemented to illustrate the grounding of a particular global outcome. This also facilitates the recording of action

histories to permit the associated trimming of the situation space that maintains the amount of system knowledge available to the observer whilst removing duplicated and redundant data without global information regarding the data content, location or relevance.

7.1 A Normative Autonomic Scenario

This section describes the implementation of a model autonomic control service based in the middleware. The control service incorporates three core services, embedded in a three-layered model comprising: the service manager, the distributed shared system tuple space service and the system controller incorporating an Observation System. The architecture is based on a control service model that continuously monitors the specified service for non-ideal behaviour, to identify conflicts and errors, prescribing repair plans and performing reconfiguration.

The system monitor, the system reconfiguration module and the system repair strategies consisting of resolution actions determining when, where and how the repair is applied, form the major part of the meta-system: It is here that the resolution strategies (intentions) are chosen based on the deliberation of the component over its role, responsibility, reward and regulatory strictures through an Extensible BDI based on beliefs, roles, the situated intentions, the normative intentions and the reward intentions:

- Beliefs correspond to service information derived from a range of sources, including domain, environment or the communicated beliefs of components via the shared space.
- Roles represent the state of affairs in an ideal world that often maximize the service's own goals in terms of fulfilling a desire. By comparing the system belief set (observed system states) against its stated role, the system may detect a mismatch and instantiate a set of intentions. So the system high-level desires are propagated throughout the system to lower level management systems in a form that sets desires local to the services. For instance the high level goal of availability manifests itself in the goal of setting a repair strategy for services the agent believes to be damaged. Component roles provide action triggers to specify the requisite resources.
- Situated intentions represent action sets for the system to undertake in a given situation to achieve its specified desires and/or to address the mismatch

between the system environment (beliefs) and the system's desires (goals) including acting as a resource for other components and maintaining system norms.

- Normative intention represents a set of actions to be undertaken to ensure a specified set of rules and regulations, including obligation and responsibility, rules are observed, via a dynamic utility representation, before a given intention is enacted.
- Reward intention represents a set of system actions to optimize goal-oriented intentions such as minimizing costs or optimizing quality of service.

For this implementation the beliefs, desires, goals and intentions can be described as being collections of constraints, each of which represents distinct pieces of beliefs, desire or goal and so on. These constraints are generated using service beliefs and desires. In this implementation, beliefs are a runtime service's states such, as a service is available for client requests or not. Intentions are the system actions (execution), which are, for instance, triggered because of a mismatch between the system beliefs and system desires sets using the norms. Thus service states can be deliberated upon and strategic management executions propagated throughout the system.

The deliberation procedures required for intention resolution is formally specified in the Situation Calculus, to illuminate the EBDI processes. For instance, a report of an unavailable service in the system space triggers a situation whereby the role of service reconnect is activated in the system controller: When the system detects a failure to connect to a service it automatically retries as a responsibility to the connecting component/agent. On failing a predetermined number of times it then attempts to connect to an alternative service and/or starts a diagnostic process assembled from its available resources resulting in a repair strategy committed intention. The Situation Calculus representation of the entire EBDI deliberation process, for a session id, can be specified, for instance:

$$retrial(id, do(a,s)) \Leftrightarrow retrial(id, s) \wedge \neg\exists t(a=read(space) \rightarrow decision(id, t)) \wedge \neg retried(id,s) \vee a=retry(id)$$

with $poss(retry(id), s) \Rightarrow \neg available(service, s) \wedge \neg retried(id,s)$

giving rise to additional successor state axioms:

$retried(id, do(a,s)) \Leftrightarrow retried(id, s) \vee [(a=read(space) \rightarrow \neg available(service, s)) \wedge retrial(id,s)]$

$available(service, do(a,s)) \Leftrightarrow available(service,s) \wedge \neg (a=read(space) \rightarrow \neg connected(service,s))$

$connected(service, do(a,s)) \Leftrightarrow (connected(service,s) \wedge a \neq f\_connect(service)) \vee a= s\_connect(service)$

with $poss(f\_connect(service),s) \Rightarrow remote\_service\_exception(s)$

$poss(s\_connect(service),s) \Rightarrow request(id, s)$

If the system remains unavailable after retrying a specified number of times then a diagnostic state would be entered leading to repair strategies based in the implemented scripts.

$diagnosing(service, do(a,s)) \Leftrightarrow [diagnosing(service,s) \wedge \neg \exists fault(a= service(fault))] \vee (a=findfault(service))$

with $poss(findfault(service),s) \Rightarrow \exists id\ (retried(id,s)) \wedge \neg connected(service,s)$

leading to

$repairing(service, do(a,s)) \Leftrightarrow [repairing(service,s) \wedge \neg \exists r(a=repaired(service)] \vee (a=repair(service))$

with $poss(repaired(service),s) \Rightarrow available(service,s)$

$poss(repair(service),s) \Rightarrow diagnosed(service,fault,s)$

Similar representations of self-governance can be derived to give a complete specification and used to specify an implementation for this autonomic control feature: Although full details of the implementation are outside the scope of this thesis, the specification can be coded and executed, further details are available in the given references. To summarise, the implementation is executed through a Cloud architecture (Miseldine and Taleb-Bendiab, 2005a). It is defined to enable the autonomic framework to function using a developed language, Neptune (Miseldine and Taleb-Bendiab, 2005a), which allows management objects to be compiled and inspected at runtime. The cloud can be thought of as a federation of services (component agents) and resources controlled by the system controller and discovered through the system space. The system space provides persistent data storage for service registration and state information giving the means to coordinate

the application service activities. Neptune exposes policies and decision models for system governance, derived from the Situation Calculus/EBDI model, as compiled objects that can be inspected, modified and executed at runtime. Thus the system can evolve as modelled by the logical specification in a safe and predictable manner giving the adjustable self-management required. Neptune objects are executed on demand through an event model exposed by the clouds architecture.

The system controller with an associated Observation System controls access to and from the individual services and resources within the cloud. It brokers requests to the system based on system status and governance rules, in Neptune objects, derived from the EBDI deliberative process as stated above, either as an abstraction that inspects calls between the System Space and services, as in Figure 7.1, or as a monitor that analyses the state information stored within the system space as in Figure 7.2.

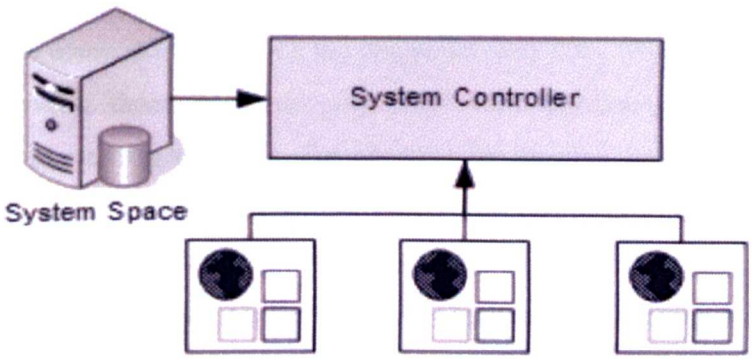

Figure 7.1: System controller managing system space

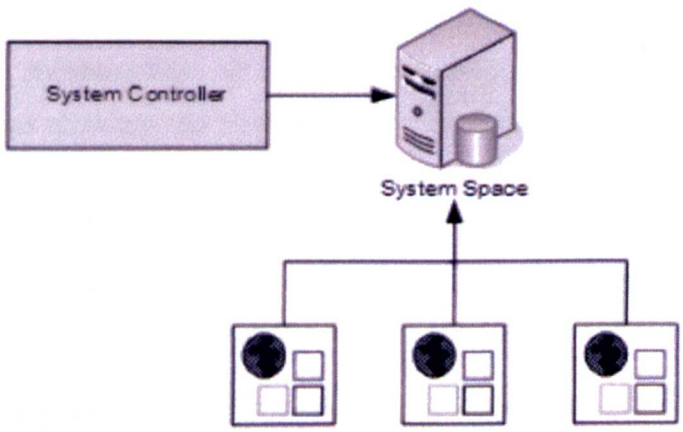

Figure 7.2: System controller as system space monitor

Each service and resource when it first registers itself to the Cloud sends a meta-object serialized from an XML definition file. This meta-object contains the properties and state data of the service it is describing and is stored within the System Space at registration. Each service maintains its own meta-object and updates the System Space when changes in state occur. The XML definition file contains all information required for the Cloud to discover the service through registration contained in the service element. This allows the querying of the service status through the published state properties contained within the state element.

In addition to the meta-objects exposing properties of a service within the Cloud, they also describe events that can be fired, caught and handled. The event model begins by the service informing the System Controller when an event is fired, which itself marshals this event to the System Space to provide the appropriate scope. Other systems within the Cloud that have a role assigned to an event instruct the controller to inform them via message when an event is fired, which the controller duly does when the event message is received. The events themselves can pass XML files containing specific information about the event at the time of firing. For example the event occurrence, when fired, can include an XML document containing the actual data that is new to the service. This is passed along with the event message and is exposed to all agents who have declared an event handler with the controller. It should be noted however, that the event model is abstracted from the agents within the system, and is controlled by the Neptune scripting language that sends and receives the appropriate event calls to the controller.

The Neptune scripting language is structured in terms of rules, conditional statements and variable assignments that are translated from the normative specification to software system objects, encapsulating all the logical inference processes and variable instantiations. Thus allowing the Neptune object to be inspected, modified, recompiled and re-evaluated at runtime, through the specification. In this way the base rules for deliberation in the EBDI to control the cloud architecture have been transcribed, from the Situation Calculus reasoned representation, into Neptune objects that can be modified as a result of Observation System deliberation on system events.

For example in the given Situation Calculus representation, an availability rule is defined together with resolution strategies in the case of a service being unavailable. This specifies that if the service is not available for calling, then the Cloud status is queried to see if a service instance alternative is available. If no instance is found, the repair strategy

is service regeneration. Calls are then rerouted to the newly generated service, or the alternative service instance if located. This is shown in Figure 7.3:

```
define service s
if (service.availableServices.likeMe.count = 0)
    service.s = regenerate(me,machineID)
else
   service.s = services.availableServices.likeMe[0]
end if
rerouteCalls(s)
```

Figure 7.3. Neptune repair strategy script

Thus a set of normative positions can be specified for the autonomic control system giving self-governance, as required, in smaller scale components or systems where centralized control, most likely involving only one observer hierarchical level, is appropriate. The deliberative features necessary for decentralized, distributed and larger-scale systems, where mission-oriented adaptation may be called for, requires a extensive cognitive Observer System to address the issues mentioned in point b) at the start of this chapter. The simulations to assess the self-governance specification for the topologies of large complex systems, through an analysis of the types of system structures that occur, and an implementation of the algorithms for the interaction dynamics presented in the preceding chapter are accomplished through Netlogo. Accordingly a brief overview of Netlogo follows.

7.2 The Netlogo Execution Environment

Netlogo 4.0beta1 is a multi-agent programming language and integrated modelling environment (Wilensky, 2007) that extends previous versions of the language. It is based on the Logo language (Feurzeig and Papert, 1968), a dialect of LISP. The Net prefix denotes its applicability to decentralised, interconnected network-like phenomena. It is particularly well suited to modelling large-scale complex systems that develop and evolve with time. It is possible to program instructions for very large numbers of independent components or agents acting concurrently, making possible the investigation of the global outcome that results from component interactions: The detection of emergent behaviour. Additionally the core structure as a functional programming language gives a natural

implementation from the mathematical logic of Situation Calculus to an executable program.

It is principally for these reasons that Netlogo will be used for the execution and implementation of the systems already discussed in this work. For example a regular lattice of adjustable size and connectivity may be constructed from the code fragment illustrated in Figure 7.4, with output as shown in Figure 7.5 when the user defined values of 40 nodes with a connectivity of 25 are used. An output graph is also included, on-screen, showing the distribution of the nodes connectivity. In the case of the regular lattice in this example, the graph shows, as expected, that all 40 nodes have the same connectivity of 25. If, however, a button is coded to rewire the lattice as in the Watts-Strogatz (Watts and Strogatz, 1998) model of the previous chapter then the output is as shown in Figure 7.5: The node degree distribution can be observed to take the typical form of a normal distribution.

```
to golat
ca
set-default-shape nodes "computer workstation"
set-default-shape edges "line"
create-nodes No.Nodes
  [
    set color red
    set neighbor-nodes []
    layout-circle sort nodes 50
  ]
  let p 0
  while [p < count nodes]
  [let first-node node (p mod count nodes)
  let k 0
  while [k + 1  < (Lattice-Connectivity / 2)]
   [let second-node node ((p + k) mod count nodes)
   make-edge first-node second-node
   set k k + 1]
   set p p + 1]
do-plotting
end
```

Figure 7.4: Netlogo code fragment to construct a regular lattice

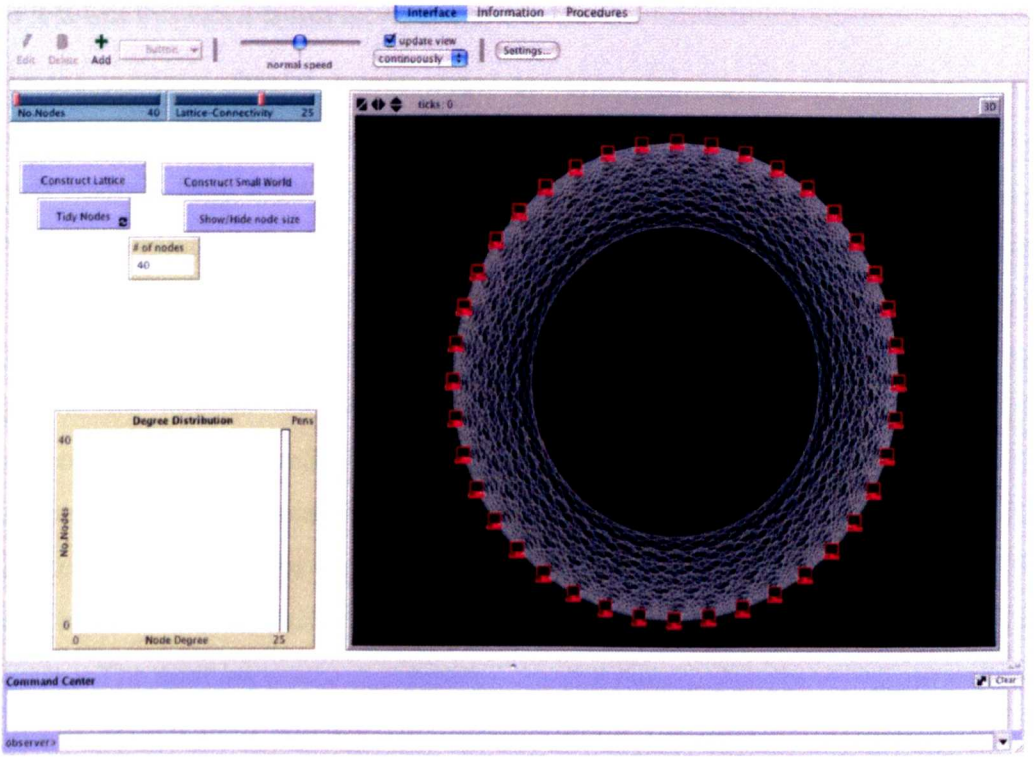

Figure 7.5: Netlogo output: Regular lattice of 40 nodes and connectivity of 25.

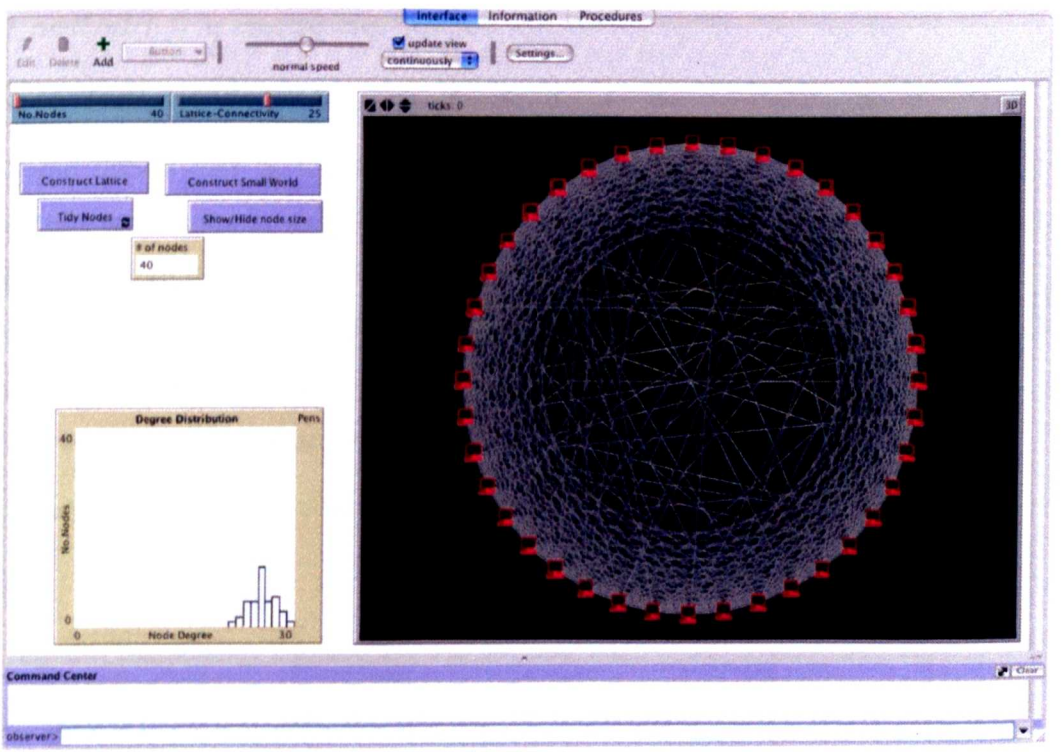

Figure 7.6: A regular lattice rewired to a small world model

7.3 Scale-Free and Random Systems

In the previous chapter it was noted that the majority of real world large, complex systems have been observed to exhibit forms of organisation, in their global topologies that emerges from some low level component behaviour. Previously random graphs were the main representational means for large network systems, where nodes connected with uniform probability across the network. The Barabási-Albert construction algorithm introduced the notion of preferential attachment to partially explain the appearance of power law tails in the nodes connectivity. In order to appreciate the difference in outcomes between random connectivity and preferential attachment versions of the models were implemented through Netlogo from the Situation Calculus propositional approach. Thus, for the propositional account, from the observer's perspective, there are two actions *addNode* and *addLink(i, j)* meaning make a connection between node i and node j. The *addLink* action is a stochastic action, in that the action may succeed or fail so can be decomposed into two deterministic actions: *s_addLink(i, j)* and *f_addLink(i, j)* (meaning the action succeeds or fails respectively). The action *addLink* is assumed to be under some cognitive systems autonomous control. If the system elects to perform the action then non-determinism arises and exactly one of either *s_addLink(i, j)* or *f_addLink(i, j)* is enacted with associated probabilities. So more formally:

$choice(addLink(i,j),a) \equiv a=s\_addLink(i,j) \vee a=f\_addLink(i,j)$

Some fluents may be introduced: *nodeNumber(s)* is the total number of nodes in the system at situation s, *requestConnection(i, j, s)* means the autonomous cognitive system has requested that a link be established between node i and node j and *connected(i, j, s)* means node i is connected to node j in situation s. Note this ought to be a directed relationship in a complex network so that node j is not necessarily connected to node i. The successor state axioms are:

$nodeNumber(do(a,s))=N \Leftrightarrow (nodeNumber(s)=N \wedge a \neq addNode) \vee$

$(nodeNumber(s)=N-1 \wedge a=addNode)$

$requestConnection(i,j,do(a,s)) \Leftrightarrow requestConnection(i,j,s) \wedge a \neq s\_addLink(i,j)$

$connected(i,j,do(a,s)) \Leftrightarrow connected(i,j,s) \vee a=s\_addLink(i,j)$

It is not known, in the nondeterministic setting, which of the choices is performed but the probabilities for node degree are known in the situation. It is necessary to align the information in the axioms with the probabilistic information. So, as previously let

$prob(a,b,s)=p \equiv choice(b, a) \wedge poss(a,s) \wedge p=prob_0(a,b,s) \vee$

$[\neg choice(b,a) \vee \neg poss(a,s)] \wedge p=0.$

Here $prob_0(a, b,s)$ will be a specification of the probability that a is selected in situation s as the outcome of stochastic action b given that a is one of the nondeterministic choices for b and that a is possible to enact in s. So the probability for a new node j to connect to an existing node i can be stated for:

a) Barabási-Albert preferential attachment

$$prob_0(s\_addLink(i,j),addLink(i,j),s) = \frac{d_i}{\sum_r d_r}$$

$$prob_0(f\_addLink(i,j),addLink(i,j),s) = 1-\frac{d_i}{\sum_r d_r}$$

b) Random connectivity

$$prob_0(s\_addLink(i,j),addLink(i,j),s) = 0.5$$

$$prob_0(f\_addLink(i,j),addLink(i,j),s) = 0.5$$

An small anomaly is apparent here in that the probabilities for s_addLink(i,j) and f_addLink(i,j) ought always to sum to 1. So for this example it must be assumed that at least one of the choices is always possible.

The Netlogo implementation produced includes:

- The option to construct a generic scale-free or random connectivity system consisting of a user defined number of nodes.
- A visual output screen to enable the identification of the various visible topological properties of random and scale-free connectivity
- A graphical representation of the resulting node degrees, showing the degree distribution.
- A "show/hide node size" button giving the option to scale the visible size of the node to its the degree.

- A "Tidy Nodes" button implementing the arrangement of the nodes into a more easily observable format using the Fruchterman-Reingold layout algorithm (Fruchterman and Reingold, 1991); adapted from the Netlogo models library (Wilensky, 2005).

Figure 7.7 shows a scale-free system constructed from 1012 nodes, whilst Figure 7.8 shows the same number of nodes connected according to random principles. In both cases the node's size represents its relative degree. The system of connected hubs is readily observable in the scale-free system, whilst most nodes in the random system have about the same degree. This arises solely from the code that is used to generate the systems from the interaction of nodes in finding connections. The connections are generated as specified through the Situation Calculus specification and coded in Netlogo as shown in Figure 7.9. The code fragment is from this system's observer's perspective. The portion of the Observer system implemented is just that which deals with the monitoring, influence and deliberation over this part of the system. Each node depicted may also possess its own observation module that interacts with this level in the Observation system as described in the previous chapter. Likewise the observations made at this part of the system may also feed up the Observation hierarchy to inform an observer module higher-up in the system, as described. Thus, for examples, the *ask nodes* forms a norm based command delivered to the nodes via the Observer System, whilst the *report connection* is a monitoring function served by the tuplespace, where the data for Observer System deliberation is held. Additionally, as output from the model, the degree distributions, shown to the left of the visualisations in Figures 7.7 and 7.8 show very marked differences in the connectivity of the two networks illustrated. Most nodes in the random system have a degree close to the mean degree around 1 with the highest degree nodes having 10 or 11 connections, consistent with the more rapid decrease away from the mean observed in a normal distribution. In contrast the scale-free system degree distribution has a much slower decay rate, indicative of a power law tail, with the small number of highest degree nodes having upwards of 90 connections.

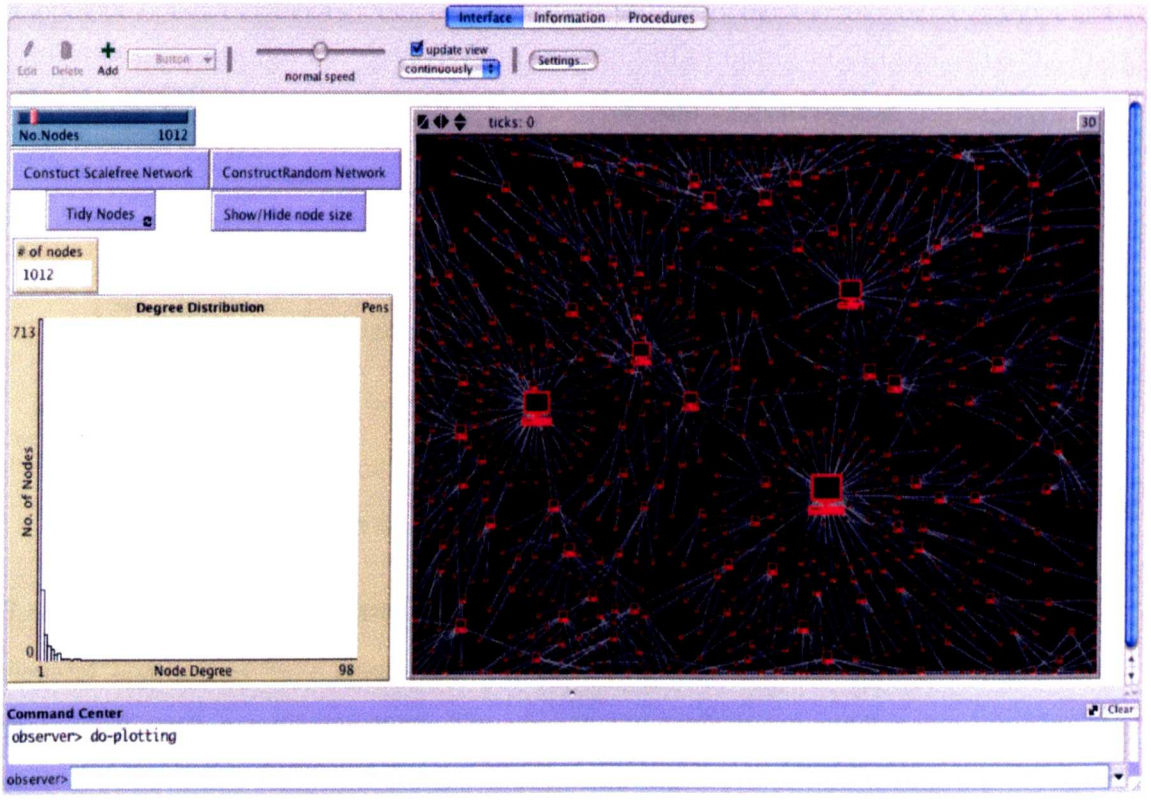

Figure 7.7: Netlogo implementation showing scale-free system topology

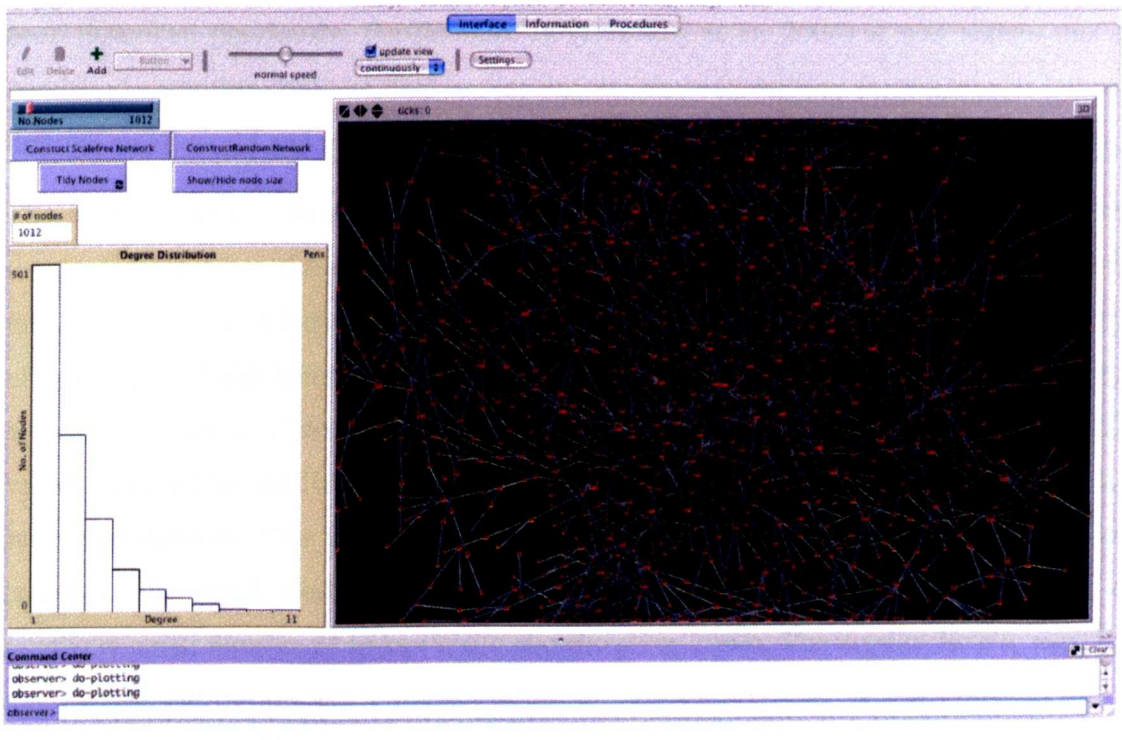

Figure 7.8: Netlogo implementation showing random system topology

```
to-report SFconnection
  let prob sum [length neighbor-nodes] of nodes
  let connection nobody
  ask nodes
  [
    if connection = nobody
    [
      ifelse length neighbor-nodes  > 0.5 * prob
        [ set connection self ]
        [ set prob prob - (length neighbor-nodes)
    ]
  ]
  report connection
end
```

```
to-report randconnection
  let connection nobody
  ask nodes
  [
    if connection = nobody
     [ ifelse random 2 = 1
      [set connection self]
      [set connection one-of nodes]
    ]
  ]
  report connection
end
```

Figure 7.9: Netlogo code to give a scale-free (left) or a random (right) network.

7.4 The Hub Connection Density

Several approaches have been taken for analysing the network structures that emerge from large-scale complex systems. The growth and preferential attachment algorithm for scale-free networks (Barabási-Albert, 1999), the small world model (Watts and Strogatz, 1998), a transition between a regular lattice, where each node is uniformly connected to each of a fixed number of its neighbours arranged in a circle, and the network that results from randomly rewiring some of the links to connect diametrically opposed nodes have been described previously. Furthermore in (Kawachi et al, 2004) it was shown that a combined algorithm could be used to demonstrate phase transitions from a regular lattice, through small world and random networks, to a scale-free network.

The Hub Connection Density measure, introduced in the previous chapter, reflects the topological changes these phase transitions exhibit. In a regular circular lattice the measure is fixed at a local maximum for the metric based on the maximum connectivity of the nodes. If the connectivity of each node is k then the Hub Connection Density measure is fixed at k^3 no matter what size the network. Figure 7.10 shows the typical values obtained for the Hub Connection Density in a regular lattice, random network and scale-free network. The values for a Small–World scenario are similar to the regular lattice as the small world-world is obtained from the regular lattice by rewiring a relatively small number of links (Watts and Strogatz, 1998). The regular lattice, used in this experiment, represents a regular network where each node is connected to its 6 nearest neighbours. Figure 7.10 shows the value of the Hub Connection Density for random systems to be constant around 18, no matter what size of system is under

consideration. As the self-organisation exhibited by the scale-free topology emerges the measure rises. If the number of connections at each node are bounded (by k_{amx} say) then the measure will approach $k_{max}{}^3$. Figure 7.10 shows the Hub Connection Density, for scale-free systems, increases slowly as the system size increases. This is to be expected because the number of hubs and hub-hub connections inevitably increases with system size. These experiments assumed unlimited connections per node. The values obtained are based on averaging the values from a number of system construction simulations of the sizes indicated. For all sizes of systems the figure obtained for the random systems was always very close to 17 or 18. The values obtained for Scale-free systems, however, vary by as much as a 30% from the values shown in Figure 7.10. Again this is to be expected, as there are many ways to wire a scale-free system consisting of the same number of nodes. The main result is that the values for scale-free systems are always significantly higher than the baseline 17/18 observed for random systems.

Figures 7.11 and 7.12 show the measurement being taken for a scale-free and random system, respectively, consisting of 500 nodes, whilst Figure 7.13 shows the code, from the observer module, to calculate the Hub Connection Density. It may be noted that the Hub Connection Density for the exhibited scale-free system of 500 nodes is 64 whilst the value for the randomly connected system of 500 nodes is 17. Thus a measure is available to the Observer System whereby a range of values for the Hub Connection Density provides conclusive evidence that a system is conforming to scale-free principles.

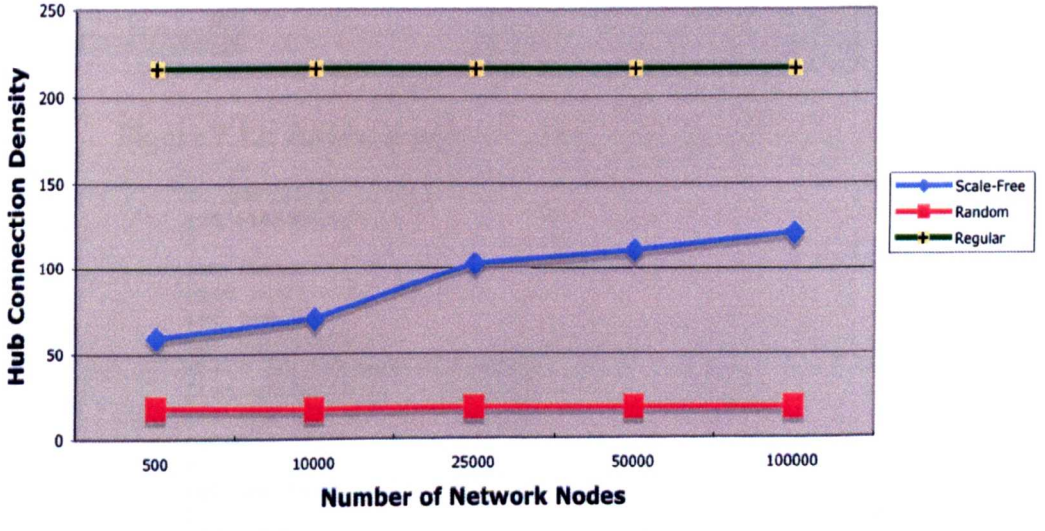

Figure 7.10: The value of the Hub Connection Density measure

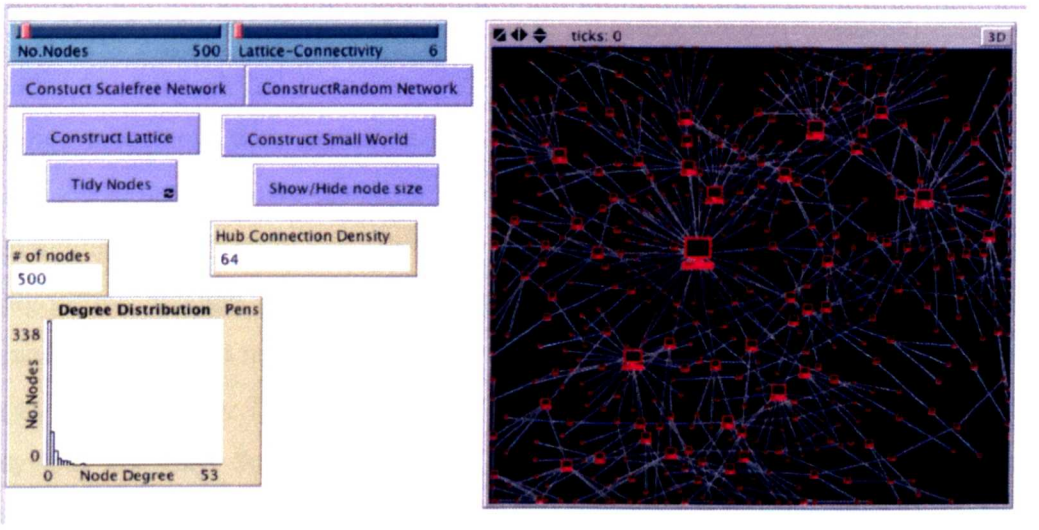

Figure 7.11: Scale-free network Hub Connection Density measure

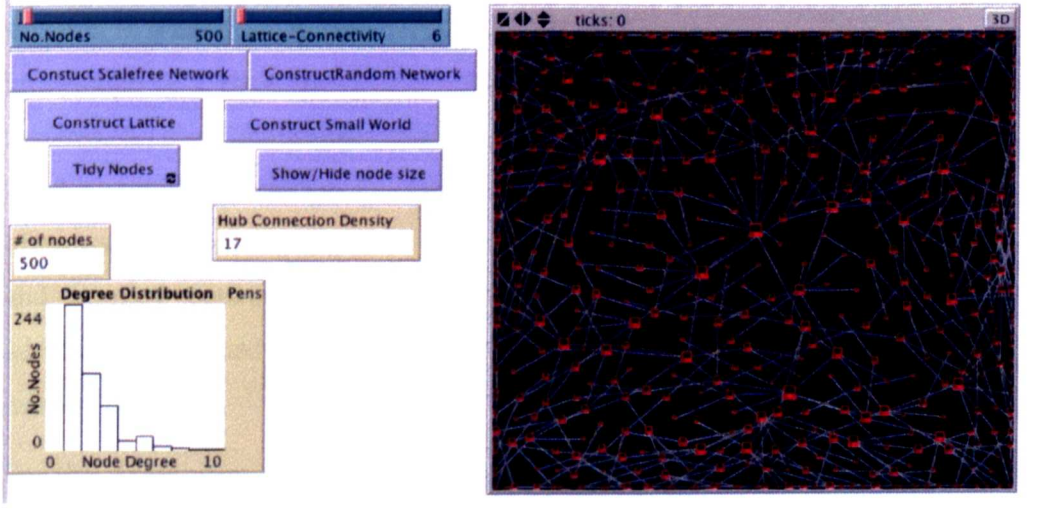

Figure 7.12: Random network Hub Connection Density measure

```
to calculate-HubDensity
set HubDensity 0

ask nodes
[set n 0
set tot 0
set localhub 0
while [n < length neighbor-nodes]
[let anode item n neighbor-nodes
set tot tot + length [neighbor-nodes] of anode
set n n + 1
]
set localhub length neighbor-nodes * tot
]
set HubDensity sum [localhub] of nodes / count nodes
end
```

Figure 7.13: Netlogo code fragment for Hub Connection Density calculation

7.5 Acquaintance Monitoring

Once the network has been constructed, and the hub density is calculated to be in the range associated with scale-free systems, the natural properties of the system can be utilised to determine an optimum monitoring strategy without using any global data and avoiding exhaustive monitoring of the network: The Acquaintance Monitoring Selection algorithm, detailed in the previous chapter, is used on a scale-free system, whereby a randomly selected node is required to nominate a random node to which it is connected (an acquaintance). This will always tend to select a higher degree node because of the system topology. So a random fraction of the node population is asked to indicate an acquaintance node with which they are connected. The acquaintances, rather than the randomly selected nodes, are the designated monitoring points: The probability that a node with k connections is selected for monitoring is proportional to its degree, k. Thus, using this algorithm, the probability that a hub is not monitored is very low, tending towards 0. Therefore system concepts that tend to propagate across a network such as knowledge or a virus are easily disseminated to the hub backbone evident in scale-free systems. As most knowledge is held in the hubs, the fraction of the nodes required to give good monitoring coverage is usually quite small. This hub backbone of super-peers is often used to maintain connectivity in peer-to-peer overlay networks such as in sensor and actuator networks (Traversat et al, 2002). Consequently this process may be additionally utilised to determine the best monitoring strategy in these peer-to-peer overlay networks.

For this experiment, a number of simulations were conducted with varied proportions of selected nodes indicating the percentage chosen. As shown in Figure 7.14, $r = 0.20$ and, for each selected node, one of their acquaintances was randomly selected for monitoring. The proportion chosen was an optimised value, as indicated in work on acquaintance immunisation, tested on random proportions of a population (Cohen et al, 2003).

In Figure 7.14, the white nodes are the monitored nodes, whilst the blue nodes are those originally selected to point to an acquaintance. Hence it can be visually observed that the main network spine and all the high degree nodes have been selected for monitoring by the Acquaintance Monitoring Selection strategy. Also, the monitoring of this system is achieved through the observation of only 79 of the 500 nodes ($f = 0.158$, 15.8%) rather than upwards of 25% when using a random walk algorithm on a scale-free network or 100% when exhaustive monitoring of the network is required.

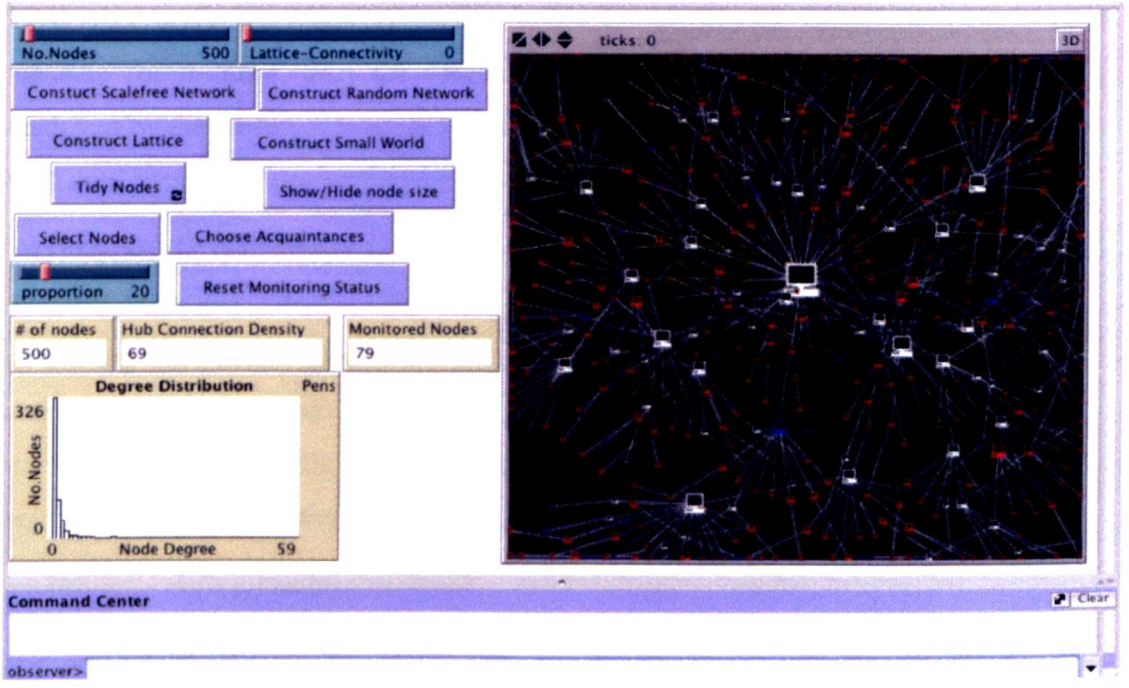

Figure 7.14: An implementation of Acquaintance Monitoring Selection

It may be further observed, in Figure 7.15 that using the same strategy on a randomly connected network results in a greater number of nodes (90) requiring monitoring facilities. It is also obvious, from the display, that the monitoring coverage gained would not be adequate for efficient observation; so that AMS is only applicable to scale-free systems.

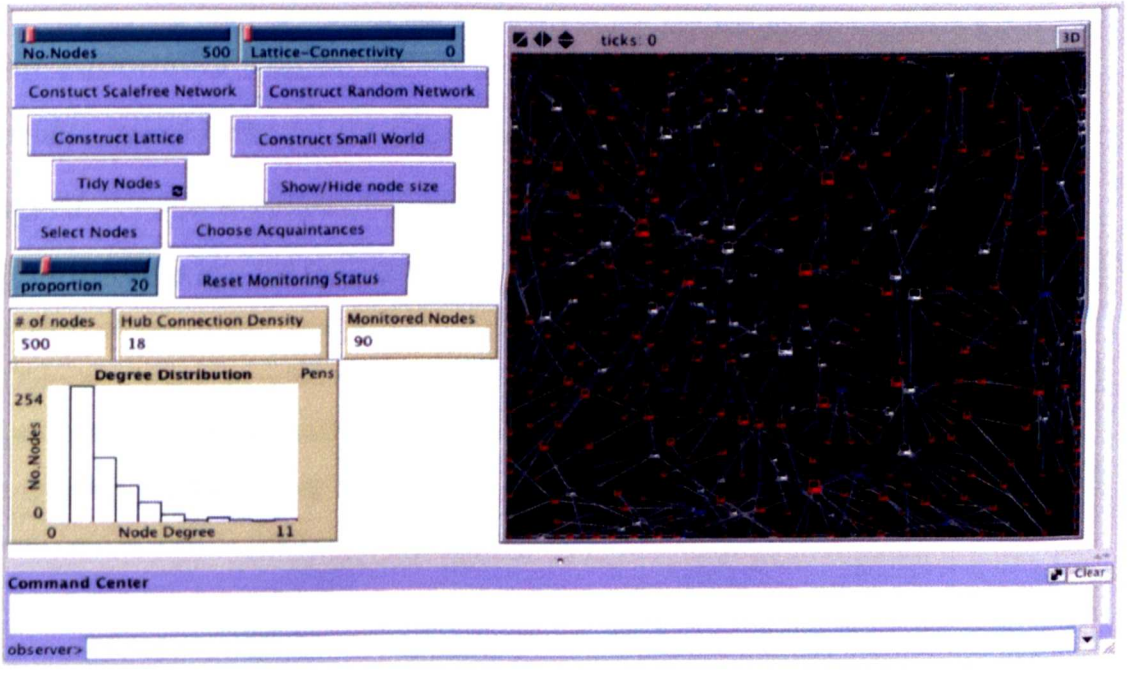

Figure 7.15: Acquaintance Monitoring Selection on a random network

Thus once scale-free connectivity has been detected, by the Hub Connection Density measure for instance, then the Acquaintance Monitoring Selection procedure can be deployed by the Observer System to provide efficient information gathering from specific high data capacity nodes without the need for global information regarding the system.

To illustrate the deliberative process that occurs for node *n* where *monitored* is a Boolean quantity assessing whether the node is monitored, *selected* is a Boolean quantity measuring the truth value of whether a node has been selected to nominate an acquaintance, *proportion* is the user defined ratio, expressed as a percentage, of the number of nodes initially selected from the total number of nodes and the actions of *select, choose, undo and applyAMS* are self-explanatory:

$monitored(n,do(a,s)) \Leftrightarrow [monitored(n,s) \wedge a \neq undo] \vee a = choose(n)$

$selected(n,do(a,s)) \Leftrightarrow [selected(n,s) \wedge a \neq undo] \vee a = select(n)$

$AMS(do(a,s)) \Leftrightarrow$

$[AMS(s) \wedge [a = sense_{HCD} \wedge SR(sense_{HCD}) > 40]] \vee [AMS(s) \wedge (a \neq sense_{HCD})] \vee a = applyAMS$

$proportion(do(a,s)) = r \Leftrightarrow$

$[proportion(s) = r \wedge \neg \exists r'(a = setProportion(r') \wedge r \neq r')] \vee a = setProportion(r)$

$poss(choose(n),s) \equiv \exists n_1 [selected(n_1,s) \wedge neighbors(n,n_1,s)]$

$poss(select(n),s) \equiv AMS(s)$

$poss(applyAMS,s) \equiv HCD(s) > 40$

$prob(s\_select(n),select(n),s) = proportion(s)/100$

$prob(f\_select(n),select(n),s) = 1 - proportion(s)/100$

This translates into the Netlogo code fragment shown in Figure 7.16. It is noted that the *select, choose, undo and applyAMS* actions are nominally under the control of the Observer System as *select and choose* provide normative statements to the nodes and thus provide low-level interaction. The *applyAMS* and *undo* actions are global in nature triggering the lower-level actions. These are in turn triggered by the global emergent outcome of a *HCD* that is greater than 40.

```
to select
ask nodes
[
  if (random-float No.Nodes) < No.Nodes * (proportion) / 100
    [set color blue
     set selected? true
    ]
]
end

to CHOOSE
ask nodes with [selected?]

[ask one-of neighbor-nodes
  [set color white

  set chosen? true]
]
end

to undo
ask nodes
[set selected? false
set chosen? false
set color red
]
end
```

Figure 7.16: Netlogo code fragment for Acquaintance Monitoring Selection

7.6 Distributed Knowledge Retrieval

Finally in this series of implementations/simulations the efficiency of the monitoring is evaluated. This is achieved through placing an identifiable piece of data or, "knowledge nugget", at each node in the network. It is assumed each node has access to its own knowledge and the knowledge initially held by its neighbours, but not the knowledge held by the neighbour gained from its neighbours. Thus the proportion of the knowledge held in the system that is gathered by the Observer, through its monitoring strategy can be evaluated. This was tested, in the Netlogo simulation, on three scenarios with varying system size:

- A scale-free system with the Acquaintance Monitoring
- A scale-free system without Acquaintance Monitoring but with monitoring spread randomly across the system.
- A randomly connected system with monitoring spread randomly across the system.

Figure 7.17 shows a graphical analysis of the results, Figure 7.18 shows the implementation in operation and Figure 7.19 illustrates the code fragment for retrieving the knowledge in an Acquaintance Monitoring Selection scenario.

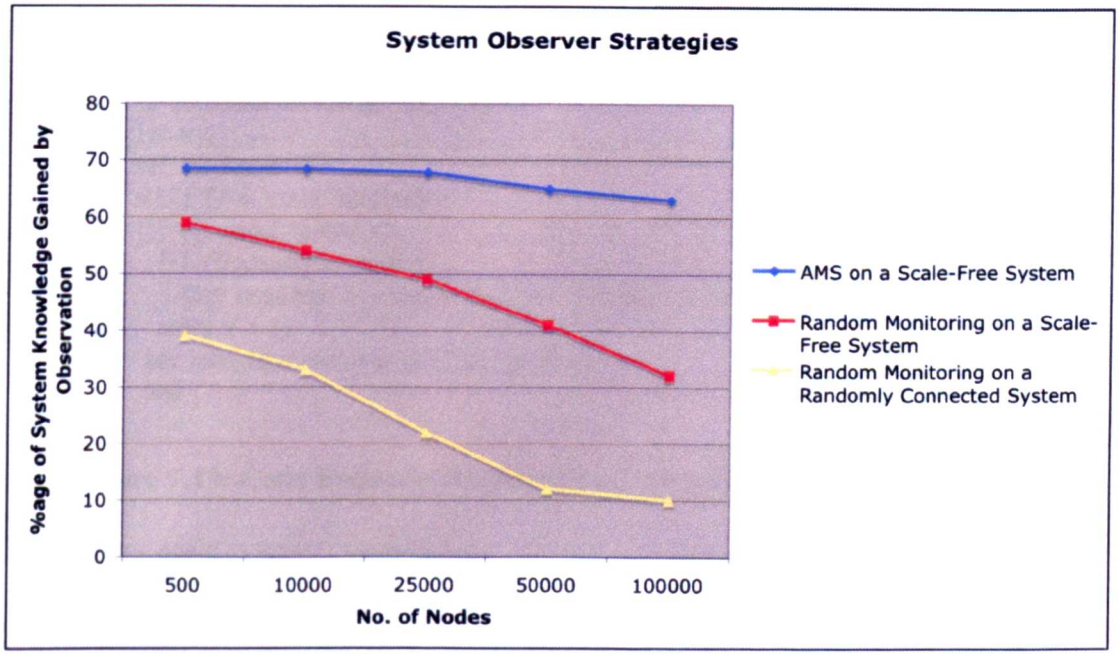

Figure 7.17: Graphical analysis of observer monitoring strategies

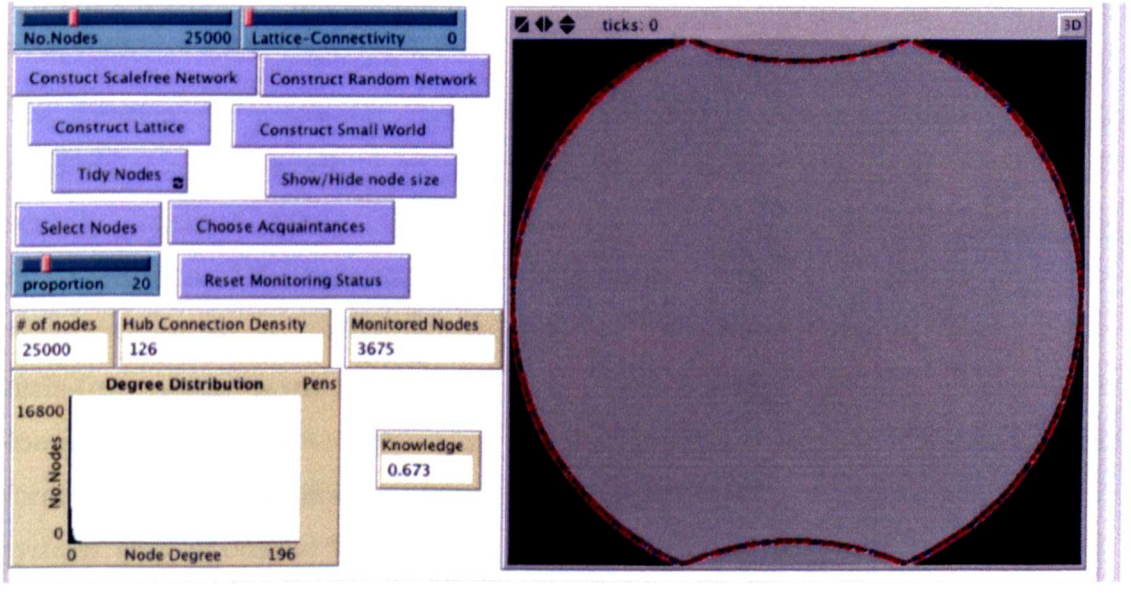

Figure 7.18: AMS on scale-free network showing 67.3% knowledge retrieval

7.7 Summary

This chapter has reported on implementations of the specification for autonomic self-governance in a relatively small-scale scenario, where normative positions supply the control parameters using parametric adjustment, and for larger scale systems, where

mission oriented adaptation is more appropriate, using Netlogo to illustrate the operations of parts of the Observer System.

```
to calculate-knowledge-retrieval
let v 0
set knowledge []
while [v <= count turtles]
[if (is-node? turtle v)
  [if ([chosen?] of node v)
    [set knowledge (sentence (node v) knowledge [neighbor-nodes] of node v)]]
  set v v + 1]
 set knowledge remove-duplicates knowledge
 end
```

Figure 7.19: Code fragment implementing Knowledge retrieval with AMS

The smaller scale system, comprising an autonomic controller, required only one level of observation and system control could be achieved through the system controller either monitoring or providing the access to the distributed tuple system space. More involved procedures are required for larger scale systems necessitating cognitive deliberation over observed or monitored system metrics and measures. There are obviously many different metrics and measures that can be used, by the Observer System, to indicate the current overall state of a system.

The prevalence of power-law connectivity distributions, high clustering and the robust handling of random attacks, observed in the real world systems, makes it beneficial to actually construct systems to these principles. Thus, after displaying a Netlogo implementation of the interpolation between a regular lattice construction and a small-world situation, the next implementation showed an Observer controlled construction of a scale-free system based on the Barabási-Albert algorithm and a simple randomly connected system with the implementation achieved directly from the Situation Calculus formulisation. The visualisation clearly showed the differences in structure between the systems with an ordered hub structure apparent in the scale-free system. The resultant plotting of the nodes connectivity also revealed a normal distribution for the random system in contrast to the long power law tail evident for the scale-free system.

The implementations produced were then used to experimentally evaluate the Hub Connection Density measure, introduced in the previous chapter, for a variety of networks. It was found that, no matter what size of system was encountered for random connectivity the measure remained constant, within very close bounds. For scale-free

systems the measure was considerably higher and increased fairly slowly according to system size. Trivially the highest values occur in well-connected lattices and the associated small-world models.

Acquaintance Monitoring Selection was shown with a clear visualisation that the algorithm specified for deliberation in Situation Calculus and implemented in Netlogo, produced a highly efficient monitoring strategy for scale-free systems. It was also clear, from a visualisation, that the algorithm would not produce similar results when random connectivity is the primary system construct, confirming previous analytical analysis.

The final testing of the implementation assessed the amount of knowledge, which was uniformly distributed over a network, available to the Observer for different sizes of system with various monitoring strategies. Acquaintance Monitoring Selection was clearly a very efficient strategy no matter what scale of system it was applied towards.

Further work in this area is illustrated in Appendix 2, a recently published, co-authored technical paper. The work is shown applied to a simple emergent phenomenon, akin to behavioural examples observed in ant nests (Bonabeau et al, 1999), with the signal for the emergent behaviour grounded by the techniques described herein. The action history is captured situation by situation and, thus, may be subject to the situation space-trimming process previously described.

Chapter 8

Evaluation: Two Case Studies

The evaluation of the proposed approach, to provide scalable self-governance facilities to large-scale autonomic systems, has been carried out through two main case studies. The central notion to the work is the establishment of a formal specification mechanism to permit not only the definition and enforcement of a system's normative positions towards itself, as a whole, and its comprising components, as normally apparent in centralised architectures, but also the establishment of a deliberative cognitive system for the run-time analysis of emergent system behaviour that may become evident in distributed systems with autonomous components. In Chapter 4 the Stochastic Situation Calculus was proposed as a formalism that displays many features to make it suitable for such a task. The first case study uses the formalism in a more traditional setting by providing the specification of a decision support system for breast cancer clinicians. This includes both the logical statement of decision tree rules and the specification of the deliberation by the cognitive system to reason on the runtime system's operation. The second case study involves a proposed NASA space mission where very many small spacecraft cooperate and interact to explore the asteroid belt. The formalism is used to specify the interactive behaviour of the component spacecraft and to consider the three aspects of this work described and analysed in Chapter 5: The detection of new emergent behaviour, the replication of previously observed interactions to engineer emergence and the utilisation of a known topology upon the system's detection of particular classes of prevalent self-organisational behaviour, such as scale-free connectivity.

The perspective used throughout is that of the Observer System, which is a separate concern to the system itself, although the norms and guards placed on the individual components, including the system itself, are relayed through the Observer System.

8.1 A Decision Support System for Breast Cancer Clinicians

This case study was completed as part of an Engineering and Physical Sciences Research Council (EPSRC) funded project entitled: Towards a Disciplined Approach to Integrating Decision-Support Systems for Breast Cancer Care Activities (The 2nrich Project, 2006). The project represented collaboration between computer scientists, statisticians and

clinicians from Liverpool John Moores University, the Christie Hospital, Manchester and the Linda McCartney Centre of the Royal Liverpool Hospital to provide decision support for post-operative breast cancer care. A major concern of the project was that clinicians' decisions ought not to be supplanted, but rather supported and adapted to, by such a system. Thus it was necessary for the system to possess a separated cognitive meta-system to reason on the behaviour of the system as a whole, including the clinician's actual decision. This facilitated the three stated aims of the project: Supporting the clinician's decision within the National Institute for Health and Clinical Excellence (NICE) guidelines; the analysis of historical data to produce new rules for treatment choice and a combination of the previous two notions to provide a decision support system that is adaptive to clinicians' needs and changing environments. Thus, in presenting this case study, the proposed Situation Calculus formalism is shown to be equally useful for defining application system rules as well as specifying meta-system operational procedures. The specification describes a centralised; observation controlled decision support system, which nevertheless may also be considered as a distributed component subject to monitoring by a higher-level observer. Through this case study the specification of a separated observer system is demonstrated for the establishment of self-governance norms with deliberation based on reward and cost functions for norm adaptations.

8.1.1 The Decision Support System

This medical decision support system, based on guidelines, can be viewed as a human and artificial/component multi-agent system (Moreno, 2003). The medical decision process comprises of an information field that includes all the agents and objects involved in the decision with a shared ontology to describe the environment. So the information field is defined by the common ontology in the form of a normative structure where the human and artificial agents, involved in the decision process, abide by the norms. However some autonomy must also be allowed within the component/human decision processes so that an agent can 'choose' not to follow a norm in certain circumstances. Thus this case study represents a fairly routine application of the concepts described in this work. The Observer System takes the form of a single meta-system governing the operation of a decision support component. The meta-system is required to maintain quality of service through the monitoring of system performance measures, such as response time, and quality of process through the assessment of compliance to a guideline

model and the inference of new rules based on the feedback from previous decision instances.

So the medical decision process is a goal-governed collection of individual agents. The agents may be autonomous, heterogeneous, rational and social. Norms arise from a social situation in that they involve providing rules for a community of more than one individual (Borman, 1999). The norms can be associated with the decision-making process, as a practical application, or with the separate meta-system for governance of the system and clinical processes. The provision of adjustable autonomous agent behaviour is necessary to deliver the system services in a way where the majority of the administrative tasks are handled within the software itself. However this autonomy must be reconciled with the governance of the system process.

Typically a medical decision support system consists of a *set of patient data inputs* that are matched against a range of treatment options resulting in the best treatment option being output as a decision. The PROforma system (Fox et al, 1996) advanced this notion to consider the medical decision support system in an object-oriented manner. A simple representation of the class structure is shown in Figure 8.1.

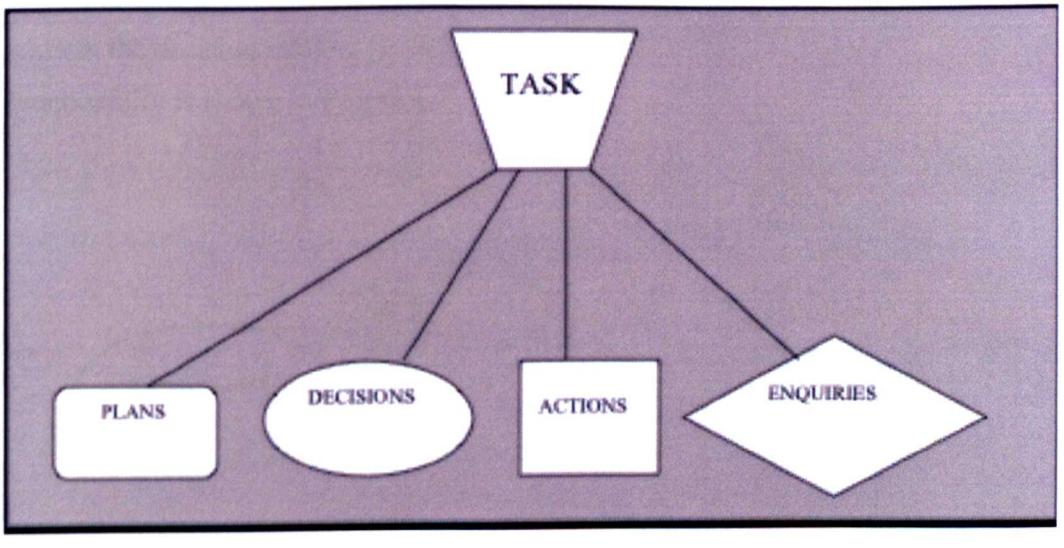

Figure 8.1: Class structure for object-oriented decision model

The decision may then be reached through employment of a regular decision process with some adaptation as illustrated in Figure 8.2.

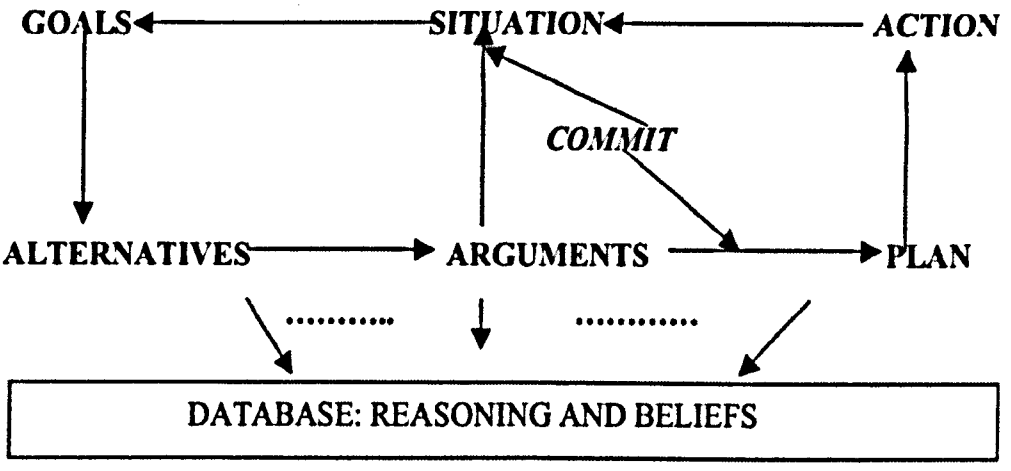

Figure 8.2 Decision process and update

Although this procedure, as illustrated, allows adaptation of the decision making process with feedback of outcome into the knowledge base, the governance process is very difficult to achieve as there is no provision for runtime adaptation of the system itself. As detailed in Section 2.2.2 messages need to be passed before classes or aspects can become active in the system. This work thus provides the formal setting of a separated Observer type system to handle the runtime deliberation encompassing quality of service and quality of process concerns for the system, whilst also providing a formal means to address the decision making processes that are the system's core functionality. This basic functionality is shown in Figure 8.3.

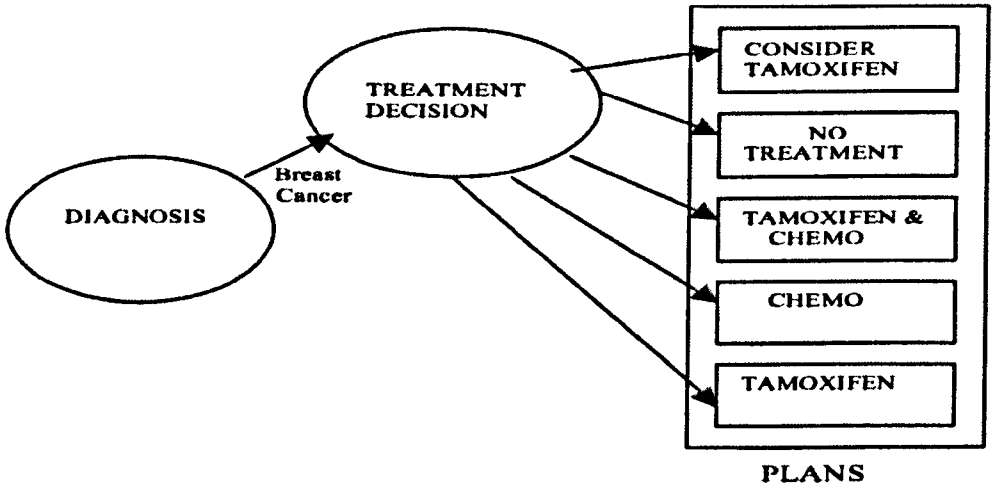

Figure 8.3: Decision options for post-operative breast cancer care

The decision support system considered in this case study was required to output treatment options based on rule sets with patient data as inputs. Additionally, however,

the system was also required to exhibit self-governing features. Thus, for instance, a simple treatment rule such for using the drug tamoxifen, which is recommended in the guidelines for post menopausal patients with positive oestrogen receptors, can be formally stated as:

$$NICEtreatment(patient, tamoxifen, do(a,s)) \Leftrightarrow [NICEtreatment(patient, tamoxifen, s) \wedge \neg \exists treatment(a = nice\_treament\_decision(patient, treatment) \wedge (treatment \neq tamoxifen))] \vee [a = nice\_treatment\_decision(patient, tamoxifen)]$$

with

$$poss(nice\_treatment\_decision(patient, tamoxifen), s) \Rightarrow (oesreceptor(patient, s) = pos) \wedge (menostatus(patient, s) = post)$$

Similar rules can, trivially, be stated for the other options, within NICE or any other set of guidelines. These rules are, however, irrelevant for the purposes of this work; merely forming a plug in component to be managed by the completely separated meta-system observer/controller. The main concerns of the autonomic type meta-system requirement are the maintenance of quality in service and process.

Quality of Process: The rules governing the meta-system are required to reason over the operation of the system. A quality of process concern may involve assessing the clinician's adherence to a specified guideline treatment:

$$compliance(patient, treatment, service\_decision, do(a, s)) \Leftrightarrow [compliance(patient, treatment, service\_decision, s) \wedge a \neq treament\_decision(patient, treatment1)] \vee [a = treatment\_decision(patient, treatment) \wedge service\text{-}decision(patient, s) = treatment]$$

In order to express the efficacy of the treatment decisions for the quality of process it is obviously appropriate to consider the outcomes of courses of treatment. These courses of treatment occur over a period of time and it is thus necessary to employ the Situation Calculus technique that deals with the duration of actions. Therefore the treatment duration is covered by two instantaneous actions *startTreatment* and *endTreatment* with a

fluent *treating* through the duration of the treatment. So, for a patient *p*, continuing with the example of tamoxifen treatment given above, there may be a reward function such as:

$$reward(treatment(p,tamoxifen), do(a,s))=r \Leftrightarrow a=endTreatment(p,tamoxifen) \wedge [(r=100 \wedge living(p,s)) \vee (r=-500 \wedge \neg living(p,s))]$$

with

$$fitness((treatment(tamoxifen) \wedge menostatus(p,s)=post \wedge oesreceptor(p,s)=pos), do(a,s)) = fitness((treatment(tamoxifen) \wedge menostatus(p,s)=post \wedge oesreceptor(p,s)=pos), s) + reward(treatment(p,tamoxifen), do(a,s))$$

In this way the success of each rule occurrence can be assessed including previously untried treatments instigated by the clinician and flagged for non-compliance. Thus less successful treatments will be deleted from the decision options whilst the decisions that lead to more favourable results will be chosen, improving the quality of process.

Quality of Service: Quality of service concerns can be dealt with by the meta-system through monitoring demand against system capacity. If response time is too slow, for instance, it may be necessary to regenerate the service at a location closer to areas of higher demand. The service location may be specified as:

$$at(location, service, do(a,s)) \Leftrightarrow (at(location, service, s) \wedge (a \neq move(service,location1) \vee a \neq delete(service, location))) \vee (a=regenerate(service, location))$$

To facilitate the regeneration of a service due to high demand requires the specification of a procedure to detect the behaviour and prescribe the action to take to rectify the potential failure. The CPU load can be monitored as in Section 6.2

$$cpuload(do(a, s)) =n \Leftrightarrow [cpuload(s) =n \wedge a \neq sense_{CPULOAD}] \vee [a =sense_{CPULOAD} \wedge SR(sense_{CPULOAD}, s)=n]$$

$$heavyload(do(a, s)) \Leftrightarrow [heavyload(s) \wedge ((a \neq sense_{CPULOAD}) \vee \neg(a= sense_{CPULOAD} \wedge SR(sense_{CPULOAD}, s) <60))] \vee [a = sense_{CPULOAD} \wedge SR(sense_{CPULOAD}, s) >60]$$

with response time given as

$$roundtriptime(do(a, s)) = n \Leftrightarrow [\, roundtriptime(s) = n \wedge a \neq sense_{ROUNDTRIPTIME}] \vee [a = sense_{ROUNDTRIPTIME} \wedge SR(sense_{ROUNDTRIPTIME}, s) = n]$$

$$unresponsive(do(a,s)) \Leftrightarrow [unresponsive(s) \wedge ((\, a \neq sense_{ROUNDTRIPTIME}) \vee \neg(a = sense_{ROUNDTRIPTIME} \wedge SR(sense_{ROUNDTRIPTIME}, s) < 1000))\,] \vee [\, a = sense_{ROUNDTRIPTIME} \wedge SR(sense_{ROUNDTRIPTIME}, s) > 1000]$$

$$regenerated(service, location, do(a,s)) \Leftrightarrow regenerated(service, location, s) \vee a = regenerate(service, location)$$

with

$$poss(regenerate(service, location), s) \Rightarrow heavyload(s) \vee unresponsive(s)$$

This shows the specification from an immediate observers perspective giving an example of an autonomic system based on parametric adaptation as defined in Section 3.3.1. The representation describes a centralised control structure, where the observer acts as the system controller. The actors, within the system, are mandated to act within the normative position of the system as monitored by the Observer System. A full implementation of this system was completed as part of the EPRSC funded project; this is, however outside the scope of the work presented in this thesis. Further details may be found in (Miseldine et al, 2006), (Taleb-Bendiab et al, 2006), (Miseldine and Taleb-Bendiab, 2005b) and (Randles et al, 2005). The decision support system component produced in this way may form part of a wider distributed system. The next case study shows the observer system ranging over very many autonomous interacting components providing an example of centralised norm based governance from the top down coupled with cognitive observation based monitoring utilising system emergence from the bottom-up.

8.2 The NASA ANTS Project.

This second case study is adapted from the behaviour specification of worker/ruler robots in the NASA Autonomous Nano-Technology Swarm (ANTS) project (Rouff et al, 2004). Formulation and deliberation of emergent properties occur at observer level with the necessary checks and guards incorporated as a norm based deliberative model. This shows how the specification method, proposed in this work, may be deployed through a large-scale system comprised of very many individual components.

Briefly stated the ANTS project arises from a class of space exploration missions termed *nanoswarms,* where many cooperating intelligent spacecraft work in teams, based on the efficiency and coordination of a hive culture. In particular the project, which is under conceptual development, by NASA, to occur in the 2020s, envisages a thousand picospacecraft working cooperatively to explore the asteroid belt. Once commissioned the teams consist of three classes of spacecraft combined in specific ways to form teams that explore individual asteroids, as shown in Figure 8.4. *Workers* make up 80% of the swarm and carry the instruments to gather data. *Rulers* coordinate data gathering by assembling teams of appropriate *workers. Messengers* manage communications between *workers, rulers* and mission control on earth.

Figure 8.4: The NASA ANTS project (Rouff et al, 2004)

This case study addresses the specification of the system through establishing logical consequences, from the top-down specification of the interactions of the participants, based on the formal setting of federated behaviour, and observing, deliberating on and utilising emergent behaviours that occur from the bottom up. Firstly a minimal rule set is formed so that additional behavioural rules follow as a result of the specification, followed by an analysis of emergent system features that might occur within the Observer system. The frameworks, previously described to specify federated behaviour, as in Section 2.4 and to deliberate on the emergence of various forms in large-scale system

tendencies, as described through Chapter 5, are particularly applicable to this NASA project. This is because the mission is knowledge critical. The overall aim is to gain knowledge for use by scientists, back on earth, whilst the epistemic state of the swarm members is crucial in stabilising the behaviour of the swarm to carry out its intended duties. Firstly the team assembly is considered through the formalism established for federated behaviour; imperatives for the mission may be stated and properties shown to be entailed as logical consequences. Secondly the formation of the ruler team is considered and it is shown how this leads to the system self-organising to robust topologies. The (Stochastic) Situation Calculus is used throughout as the logical formalism.

8.2.1 NASA ANTS Formal Specification

Firstly, for this specification, it is necessary to account for the domain behaviour as a large team of interacting component spacecraft. This builds upon the work reported in Chapter 2, in particular the federated behaviour described in Section 2.4 set in a Situation Calculus representation.

It is known that as complexity increases and failures tend to occur more frequently federations of component agents, or spacecraft in this case, waste fewer resources and are more robust than solipsistic entities (Jennings, 1995). The formation of a component federation is characterized, as described in Section 2.4, by the concept of a Joint Persistent Goal (JPG). Thus member spacecraft are committed to the team, as it is comprised during their membership. The JPG formalises the joint commitment of the collective. That is the JPG imbues the federation of Pico-spacecraft with the goal-directed behaviour of a single component agent. Tasks, to realise completion of the goal, are performed by sub-agents, within the team, as part of the federated society. To have a joint intention to perform an action, a, the team will have a JPG to do a within a particular subset of the situation space. The joint intention assumes the initial belief, by all the members, that the swarm is going to complete the intended action next. Following the enactment of a joint intention the team of spacecraft will mutually come to believe one of three things, from the initial mutual belief that the team members are going to work jointly on the intended action:

1. That a has been done.

2. That a is impossible to do.

3. That *a* is irrelevant

Firstly considering for a single craft a persistent goal can be represented in the Situation Calculus by:

$$PG(p, do(a, s)) \Leftrightarrow$$
$$(PG(p,s) \wedge \neg(\neg B(p,s) \vee knows(p,s))) \vee (a = setPG(p) \vee knows(\neg p,s))$$

As previously introduced, in Chapter 4, *knows* is the knowledge operator in the Situation Calculus with the analogous belief operator B [B(p) = ¬knows(¬p)]. Obviously with only one component agent these are $knows_i$ and B_i, with the number of agents, n=1. This agrees with an intuitive notion that a goal persists while an agent doesn't know p to be true and doesn't know that p cannot be true. Replacing the belief operator with its corresponding formulation using the knowledge operator in the above formula and working through the expression gives an equivalent formula:

$$PG(p, do(a, s)) \Leftrightarrow$$
$$(PG(p,s) \wedge (\neg knows(\neg p,s) \wedge \neg knows(p,s))) \vee (a = setPG(p) \vee a = SR(sense_p, s) \rightarrow \neg p)$$

Now to extrapolate this to n component team members in a multi-agent type setting, following the reasoning of Section 2.4, it is necessary to think of the spacecraft team as acting as a single entity. However this is not sufficient as it must also be possible for any agent to become aware individually that a goal has been achieved or is impossible to reach. So when a team member comes to believe a fact, regarding the goal, which entails the dropping of the commitment by the individual member, the federation must also drop the commitment. Thus each member has as a sort of lesser goal one of three choices. Either it believes p not to be the case but has p as a goal or it believes p does hold and has a goal to make this common knowledge or it believes p to be impossible and has a goal to make this common knowledge. So if there is a lesser individual goal, for each component spacecraft i, relative to the other members of the team τ, LIG_i:

$$LIG_i(\tau,p, do(a,s)) \Leftrightarrow LIG_i(\tau,p,s) \wedge \neg((knows_i(\neg p,s) \wedge G_i(p,s)) \vee (knows_i(p,s) \wedge G_i(C(p),s)) \vee (\neg B_i(p,s) \wedge G_i(C(\neg p),s)))$$

Where C is the common knowledge operator, as defined in Section 2.4 with a situational term. In this way each federation member of the spacecraft team doesn't assume that the other federation members of τ have p as a goal but rather as a lesser individual goal. Thus any federation member may have discovered p has been achieved or is impossible and be in the process of making this common knowledge in the federation. So a joint persistent

goal is established, using the *"everyone knows"* operator defined in Chapter 2, through team mutual belief with single agent belief replaced by common knowledge and lesser individual goal as appropriate:

$JPG(\tau,p, do(a,s)) \Leftrightarrow$

$$JPG(\tau,p,s) \wedge (\neg(E(\neg p,s) \vee E(p,s)) \wedge (\exists i \in \tau(LIG_i(p,s)))) \vee (a=setJPG(p) \vee E(\neg p,s))$$

This then provides an example mathematical model of the teamwork behaviour, required in the system, to set defining goals for the spacecraft federation. In this way a scaleable method of specifying distributed control emerges, from a minimal set of system imperatives for the component entities, to determine behaviour by logical entailment and emergent novelty.

The goals to be satisfied rely on team-work within the federation. So without team formation, or the notion of the logical process of team assembly, it is impossible to continue with the process. The workers under each Ruler obviously form a federation, additionally the Rulers can be conceived of as forming a Ruler team that is connected to the workers so that the most appropriate team members can be selected for particular mission tasks. Each Pico-spacecraft, in a team, is committed to successfully performing its quota of tasks as well as to the success of the team as a whole. So the formalism of the domain can proceed as follows with situation terms added, as the final argument of the fluents, where required:

- worker(w) w is a worker spacecraft
- ruler(r) r is a ruler spacecraft
- W, R are worker and ruler teams with τ representing a general team that may consist of heterogeneous units.
- knows(w,p) means p follows from a worker w's knowledge
- G(w,p) means p follows from a sequence of worker w's available actions
- I(w,a) means a is an intention of worker w
- $E(\tau,p)$ means everyone believes p in team τ (eg $knows(w_1,p) \wedge knows(w_2,p)$
- $\wedge$.....$knows(w_n,p)$ where $\tau=\{w_1, w_2,.....w_n\}$
- $C(\tau,p)$ means p is common knowledge in team τ [ie $E(\tau, p) \wedge E(E(\tau, p)) \wedge$.....]. This represents universal mutual belief in that every team member believes that every other team member believes p
- member(r,R) means ruler r is a member of ruler team R
- registered(w,R) means worker w is registered with ruler team R

- connected (w, r) means worker w is connected to the ruler team via ruler r.

Using the formalism of joint persistent goals set in the Situation Calculus, an augmented variation of dynamic logic (Harel, 1979), based on the composite action defined in Section 4.4.2, will be applied to the Situation Calculus representation to produce a logical implementation, similar to pseudo-code, to provide formal reasoning techniques for the ensuing programs. So:

- p? means give p a valuation (ie true or false)
- $a_1;a_2$ means action a1 followed by action a2
- $a_1|a_2$ is non deterministic choice
- Complex action expressions can be used such as IF-THEN and WHILE-DO
- occurs (a*) means a sequence of actions a* (= $a_1,a_2,.....a_n$) is scheduled to happen next. It is noted that $a^*=do(a_n,do(a_{n-1},..do(a_1, S_0))...)$ in the Situation Calculus where S_0 is the starting situation for the action sequence.
- finished (a*) means the sequence of actions has occurred.

This treatment is mostly achieved using First Order Logic (FOL). There is an appeal to some Second Order Logic (SOL), however this is solely to describe smallest sets with certain properties, there is no requirement to quantify over function variables.

This is done in order to more easily semantically state the properties of actions and their consequences.

Now it is possible to state some imperatives that the teams must follow. In this way low level interactions are specified that promotes emergent self-organisation.

Ruler Team Perspective: Imperative I: *When a worker (w) registers on the mission, by registering with the Ruler team (R), and has not entered any failure state leading to its "death", the Rulers possess a team intention to connect with the worker, should it ever be disconnected.*

So in situation s, after the team is formed, the ruler team have a Joint Persistent Goal

$JPG([\forall w \exists r \in R(connected(w,r))], s)$

The *connected* fluent can take the situational value independent of its goal attribution

$connected(w,r,do(a,s)) \Leftrightarrow (connected(w,r,s) \wedge a \neq [SR(sense_w,s)=nil]) \vee a=connect(w,r)$

$poss(connect(w,r),s) \Rightarrow (r \in R) \wedge registered(w,R,s)$

This converts into a dynamic logic representation:

$\models \forall w\ [(worker,w) \wedge finished(registered(w, R)?) \Rightarrow$

$$(I(R, connect(w, R))]$$

Where $connect(w, R) = (WHILE\ registered(w,R)\ DO$

$$[IF \neg \exists R\ connected(w, r)\ THEN$$

$$[\ UNTIL\ [\exists r \in R\ connected(w, r)]\ \forall r\ I(r, connect(w, R))\ ]]$$

It should be noted that a team intention (I(R, connect(w)) in the above formula) is stronger than just a collection of individual team member intentions. It is based on the team joint persistent goal, which requires the team members to have beliefs regarding team membership. This requires team members to hold the lesser individual goals (LIG) in certain circumstances. That is if a ruler team member believes that the goal p (say) is not achieved it has an individual goal to see to it that p eventually holds.

If it believes: the goal has been achieved; cannot be achieved or is of no consequence then it has an individual goal to make this mutually believed by the ruler team. If it believes there to be a new team member or that a member has left the team then it has a lesser individual goal to make this a belief for all the other team members. Thus for each r∈R:

$$LIG_r([(\forall r' \in R)C(R, (r' \in R)], s) \wedge LIG_r(p,s)$$

This lesser individual goal for r∈R is represented by

$LIG_r(R, p) \Leftrightarrow$

$$[[\neg knows(r,p) \wedge G(r,p)] \vee [knows(r,p) \wedge G(r,C(R,p))] \vee [knows(r,\neg p) \wedge G(r,C(R,\neg p))]] \wedge$$

$$[[\forall r'\ (knows(r, \exists a(finished(member(r',R))?;a;\neg member(r',R)?)))] \Rightarrow$$

$$G(R, C(R, \neg member(r',R)))] \wedge$$

$$[[\forall r'\ (knows(r, \exists a\ (finished(\neg member(r',R))?;a;\ member(r',R)?)))] \Rightarrow$$

$$G(r, C(R, member(r',R)))]$$

Where a is a single action

A lesser team goal (LTG) means that the ruler team mutually believes that every ruler in the team has the LIG:

$$LTG(R,p) \Leftrightarrow C(R,(\forall r \in R\ LIG_r(R, p)))$$

So now the team R has p as a mutual goal when the team mutually believes p is not achieved, there is a mutual goal to achieve p and there is a lesser team goal to achieve p until the team come to believe that p is achieved, unachievable or irrelevant:

$$G(R,p) \Leftrightarrow knows(R,\neg p) \wedge G(R,p) \wedge [UNTIL[knows(R,p) \vee knows(R,\neg p)]LTG(R,p)]$$

So finally it can be written:

$I(R,connect(w)) \Leftrightarrow G(R, finished(\ C(R,\ occurs(R,\ connect(w,R)))?;connect(w,R))$ **(1)**

This gives the power of mutual beliefs (from common knowledge) leading to mutual team goals to give team intentions and shows how the team intention I(R,a), for action a, is syntactically different from the worker intention I(w,a) just because the goal G(R,p) is similarly different from the goal G(w,p).

In general this formalism also holds for any team. So the team under consideration could be the worker team or a heterogeneous team of variable members. This distinction then makes it relatively straightforward to specify team goals, in a similar way to individual goals, but with the proviso that the above team goal formalism holds.

So a number of logically provable consequences emerge as a result of **imperative I**. For instance:

When a registered worker becomes disconnected the ruler team have a team commitment to reconnect with the worker throughout the lifetime of the team.

Proof: The assertion can be logically stated as:

$$\models \forall w[C(R,\ registered(w,R)) \wedge C(R,\ (\neg \exists r \in R\ connected(w,r)))] \Rightarrow$$
$$G(R,\ (finished(connect(w,\ R))))$$

So assume the left hand side of the implication is true then expanding imperative I using the definition of team intention **(1)** gives a team goal:

$$G(R,\ (finished(C(R,\ occurs(connect(w,R))))?;connect(w,\ R)))$$

So let

$$connect(w,R) = pre(w,R);(registered(w,R))?;$$
$$[(\neg \exists r \in R connected(w,r))?;connect(w,R) | (\exists r \in R\ connected(w,r))?];post(w,R)$$

Where $pre(w,R)$ represents the previous iterations and $post(w,R)$ denotes the remaining iterations.

So substituting gives the ruler team goal of:

$$G(R,\ (finished(C(R,\ occurs(connect(w,R))))?;pre(w,R));(registered(w,R)?;$$
$$[(\neg \exists r \in R connected(w,r))?;connect(w,R) | (\exists r \in R\ connected(w,r))?];post(w,R))$$

The iteration of interest is the one where the conditional becomes true. So assume the action sequence

$$C(R,(occurs(connect(w,R))))?;pre(w,R);registered(w,R)?;(\neg \exists r \in R\ connected(w,\ r))?$$

has just occurred. Now for actions a and b with a general team τ it is known that:

$$\models G(\tau,\ finished(a;b)) \wedge C(\tau, finished(a)) \wedge C(\tau, \neg finished(b)) \Rightarrow G(\tau, finished(b))$$

Applying this to connect(w,R) in imperative I gives:

$G(R, (finished(C(R, occurs(connect(w,R))))?;pre(w,R));(registered(w,R)?;$

$[(\neg\exists r{\in}R connected(w,r))?;connect(w,R)|(\exists r{\in}R\ connected(w,r))?];post(w,R))\Rightarrow$

$G(R, finished(connect(w,R));post(w,R))$

However by assumption $C(R,\neg connected(w,R))$ holds so that members of the ruler team mutually believe that the action [$\neg\exists$r∈R connected(w,r)]? has occurred so they then have a team commitment to do connect(w,R). Thus:

$G(R, finished(connect(w,R));post(w,R)) \Rightarrow G(R, (finished(connect(w,R))))$

which was the result to be proven?

In this way logical consequences of the specification emerge as verifiable, provable outcomes for the system. Thus, with a meaningful specification of teamwork, imperatives can be stated from which emerge additional, not necessarily intended, features of the system. To continue a further imperative for the team might be stated as:

Ruler team: Imperative II: *The ruler team must have be made up of at least a specified number of ruler units:*

The Joint Persistent Goal for a team of rulers *R*, where the minimum ruler number is *N* is

$JPG(numberofRulers(R)>=N, s)$

with the usual Situation Calculus definitions:

$numberofRulers(R,do(a,s))=M\Leftrightarrow(numberofRulers(R,s)=M)\vee$

$[(numberofRulers(R,s)=M-1)\wedge$

$\exists r\ a=join(r, R)]\vee[(numberofRulers(R,s)=M+1)\wedge\ )\wedge\exists r\ a=leave(r, R)]$

$poss(join(r,R),s)\Rightarrow ruler(r,s)$

poss(leave(r,R),s)⇒member(r,R,s)

So the imperative states:

$\models G(R,numberofRulers(R)>=N)$

A number of results follow in a logically provable manner, for example:

Rulers as individuals have a commitment to maintain the number of rulers above a specified level.

Thus when one ruler believes that the number of rulers is less than required and believes that it is not mutually believed by the team and believes it is not impossible to establish mutual belief for the team then it has an individual commitment to bring about this mutual belief.

Proof: The assertion states:

$\vdash G(R,p) \Rightarrow \forall r \in R[knows_r(\neg p \wedge \neg C(R,\neg p)) \wedge \neg knows_r(\neg C(R,\neg p)) \Rightarrow$

$G(r, C(R,\neg p) \wedge knows_r(\neg p))]$

where $p=(numberofRulers>=N)$

proof: Let $r \in R$ and assume

$$\forall r \in R\ knows_r(\neg p \wedge \neg C(R,\neg p)) \wedge \neg knows_r(\neg C(R,\neg p))$$

Now from the definition of team goal it can be stated:

$$knows_r(\neg p) \wedge \neg C(R,\neg p) \Rightarrow G(r, C(R,\neg p))$$

and $G(r, C(R,\neg p))$ is satisfied because if one member of the team does not believe there is mutual belief then there is no mutual belief. So since the consequent of an implication must remain true until the antecedent or the implication statement becomes false

$G(r, C(R,\neg p))$ holds until $r \in R\ knows_r(\neg p \wedge \neg C(R,\neg p)) \wedge \neg knows_r(\neg C(R,\neg p))$ doesn't; that is until

$$\neg knows_r(\neg p) \vee C(R, \neg P) \vee knows_r(\neg C(R,\neg p)) \text{ is true.}$$

So all the conjuncts in the definition of $G(r,C(R,\neg p) \wedge knows_r(\neg p))$ are satisfied thus proving the result.

The complex nature of such a system is evident in the aggregation, labelling, data flow, nonlinear responses and diversity, which occur to provide a meta-monitoring structure from the agents. The entire mission comprises of an aggregated team of robots whilst individual workers, for example, consist of aggregated systems for propulsion, monitoring, communication, etc. As such each system can be labelled from worker/ruler down through lower level systems. Data flow is evident in the sensing functions to determine action. The nonlinear response occurs in the teamwork dynamic.

Diversity occurs as separate federations develop and evolve to specialize at separate specific tasks. The prerequisites for self-organising behaviour, as defined in Section 2.5 (Glansdorff and Prigogine, 1978), can be accommodated in the proposed knowledge acquisition framework, namely:

1) The interaction of Pico-spacecraft, in the swarm, is mutually causal in that the state of one component causes an action to be instigated in another and vice versa.
2) Autocatalysis is common as when a ruler, for instance, fails another is influenced to increase itself to take on the extra duties or impel the collective to do so.

3) The system is open to the environment. Indeed its main concern is to deal with environmental effects taking in resources from the environment, building itself into an ordered structure and feeding back into the environment.
4) Random variations are very evident in such systems with hardware malfunctions, variable demand, exogenous actions etc.

Hence although a simple system scenario is presented the interaction between components enforce a complexity that engenders self-organising behaviour. Simple norms in the form of imperatives can be laid down which in turn affect the system through the consequences of logical entailment that can be rigorously established.

Additionally certain behaviours can emerge that are completely separate from any rules that can be derived logically. For instance certain team formations may emerge as more efficient in carrying out specific tasks. In this treatment it is the observer's perspective of the ruler team that has been considered. However the general perspective of the observer can be treated in a similar manner to provide the signal grounding, emergence engineering and utilization of known properties of widely prevalent large system emergent outcome, such as scale-free connectivity.

For example the ruler team can be assembled, to robust organisational principles, where the large-scale organisation emerges from the small-scale actions/interactions, as described above, under the influence of the hierarchical observer systems equipped with the self-organising epistemology.

In the first instance a ruler team is required to be in place before any workers can register with it. So, in order to assemble the system to the robust scale-free principles, it would be advantageous for the rulers to form the hub structure spine of the system as the earliest 'nodes' introduced. This is similar to the usual Leadership Election Algorithm (Francis and Saxena, 1998). Additional to this, as shown in the initial specification given above, it is also required that rulers are allowed to fail with existing rulers possessing a Joint persistent Goal to connect and take up their responsibilities with new rulers available as required. So a mechanism needs to be in place, at the small scale, to ensure new leaders can emerge, at the large scale, throughout the life of the mission. Thus the combined effect of preferential attachment, through being the first assembled nodes, and fitness, to be rulers, can be used to make up the ruler team, to an augmented Leadership Election Protocol. The nodes that become ruler nodes have a high fitness value that ensures their

position is promoted to ruler status. In the previously described formalism this is stated, for a node α as:

$fitness(\alpha, do(a,s))=F \Leftrightarrow [fitness(\alpha,s)=F \wedge a \notin A(\alpha)] \vee \exists m[fitness(\alpha,s)=F-m \wedge value(a,s)=m]$

where A(α) is the action set of component/node α and value is a function from the set of actions to the integers, mapping each action to a reward (positive integer), cost (negative integer) or no effect (0).

A simple reward system, where the main features necessary to be a ruler consist of a communicator and controller module together with the successful deployment of worker controller and location analysis software may be:

$reward(getComponent(c),s)=r \equiv c=communicator \wedge r=50 \vee c=controllermodule \wedge r=100 \vee component(c) \wedge c \neq communicator \wedge c \neq controllermodule \wedge r=10$

$reward(s\_deployFunction(f),s)=r \equiv f=workerController \wedge r=100 \vee f=locationmonitor \wedge r=50 \vee function(f) \wedge f \neq workerController \wedge f \neq locationmonitor \wedge r=20$

$reward(f\_deployFunction(f),s)=r \equiv f=workerController \wedge r=-50 \vee f=locationmonitor \wedge r=-20 \vee function(f) \wedge f \neq workerController \wedge f \neq locationmonitor \wedge r=-10$

This gives for each potential ruler:

$fitness(do(a,s)) = fitness(s) + reward(a,s)$

There are then probabilities associated with the actions. So, for instance it may be the case that:

$Prob_0(s\_deployFunction(f), deployFunction(f), s) = 0.8$

$Prob_0(f\_deployFunction(f), deployFunction(f), s) = 0.2$

$Prob_0(getComponent(c), getComponent(c), s) = 1$

This is a typical form of a Markov Decision Process (MDP), expressed in the Situation Calculus, as previously described and analysed: There is a specification of a dynamic probabilistic domain together with a reward function giving a value for each deterministic outcome of a stochastic action. Any state the system may find itself in can then be measured for fitness. (In this case via the $fitness(do(a,s)) = fitness(s) + reward(a,s)$ linear

relationship) So rulers are 'elected' based on their fitness to be rulers and the rulers are programmed to attain fitness for purpose. It is outside the scope of this paper to define what features contribute to a ruler's fitness. Rather mission scientists would need to define and specify costs and rewards for these properties. To assemble the system, so that the probability that a new worker/ruler/messenger connects to a particular existing ruler r, the formal representation of the system defines

$prob_0(s\_addLink(worker,ruler),addLink(worker,ruler),s)$ as

$$\frac{f_r d_r}{\sum_{ruler} f_{ruler} d_{ruler}}$$

where f_r is the fitness of r and d_r is its degree.

Corresponding probabilities for action failure will also apply. In this way the system is initially assembled, under observer influence, to a scale free model. In practice, however, it will be necessary for rulers to be added or built through the entire lifetime of the mission. This is where the fitness functions are crucial to enable specified craft to take on ruler duties at later stages in the mission's execution, as specified in the previous analysis of the federated behaviour.

The Pico-spacecraft, initially joining the team with few links to other craft, will rapidly acquire links if it possesses a high fitness parameter. Additionally, as discussed in Section 6.5, if the power law governing the connectivity has an exponent of 3 or less then nearly all the links would have to be removed before the network broke down (Cohen et al, 2002). Thus it would be beneficial to ascertain the exponent of the power law distribution determining this process.

This method of assembling the ruler team may be concisely portrayed in the algorithm for a team of rulers. At each time step the following two steps are performed:

(i) A new potential ruler (ie high fitness) entity is created and connects to a randomly chosen ruler.

(ii) Two randomly selected rulers are chosen. If the rulers were connected the link is removed and the two rulers are merged into one new ruler. Multiple links are removed.

If the second process is run at lower rate than the first then the team grows, as required in the initial composition stage of the mission. A fixed team is described if the processes are run together.

To illustrate: The ruler creation algorithm for the main mission may be stated as:

When a ruler fails; transfer all of its connections to an established ruler and introduce (send, build or convert) a new highly fit (for rulership) craft connected to a randomly chosen existing craft. An efficient method of maintaining and transferring ruler connections was provided in specification for federated behaviour.

The worker (less fit for ruler) craft connect to the new and existing rulers via the probability for the expression

$$prob_0(s\_addLink(worker,ruler),addLink(worker,ruler),s)$$

given above, whilst the ruler connections may be specified:

$$connections(r, do(a,s))=N \Leftrightarrow [(connections(r,s)=N) \wedge a \neq connection\_transfer] \vee$$
$$connections(r,s)=N-m \wedge \exists r'(failed(r's) \wedge connections(r's)=m)$$

$$poss(connection\_transfer,s) \Rightarrow \exists r' failed(r',s)$$

$$rulerNumber(do(a,s))=M \Leftrightarrow rulerNumber(s)=M \wedge \neg \exists r' failed(r',s) \vee$$
$$(rulerNumber(s)=M+1 \wedge a=connection\_transfer) \vee$$
$$(rulerNumber(s)=M-1 \wedge a=addcraft(r) \wedge fitness(r,s)>100)$$

So if r is the existing craft that acquires the connections of a failed unit and r' is the failed unit and r'' is the new ruler created then:

$$connections(r, do(connection\_transfer,s)) = conections(r,s)+connections(r',s) - m_{rr'} - h_{rr'}$$
$$connections(r'', do(connection\_transfer,s)) = 1 \qquad [A]$$

where $h_{rr'} = 1$ if r was connected to r' and 0 otherwise and $m_{rr'}$ is the number of craft with mutual links to r and r':

$$\sum_p h_{rp} h_{r'p}$$

Now the probability that the connection r to r' exists is:

$$\frac{connections(r,s)connections(r',s)}{rulerNumber(s) < connections(rulers,s) >}$$

where $<connections(rulers,s)>$ is the average number of connections for the ruler team. Similarly, denoting c_r for the number of connections r has in s and N for the number of rulers in s, the average number of rulers connected to r and r' is

$$< m_{rr'} > = \sum_p \frac{c_r c_p}{N < c_{rulers} >} \frac{c_p c_{r'}}{N < c_{rulers} >} \qquad [B]$$

which is $Kc_rc_{r'}$ where

$$K = \frac{< c^2_{rulers} >}{N < c_{rulers} >^2}$$

Now substituting in the probability generating function for power law distributions, using the expression given by [A] and observing that $m_{rr'}$ is a Poisson variable with its mean given by expression [B] gives

$$P(c) = \frac{1}{2}E(c^{c_r+c_{r'}-m_{rr'}-h_{rr'}-h_{r'r}}) + \frac{1}{2}c$$

$$= \frac{1}{2}\left[c + E(c^{c_r+c_{r'}}e^{Kc_rc_{r'}g(c)}(1+\frac{c_rc_{r'}g(c)}{N < c_{rulers} >}))\right]$$

where $g(c)=(1-c)/c$. Also note $h_{rr'}$ is a random bit.

Now as N→∞ K and $\frac{1}{N < c_{rulers} >} \to 0$

So letting $K^* = \frac{1}{N < c_{rulers} >}$. $P(c)$ can be expanded as a convergent power series in K and K^*. The first term when $K=K^*=0$ gives $2P(c) = P^2(c)+c$ so that

$$P(c) = 1 - \sqrt{1-c} = \frac{1}{2\Gamma(1/2)}\sum_{r=1}^{\infty}\frac{\Gamma(r-1/2)}{r!}c^r$$

with $r = c_{rulers}$.

Thus for N→∞ $P(c_{rulers}) = \frac{\Gamma(c_{rulers}-1/2)}{2\Gamma(1/2)c_{rulers}!} \sim c_{rulers}^{-3/2}$

So the connectivity of the ruler team is determined by a power law distribution with exponent 1.5. This means that the average number of connections to a ruler grows with system size so that the total number of connections grows faster than the number of team members. This, in turn, gives a greater degree of clustering, a more highly developed hub structure. Additionally the cut-off for the maximum number of connections a ruler can attain diverges with the system size. In contrast when the exponent, in the power law distribution, is greater than 2 the average number of connections is finite and the maximum number of connections settles to be of the order $\sqrt{N}$ where N is the total number of rulers. Thus the ruler team possess a high clustering coefficient indicative of the high-density core. So the average number of connections for a ruler grows with the number of Pico-spacecraft assembled for the mission. This suggests that connecting to an existing ruler is relatively inexpensive.

This specification of the ruler team assembly has been described entirely in terms of the low-level interactions of the participant component spacecraft. The above analysis has shown that this leads inevitably to a system with connectivity given by a power-law tail with exponent 1.5. In addition various properties of the system are derivable as demanded by the Knowledge Representation Hypothesis: Determining how a system behaves amounts to deducing how it must behave given the system's logical description. The Observer System applied to this scenario would thus determine the existence of scale-free connectivity exactly as shown in the implementations of Chapter 6 and be able to apply its knowledge base of monitoring and governance techniques appropriate to the system domain and topology. For instance the Acquaintance Monitoring Selection would follow in an identical manner to that shown in the implementations. Thus the observer system would not be required to exhaustively monitor the entire ruler team, rather the majority of relevant data could be gained from a selected subset of the team as determined by the algorithm and analysis.

8.3 Summary

The main contribution of this work is to provide a formal representation technique to aid the provision of self-governance and autonomic functions to any scale of system. A unifying formalism was, thus, proposed to handle both the centralised top-down control, most often seen in traditional distributed application, and the emergence of topology and function from the bottom-up, most often observed in large-scale natural and man-made systems subject to organic growth, such as the World Wide Web. Two case studies have therefore been presented in this chapter. The first, describing an adaptable decision support system for a medical application, represented a system that was entirely controllable through a centralised meta-system. The novel features of the system were evident in the use of parametric adaptation to adapt the response of the autonomic meta-system to the runtime state of the system. This is in contrast to most current implementations of autonomic systems that rely on passive adaptation. As described in Section 3.2.1 the current state of the art in autonomic systems is in parametric adaptation, where the parameters used to determine autonomic response are deduced from the run-time system and the system remains adaptable throughout runtime. The case study showed that the logical specification could be used in a traditional manner to encode the rules for deriving a treatment decision. The main use of the formalism, however, was to provide a meta-system that was capable of adapting the system as required, to clinicians

needs, for example. The system concerns analysed were in the provision of quality of process and quality of service. Quality of process was specified in the mechanisms used to improve the process decision models with extra input from clinicians deviating from expected treatment decisions. Quality of service was illustrated through the decision support service being kept available throughout periods of high demand through the monitoring of key metrics. This case study was fully implemented as part of an EPRSC funded project.

The second case study described the NASA ANTS project where very many components interact to produce global system outcomes. This represents the current state of research. At present there is little or no support for the mission-oriented adaptation, of Section 3.1.1, for reasoning on emergent behaviour in large-complex systems. The formalism was shown to be equally adept at specifying system properties and deriving behaviour through logical entailment as well as reasoning on emergent outcome in the global system. The system considered, with a specified algorithm for robustness, was mathematically proven to display a classic scale-free connectivity topology, most often displayed in large-complex, dynamic systems. In addition the topology exhibited could be ascertained to be particularly robust to random or accidental attacks. This analysis could be further utilised by the Observer System to harness the properties of the topology to implement efficient strategies in system monitoring and governance. The more generic implementation of the scale-free topology with Acquaintance Monitoring Selection, described in Chapter 6, is entirely relevant to this case study: The implementation would follow in exactly the same manner.

Thus the same formalism has been shown to work at the application level, specifying decision rules; at the level of specifying a meta-system that is centralised to maintain full governance over the system and at a level to reason on the outcome of low-level component interaction to detect new behaviour, engineer emergence and utilise known properties of prevalent emergent network topologies.

Chapter 9

Conclusions

The research work reported on in this thesis describes the analysis and design for the specification of a meta-system implementation based on a mathematical logical approach, giving system self-governance in an autonomic manner, to any scale of system through the handling of emergence both through the engineering and utilisation of previously observed instances and signal grounding. This chapter concludes the thesis with a review of the results and proposals, described in the work, set against the identified problems and challenges.

9.1 Motivations and Approach

The high performance and maintenance requirements emerging from the pervasive and ubiquitous distributed networks of computational devices and software, including grid-based applications, that are becoming evermore prevalent has led to a demand for agile and efficient meta-systems enabling system self-governance. These larger-scale and complex systems, whilst being beyond the scope of any single person's control, due to the vast amount of observational data emanating at any instance from the system, also present extreme challenges for purely computational systems in interpreting and acting on this observational data. Thus, while large amounts of data can be handled efficiently by an automated computational meta-system, additional support is required in manipulating and interpreting this data for optimum self-governance. It is only by the execution of the most appropriate actions by the self-governance meta-system, as a result of system observation, that the high assurance, dependability, availability, ease of use and managing of complexity can be achieved.

For the most part, following IBM's autonomic initiative, the autonomic meta-systems have been composed of rule sets and policies dictated for the system at design time. Such approaches, however, do not scale to larger systems or remain applicable for planet scale systems: Design time software meta-system solutions cannot cope with dynamic heterogeneous environments, where there is an absence of any centralised system, for the sharing of resources, enforcing security, maintaining effective management strategies, enabling communication between and allowing common data and system representation

among components and the comprised system. This work, therefore addresses the specification of a meta-system for self-governance in an autonomic system, which is applicable to large-scale complex systems, whilst also permitting the formal specification of smaller systems that may indeed form the components at the larger scale. A substantial portion of this work, however, necessarily concentrates on the features and monitoring techniques for larger-scale systems. In this way a unified formalism is gained whereby middleware services are specified to permit effective autonomic self-governance for the smallest sized system up to very large-scale planet wide networks. The proposal is to specify system self-governance for autonomic systems by a combination of observation and deliberative capabilities. Furthermore the application system, itself, the knowledge, observation and sensing and the deliberative capabilities can all be specified using the same formalism.

The initial impetus for the work arose from viewing system behaviour, for monitoring and influence by the meta-system, as the result of individual responses to a collectivist position. The Knowledge Representation Hypothesis naturally led to the adoption of mathematical logic as the primary formulation to account for the "...propositional account of the knowledge that the overall process exhibits". This brought up a number of theoretical and technical issues that were required to be addressed in order to further this work:

- **Emergent Self-Organisation:** The collectivist position that arises from individual actions and interactions cannot be predicted. It has been observed that in large, man-made and natural complex systems, behaviour or events emerge for the whole system that are inexplicable and thus unpredictable simply in terms of the actions executed, within the system, by its participant components.
- **Relevant Models (Autonomic Self-Governance):** Currently contrived autonomic system models do not allow for system evolution away from a design time model, as the rules governing the system are set at deployment: All states must be enumerated beforehand. The most recent research permits parametric adaptation whereby new values for threshold adjustment, for instance, may be inferred from the runtime system. There are no facilities, however, to specify autonomic response for global emergence, which requires mission-oriented adaptation. The problem identified in this work is that presently autonomic self-governance is achieved on the smaller-scale simply by symbol manipulation: The

system on observing a certain symbol or signal, comprising a particular ordered set of symbols, merely matches the symbol/signal to a pre-described set of action commands, without having any notion of the intrinsic meaning for the systems operation of the occurrence of the symbol/signal. This has been termed, through this work, as the Signal Grounding Problem.

- **Relevant Models (Formal Specification):** There is currently little or no support to facilitate the specification and/or detection of emergent self-organisation in component/agent architecture/systems. Following models based on Beliefs-Desires-Intentions (BDI) there have been many attempts to provide a formalism to adequately specify federated behaviour. These have given methods to capture the dynamics of cooperation and coordination in multi-agent systems, for example, through the notions of utility and obligations in normative settings. An associated specification of emergent outcome for the global ensemble, using the formalisms is, however, lacking.

- **Experimental Representation:** This work is concerned with the formal specification of self-governance for autonomic systems. Full details of any implementation are outside the scope of this work. Nevertheless it is necessary to evaluate and demonstrate the functions and operation of the specification in an executing environment.

Addressing these motivations and associated challenges this thesis describes specification formalism, based on a cognitive observation system, which can be applied to any scale of system. A number of research subject areas from Computing, Artificial Intelligence and Graph Theory were utilised in the proposed specification procedure:

- **Federated Behaviour:** Independent of any specific formalism the mechanism for joint intentions in a teamwork setting was investigated, as a joint intention involves more than each team member possessing the intention. Rather the specification must be based on logic with the modal operators of Mutual Belief, General Knowledge and Common Knowledge.

- **The Autonomic Computing Model:** The Autonomic computing model is used to design and develop the specification technique. In particular, further work, to advance the model, is used to attribute adaptive classifications to systems based on passive, parametric or mission-oriented adaptation.

- **Distributed Artificial intelligence (DAI):** There are many formal setting for specifying concepts and deriving solutions in a distributed environment. This works seeks the utilisation of the most relevant and applicable.
- **Network Theory:** Recent discoveries relating the underlying dynamics of large-scale networks to a widely observed global arrangement with associated transitions through system types are used to harness the properties of regular, small world, random and scale-free large-scale systems.

In particular this work provided a means of specification for a self-governing autonomic meta-system that is scalable to planet wide capabilities by:

- **A Practical Specification Formalism:** Stochastic Situation Calculus is identified as possessing many desirable characteristics for specifying large-scale system self-governance:
    - System specification based on normative positions can be described as well as the deliberative capabilities necessary for the associated cognitive system in the Observer Model.
    - A logical specification means the behaviour of the system is entailed by the specification of the system, thus removing the necessity of prior state enumeration.
    - Knowledge, sensing and joint intentions, using previously defined logical models of the modal operators, are easily specified in the Situation Calculus.
    - Stochastic actions permit deliberation in partially observable environments and provide facilities to model and engineer, through parameter adjustment, naturally occurring emergent behaviour through Markov Decision problems and other DAI techniques.
    - Its applicability in dynamic environments gives conterfactual reasoning facilities and methods to address the Signal Grounding Problem.
- **An Observer Model:** The specification of the application system and the normative positions controlled by the autonomic meta-system is achieved through the same formalism that provides for the deliberation necessary to detect newly emergent signals (Signal Grounding), the specification of component level

interactions that lead to a known global outcome (engineering emergence) and the utilisation of the properties of widely prevalent network topologies, upon detection of such organisational behaviour occurring (using the properties of scale-free networks for efficient observation).

9.2 Thesis Summary

Succeeding generations of computational systems will be characterised by increasing levels of complexity and distribution, which is required to be hidden from the users. Thus middleware services must be in place to handle the system management and relieve the users of any notion of the technical or complex processes occurring in the service delivery. Hence, at a low level, rules and normative positions must be provided by a meta-system to ensure the system operates within safe limits. At a higher level, however, facilities must exist to permit the autonomic type control of the system based on its observed operation. In order to achieve this for large-scale systems, subject to emerging behaviour patterns, cognitive systems are required within the middleware resident meta-system. This thesis has, therefore presented a specification method that can be used across all types and scales of system using techniques from the related disciplines of psychology, Artificial Intelligence, robotics, mathematical logic and advanced software engineering. The detailed descriptions of the work are presented in the following manner:

- Chapter 1 provides the motivation for the work, setting up the challenges that needed to be addressed with the research hypothesis and questions, outlining the approach and applicability of the research and providing an indication of the contribution.
- Chapter 2 introduces the notion of a complex system and presents a review and background for the current procedures for looking at complex systems including the representation of the complexity in a specification. It is observed that in large interacting systems the whole system behaviour is quite often more than the sum of its component parts. This meant that a fully reductionist approach is not appropriate leading to the understanding that a model and modal logic of federated behaviour is required to consider interaction between components or agents. Additionally emergent outcome is a major feature of large complex systems; the various types of emergence seen in nature are described and the sort of emergence seen in large man-made systems is described through the example

of the World Wide Web. The proposal for a knowledge plane, for the web, is considered as being someway addressed by this work.

- Chapter 3 provides the basic background and concepts of autonomic computing; the most widely accepted method of providing self-governing meta-systems. The Signal Grounding problem is identified as a major obstacle in achieving autonomic functions in large-scale systems requiring mission-oriented adaptation. The background covers the major properties of autonomic computing standards, architectures and capabilities. The architectures, systems and agent frameworks that can provide the decision mechanisms for actions in meta-system middleware are reviewed and the state of the art is assessed against the classes of system adaptation that can be realised. It is noted that natural models of self-organisation can be incorporated into many middleware services. Most meta-systems for self-governance, as exemplified by autonomic systems, are seen to rely on passive adaptation: The system acts according to a meta-system founded on a static preconceived rule set. The more recent research allows parametric adaptation. A fundamental requirement for large-systems, however, is mission-oriented adaptation to cope with emergent outcomes necessitating the addressing of the Signal Grounding problem.

- Chapter 4 gives a literature review of mathematical formalisms for specifying systems. The two main types considered are process algebras and logical formalisms. It is discovered that the algebraic specifications require prior enumeration of the state space. In logical formalisms it is observed that the Situation Calculus allows a good representation of all the notions requiring formulisation, in particular only Situation Calculus permits counterfactual reasoning.

- Chapter 5 introduces the Observer System in relation to systems exhibiting emergent outcomes. It shows the specification of Observer System reasoning to address the Signal-Grounding problem and to engineer emergence through adjusting the parameters in a Situation Calculus representation of a Markov Decision Problem specifying the emergent outcome of the previously defined food foraging example.

- Chapter 6 further introduces properties and evolutionary features of large-scale systems and networks that can be utilised by the Observer System when such behaviour is detected. The classes of behaviour and connectivity are reviewed through a model of phase transitions leading from a regular lattice to a scale-free connected topology. A metric is proposed, the Hub Connection Density Measure, to determine what topological phase a particular system is exhibiting, permitting the deployment of the most appropriate monitoring strategy. In particular, when a scale-free topology is detected, a newly defined Acquaintance Monitoring Selection algorithm can be employed by the observation system to ensure optimum system knowledge gathering. Moreover a representation of the situation space that gives a scale-free topology can utilise similar methods to remove less important or redundant data.
- Chapter 7 gives details of the implementations based on the work in the previous chapters. Quantitative results are also presented for the simulations to validate the use of previously defined metrics and properties.
- Chapter 8 presented two case studies as evaluation of the power of the specification methods. Accordingly the specification was shown for a norm based adaptable system and for a more large-scale system where federated behaviour was specified through which further properties could be inferred by a cognitive observation system. Additionally this gave rise to scale-free connectivity being detected, giving a particularly robust construction to this application.
- Chapter 9 gives the motivations and approach, detailing the contributions, discussing the outcomes and suggesting further work.

9.3 Achievements and Contributions

This thesis has presented a procedure to aid the development of a specification for self-governance meta-systems to enable autonomic management for varying scales of system. The thesis commenced by *recognising the importance* of knowledge dynamics in large distributed groups of components or agents. This is necessary because the *meta-system* for self-governance must be able to extract meaningful knowledge from the system. Additionally it was observed that exhaustive monitoring of the components does not necessarily provide a complete picture of the knowledge contained in the system: Global properties of behaviour emerge in the global setting of the system that cannot be

explained by aggregated actions. In other words component interactions may be responsible for emergent outcome in the system. This gives rise to observable system signals which, in order to accomplish the adaptations necessary for autonomic self-governance, require a grounded definition by the system. Thus the Signal Grounding problem was identified. A major feature of this work is that the specification method proposed is equally valid for use in norm based centralised control meta-systems based on feedback and control commands as well as for meta-systems applied to decentralised distributed autonomous components in large-scale systems. The latter case includes the facilities to detect newly emergent behaviour (ground system signals), engineer, from previously observed instances of component interactions, globally emergent properties or make use of the properties of widely occurring emergent topologies in autonomic networks: Address Signal Grounding, engineer emergence and utilise the statistical mechanics of large-scale networks. This is achieved through the use of mathematical logic enabling the verification of specifications as well as providing the specification for the inferential properties for a cognitive observation system providing facilities for system autonomic self-governance. Thus this work has ranged over a number of issues, as outlined above, and made a variety of contributions, including addressing the aims and objectives stated in Section 1.4.4:

- **An identification of the Signal-Grounding problem:** It was noted that for systems to perform genuine self-governance the systems themselves must possess grounded definitions for the observable phenomena emanating from the system.
- **Proposals for addressing the Signal-Grounding problem:** A deliberative mechanism was proposed to analyse the results of action histories and assess the recurrence of observed phenomena. This was further refined to give an adjustable engineering of emergence, through obtaining the solutions of (Partially Observable) Markov Decision Problems, which was illustrated through a specification of ant food foraging.
- **The adoption of Situation Calculus for autonomic systems specification:** The Situation Calculus was shown to possess many useful characteristics for the specification of self-governance meta-systems. In particular

- A specification of federated behaviour was given in the Situation Calculus based on an introduced concept of the modal logic of mutual knowledge.
- A treatment of stochastic actions in Stochastic Situation Calculus allows support for the specification of self-governance meta-systems in partially observable domains subject to uncertainty.
- Sensing and knowledge manipulation for deliberation, based on the central notion of action histories, provides means to address the identified Signal Grounding problem and harness some often occurring generic properties of large-scale complex systems.
- The Stochastic Situation Calculus is a very natural specification medium for Markov Decision Problems allowing deliberative observers to solve the problems and thus optimise the system.

- **A unifying specification procedure for autonomic self-governance:** The work presented in this thesis gives methods to support autonomic software engineering to give self-governance to software systems. This is achieved through a unifying formalism, based on Situation Calculus, that gives the specification for a deliberative observation system that can represent the norm based control of centralised command and the deliberative properties required for the monitoring influence and induced self-* capabilities of large planetary scale systems: The (Stochastic) Situation Calculus permits the specification of:
  - Bounded autonomy, specifying individual component system norms of any form.
  - Interactive social behaviour, a model of social federated behaviour, for teamwork solutions, was proposed in a form suitable to be specified in Situation Calculus.
  - Deliberation, it was shown how the Situation Calculus specification allows the cognitive facilities to detect newly emergent outcomes based on grounding signals, engineer emergence from previously derived models and utilise the features, such as scale-free connectivity, associated with the evolution of large-scale networks.

- **An assessment of the evolution and properties of large-scale networks:** It was shown how large-scale networks maybe transformed, through a phase transition process from a regular to a small world to a random to a scale-free network. This led to the proposal of the Hub Connection Density measure to assess the phase a particular system maybe in, influencing the chosen monitoring strategy.
- **A postulated means by which stored system knowledge may be reduced without global system awareness:** Based on the scale-free model an initial proposal was outlined to identify important data in the Situation Space without necessarily having global knowledge of the content. In this way it is possible to remove less crucial data from the Situation Space, improving the speed and efficiency of deliberation.
- **An implementation/simulation:** The implementations were reported upon together with fully implemented simulations quantitatively showing the efficacy of the deliberative procedure, for the Observation System, to choose the optimum monitoring points. It was shown, through the simulation results, that a significant proportion of system knowledge was returned by such techniques; much more than would be gained through any other currently available scalable method.
- **An evaluation:** The specification technique was evaluated through two case studies, showing the proof of concept, the ease by which a specification may be achieved and the deliberative processes that may occur within the Observation System. The applications considered, to demonstrate the approach to all system scales, firstly showed a relatively straightforward, norm based meta-system. The second application showed the model of federated behaviour in operation leading to observable and analysed emergent outcome. The applications presented were:
    - o A decision support system for post-operative breast cancer care. The specification was shown working through rule representation and the specification of meta-system concerns involving quality of service and quality of process issues. This equipped the system with facilities to handle demand spikes and to adapt to clinicians requirements whilst allowing the assimilation of novel treatments and treatment combinations. Although full details of the achieved implementation are outside the

scope of this work, a fully functional prototype was produced as part of the EPSRC funded project.

- The NASA ANTS space mission. The interactions of the numerous Pico-spacecraft were firstly modelled using the Situation Calculus specification of federated behaviour. It was shown how the Observation System's deliberation would proceed to infer extra knowledge about the system, demonstrating that the approach does not require a prior complete state/situation enumeration. The further analysis showed that scale-free behaviour results from the described topology, thus allowing customised observation system techniques to be applied.

9.4 Conclusion and Discussion

The purpose of this work, as summed up in the stated research hypothesis, is to provide a method, applicable to varying sizes and complexities, for the specification of self-governance giving the system autonomic management. Typically, in presently deployed systems, this is achieved by the prescription of a set of rules stipulating the action that ought to occur for a specific set of observed circumstances. It can readily be seen that such approaches do not scale for two main reasons: Firstly the behaviour as systems become larger and more interactive, quickly outstrips the representative capacities of finite rule sets and secondly such an approach demands that all behaviour must be predictable at design time so a rule can be formulated, this is obviously infeasible, particularly given circumstances of emergent behaviour. Thus it is postulated that the best way to achieve self-governance over a system is to combine the data handling capacities of a computational system with the cognitive reasoning techniques of an observer. Furthermore the observer is required to be a separated software component to enable deliberation for self-governance. This leads to the central tenet for the motivation of this research that there is very little formal support for the specification of autonomic systems or for self-governance in general. Furthermore mechanisms to enable the necessary deliberation to achieve this are fragmented between a relatively wide variety of centralised command and control type architectures and a very few offering support for the increasingly prevalent decentralised architectures subject to self-organisation and emergence, capable of confronting the Signal-Grounding problem.

Deliberation, in centralised control meta-systems, has been extensively investigated through extensions to BDI, as described in Section 3.2 whilst deliberation in a decentralised architecture has received less attention. Some approaches have tended to investigate the emergence of component roles in such systems (Hales and Edmonds, 2003) whilst others seek merely to define emergence (Gillett, 2002). The emergent outcome of large-complex systems, seen as a multi-agent system, usually follows a biological inspired approach (Mano et al, 2006) or focuses on the social interactions of the participants (Watts and Strogatz, 1998). The most usual approach seeks to emulate previously observed instances of naturally occurring emergence by re-enacting the low-level component interactions that presaged the exhibited global outcome by engineering emergence (Zambonelli et al, 2004). The algorithms proposed for engineering emergence provide the assurance that the system will converge to the eventual predicted global state that is stable to perturbation from further low-level interactions. Stigmergy (Grasse, 1959) is one such technique that has aroused some interest in this regard. It is based on the indirect interactions of system participants, where messages are deposited through the environment to be picked up by subsequent encountering entities. It has been applied to a number of practical problems including manufacturing control scenarios control (Hadeli et al, 2004) and agent meditated security (Foukia, 2005). There is little support, however, *for a related general formal* specification and there is no proof of convergences to the required emergent state. In the Observer System, *proposed in this work*, signal-grounding may be achieved through the establishment of a proposed grounding assessed for accuracy and relevance by the future recurrences of the signal: Behaviour is adapted according to reinforcement. The model of adaptive agents (Weyns et al, 2004) dynamically adapts logical relations between different behaviours in a manner that can be utilised so that the observer system may monitor the recurrence of some recognised emergence. Cooperation at the component level is frequently used to engender desired collective behaviour, emerging to provide the system's functionality. AMAS theory (Gleizes et al, 1999) specifies that each cooperative agent is able to rearrange its local interactions dependent on its knowledge of the emerging system function. It is also possible to model systems based on meta-models of agent organisation. The PROSA architecture (Bongaerts, 1998) for example, involves a holonic hierarchy model. Agents participate in holons, forming holonic structures with self-organisation occurring by adapting the holonic hierarchy to environmental perturbations. A model where there is

direct interaction between system participants allows the engineering of emergence through previously observed outcomes; this is only useful, however, for simple global equilibrium states modelled in a strictly linear manner (Zambonelli, 2004). Stigmergy type scenarios have the added benefit of giving an implementation from an observed calibrated simulation. This gives some ideal solutions for specific application domains but, as mentioned earlier, does not permit a general formal explanation. Cooperation behaviour requires an exhaustive enumeration of cooperative states and adaptations, which is not always possible for large systems. Additionally the formal setting for such systems is allied to the system classification. Thus there are separate specifications for norm-based operation, cooperative, coordinated or socially interactive systems and for situations where emergent self-organisation may occur. This thesis has proposed a method of specification, for a self-governance system giving autonomic response, suitable for all types and scales of system. The proposal for use of the Situation Calculus, augmented with the modalities suggested in Chapter 4, gives the basis for specification formalism for autonomic self-governance appropriate to all scales of systems. The next section presents a number of areas where further research may populate the observational systems with additional functionality to better deliberate over the system state, restrict their deliberations to the more relevant system data and further address the Signal-Grounding problem. The Situation Calculus remains as a powerful specification medium through which to represent the further development of these ideas. Indeed the tools and techniques established through this work can be applied to many other areas of application, other than autonomic middleware. For instance smart devices to aid the disabled would gain in robustness and versatility with the addition of cognitive facilities. In practice any applications that call for large data set manipulation, such as GPS information for road pricing, can be better handled by the establishment of the type of systems proposed throughout this thesis.

9.5 Proposed Further Works

The work presented in this thesis is intended to advance the notion that a specification of self-governance for autonomic systems is achievable for any scale of system using a single formal specification method. This has been demonstrated within this work. The deliberation, however, requires content or monitoring data to reason over and established models to reason against.

- Further work is required to refine and develop new models of autonomic networks. The transitions through regular, small world, random and scale free topologies are characterised by wide limits in the low-level parametric specification and the global measures. More fine-grained models may lead to better use of more targeted monitoring resources. Indeed alternative models of emergent topological outcome can provide extra deliberative resources for the meta-systems.

- The accumulation of monitoring data, such as complete action histories, is very difficult to regulate and utilise efficiently. As the system develops more and more data needs to be retained and the associated searches, through the data, become more laborious. A method has been proposed in this work to trim the situation space by ensuring it arrangement into a scale-free system and using that topology, in a similar way to acquaintance monitoring/immunisation, to remove some stored action histories (situations) without complete global knowledge of the system. Although initial results are promising much more work is required in this area to assess the best way in which the situation space may be represented for trimming and how often the trimming may occur.

- The specification of the self-governance does not account for the timeliness of the specified deliberation. In practice the results of deliberation are always going to lag behind the actual system's state. In the implemented simulations, described in Chapter 7, the deliberation required within the system caused many long delays in constructing the systems from the component interactions, gathering data and executing algorithms.

- Further work is needed in extracting the most timely and greatest volume of data from the system. One approach is to investigate the use of a *connected dominating set* of nodes to delegate the monitoring tasks to a subset of nodes in the system. These nodes are called supervisor nodes. Each supervisor has to monitor a limited number of nodes called supervised nodes. In order to monitor the entire system, the nodes of the network must be either supervisor or supervised nodes. Further investigation can improve the robustness of this system by constructing a *k-connected dominating set*, where each supervisor node possesses, in effect, k replacements. This all requires further analysis and research to incorporate these

techniques into autonomic system monitoring, including the optimization of the data flow between nodes.

- The Signal-Grounding problem is still relevant and improved methods are required to address the timely runtime detection of emergence. An indication of one future direction of this work is detailed in the workshop paper reprinted in Appendix 2, where the storage of situations or action histories is used to ground a particular system signal. All action histories are stored in the system as individual objects, which are subject to trimming by the specified process. This work, however, still requires a proper evaluation to determine suitable parameters to maintain maximum knowledge with minimal storage.

References

2nrich Project (2006) Towards a Disciplined Approach to Integrating Decision-Support Systems for Breast Cancer Care Activities. http://www.cms.livjm.ac.uk/2nrich/

Aicklen U., P. Bentley, S. Cayzer, J. Kim, J. McLeod (2003) Danger Theory: The Link Between AIS and IDS? In J. Timmis, P. Bentley, E. Hart (editors) Springer LNCS 2787 pp: 156-167.

Aiello, W., F. Chung, and L. Lu (2000) A Random Graph Model for Massive Graphs. In Proceedings of the 32nd Annual ACM Symposium on Theory of Computing (STOC), pp: 171-180. ACM Press.

Albert, R., R. H. Jeong, A.-L. Barabasi (1999) Internet: Diameter of the World-Wide Web. Nature 401, pp: 130.

Albert, R., A.-L. Barabasi (2002) Statistical Mechanics of Complex Networks. Reviews of Modern Physics 74, pp: 47-98

Albrecht, C.C., D. D. Dean, J. V. Hansen (2003) Using Situation Calculus for E-business Agents. Expert Systems with Applications, 24(4), pp: 391-397.

Alvarez-Ramirez, J., C. Ibarra-Valdez, E. Rodriguez, R. Urrea (2007) Fractality and Time Correlation in Contemporary War. Choas, Soliton and Fractals 34(4), pp: 1039-1049.

Andrzejak A., S. Graupner, V. Kotov, H. Trinks, (2002) Algorithms for Self-Organization and Adaptive Service Placement in Dynamic Distributed Systems, Tech. Rep. HPL-2002- 259, Hewlett-Packard Labs Palo Alto.

Appavoo J., K. Hui, C. A. N. Soules, R. W. Wisniewski, D. M. Da Silva, O. Krieger, M. A. Auslander, D. J. Edelsohn, B. Gamsa, G. R. Ganger, P. McKenney, M. Ostrowski, B. Rosenburg, M. Stumm, and J. Xenidis (2003) Enabling Autonomic Behaviour in Systems Software with Hot Swapping. IBM Syst. J., 42(1) pp: 60-76.

Arnold K., A. Wollrath, B. O'Sullivan, R. Scheifler, J. Waldo (1999) The Jini Specification. Addison-Wesley, Reading, MA, USA.

Ashby, W. R. (1947) Principles of the Self-Organizing Dynamic System. Journal of General Psychology (1947), volume 37, pp: 125-128

Badr N., A. Taleb-Bendiab, M. Randles, D. Reilly (2004) A Deliberative Model for Self-Adaptation Middleware Using Architectural Dependency. In Proceedings of the 15th International Workshop on Database and Expert Systems Applications (DEXA'04), SAACS'04: 2nd International Workshop on Self-Adaptable and Autonomic Computing Systems, pp: 752-756.

Baeten. J.C.M. (1993) The Total Order Assumption. In S. Purushothaman and A. Zwarico, editors, Proceedings First North American Process Algebra Workshop, Workshops in Computing, pp: 231–240. Springer Verlag.

Baillie, J.C. (2004) Grounding Symbols in Perception with Two Interacting Autonomous Robots. Proceedings of 4th Intl. Workshop on Epigenetic Robotics: Modelling Cognitive Development in Robotic Systems 117 pp: 107-110.

Barabási, A.-L. (2002) Linked: The New Science of Networks, Perseus Publishing.

Barabási, A.-L. and R. Albert (1999), Emergence of Scaling in Random Networks, Science 286, pp.509–512.

Baresi, L., M. Baumgarten, M. Mulvenna, C. Nugent, K. Curran, P. Deussen (2006) Towards Pervasive Supervision for Autonomic Systems. In Proceedings of IEEE Workshop on Distributed and Intelligent Systems, Prague, pp: 365-370.

Becht, M., T. Gurzki, J. Klarmann, M. Muscholl, (1999) ROPE: Role Oriented Programming Environment for Multi-agent Systems. Proceedings of the Fourth IFCIS Conference on Cooperative Information Systems (CoopIS'99), Edinburgh, Scotland, September 1999.

Beer, S. (1979) The Heart of the Enterprise. John Wiley & Sons, Chichester, UK.

Belleghem, van K., M. Denecker, D. De Schreye (1997) On the Relation Between Situation Calculus and Event Calculus. The Journal of Logic Programming 31(1-3), pp: 3-37

Bellman R. E. (1957) Dynamic Programming. Princeton University Press, Princeton, USA.

Beneken, G., U. Hammerschall, M. Broy, M.V. Cengarle, J. Jürjens, A. Pretschner, B. Rumpe, M. Schoenmakers (2003) Componentware State of the Art 2003 Background Paper for the Understanding Components Workshop of the CUE Initiative at the Univerita Ca Foscari di Venezia Venice, October 7th-9th 2003.

Bergstra, J. A. and J. W. Klop (1984). Process Algebra for Synchronous Communication. Information and Control 60(1/3), pp: 109–137.

Bernon, C., V. Chevrier, V. Hilaire, P. Marrow (2006). Applications of Self- Organising Multi-Agent Systems: An Initial Framework for Comparison. Informatica 30(1), http://ai.ijs.si/informatica/PDF/30-1/06_Carolle-Applications%20of%20Self-Organising%20Multi-Agent...pdf (Accessed April, 2007)

Boella G., L. van der Torre (2003) Permissions and Obligations in Hierarchical Normative Systems. Proceedings of the Eighth International Conference on Artificial Intelligence and Law (ICAIL'03), pp: 109-118.

Bollobás,, B. (1985) Random Graphs, Academic Press, NY,

Bonabeau E., M. Dorigo, and G. Theraulaz (1999) Swarm Intelligence: From Natural to Artificial Systems. Santa Fe Institute Studies on the Sciences of Complexity. Oxford University Press, UK.

Bongaerts L. (1998) Integration of Scheduling and Control in Holonic Manufacturing Systems. PhD Thesis, Katholieke Universiteit, Leuven.

Booch, G. (1994) Object-Oriented Analysis and Design with Applications. Addison Wesley.

Boman, M. (1999) Norms in Artificial Decision Making. AI and Law 7(1), pp: 17-35.

Boutilier, C., T. Dean, and S. Hanks (1999) Decision Theoretic Planning: Structural Assumptions and Computational Leverage. Journal of Artificial Intelligence Research11. pp: 1–94.

Boutilier, C., R. Reiter, R. Price (2001) Symbolic Dynamic Programming for First-order MDPs. In Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence (IJCAI-01), Seattle, pp: 690—697.

Bratman, M.E. (1987) Intentions, Plans, and Practical Reason. Harvard University Press: Cambridge, MA.

Bratman, M. E., D.J. Israel, M.E. Pollack (1988). Plans and Resource-Bounded Practical Reasoning. Computational Intelligence 4 (4), pp: 349-355.

Broersen, J., M. Dastani, J. Hulstijn, Z. Huang and L. van der Torre (2001) The Boid Architecture. In Proceedings of the Fifth International Conference on Autonomous Agents (Agents'01), Montreal, Quebec, Canada, pp: 9-16.

Brooks, F.P. (1995) The Mythical Man-Month. Addison Wesley.

Brooks, R.A. (1991) Intelligence Without Representation. Artificial Intelligence 47 pp: 139-159

Bourjot, C., V. Chevrier, V. Thomas (2003). A New Swarm Mechanism based on Social Spider Colonies: from Web Weaving to Region Detection. Web Intelligence and Agent Systems. 1(1), pp: 47-64.

Bullock, S., D.Cliff (2004) Complexity and Emergent Behaviour in ICT Systems. Technical Report HP-2004-187, Semantic & Adaptive Systems, Hewlett-Packard Labs. Available from: http://www.hpl.hp.com/techreports/2004/HPL-2004-187.pdf (Accessed May, 2007).

Cardelli, L., A.D. Gordon (1998). Mobile Ambients. In Proceedings of the First international Conference on Foundations of Software Science and Computation Structure M. Nivat, Ed. Lecture Notes In Computer Science 1378 pp: 140–155.

Camazine, S., J.-L. Deneubourg, N. Franks, J. Sneyd, G. Theraula, E. Bonabeau (2003) Self Organization in Biological Systems, Princeton Univ Press.

Cerone, A., G.J. Milne (2005) Property Verification of Asynchronous Systems. Innovations in Systems and Software Engineering 1(1), pp: 25-40, Springer-Verlag London, UK.

Clark, D. D., C. Partridge, J. Christopher Ramming, John T. Wroclawski (2003) A Knowledge Plane for the Internet. In Proceedings of Special Interest Group on Data Communications (SIGCOMM'03) Karlsruhe, Germany, pp: 3-11

Chomsky, N. (1965) Cartesian Linguistics. Reprinted in Cartesian Linguistics. A Chapter in the History of Rationalist Thought. Lanham, Maryland: University Press of America, 1986.

Cohen, P.R. and H. J. Levesque (1990). Intention is Choice with Commitment. *Artificial Intelligence*, 42 pp: 213–261.

Cohen, R., S Havlin, D ben-Avraham (2002)Structural Properties of Scale Free Networks, S. Bornholdt and H. G. Schuster(eds) Chap. 4 in Handbook of Graphs and Networks, Wiley-VCH.

Cohen, R., S. Havlin, D. ben-Avraham (2003) Efficient Immunization Strategies for Computer Networks and Populations,"Physical. Review Letters 91(24), 247901.

Conte, R., C. Castelfranchi, (1995). Understanding the Effects of Norms in Social Groups through Simulation. In Artificial societies: the computer simulation of social life. Eds. G.N. Gilbert and R. Conte, London, UCL Press.

Deussen P.H (2006) Supervision of Autonomic Systems. International Transactions on Systems Science and Applications 2(1), pp: 105-110.

Dawkins, R. (1996) The Blind Watchmaker. W. W. Norton & Co. New York

Denecker, M., D. De Schreye (1995) Representing Incomplete Knowledge in Abductive Logic Programming. Journal of Logic and Computation 5(5), pp: 553-577.

Denecker, M., E. Ternovska (2007) Inductive Situation Calculus. Artificial Intelligence 171(5-6), pp: 332-360

Dignum F. (1999) Autonomous Agents with Norms. Artificial Intelligence and Law 7(1) pp: 69–79.

Dignum, V., J.J. Meyer, F. Dignum, H. Weigand (2003) Formal Specification of Interaction in Agent Societies. In: M. Hinchey, J. Rash, W. Truszkowski, C. Rouff, D. Gordon-Spears (Eds.): Formal Approaches to Agent-Based Systems (FAABS), Lecture Notes in Artificial Intelligence, Springer-Verlag, Volume 2699.

Dong, W., K. Xu, M. Lin (2004) A Situation Calculus-based Approach To Model Ubiquitous Information Services. Computing Research Repository (CoRR) cs.AI/0311052. Available at: http://arxiv.org/abs/cs.AI/0311052 (Last Accessed May, 2007)

Erdös, P.,A. Rényi (1959). On random graphs I. Publ. Math. Debrecen 6, pp: 290–291.

Erdös, P.,A. Rényi (1961). On the Evolution of Random Graphs. Bull. Inst. International Statistics. 38, pp: 343-347.

Fagin, R. and J. Y. Halpern (1988). Belief, Awareness, and Limited Reasoning. Artificial Intelligence 34, pp: 39–76.

Fagin, R., J.Y. Halpern, Y. Moses, M. Vardi (1995) Reasoning about Knowledge. MIT Press, Cambridge, MA, USA.

Faloutsos M., P. Faloutsos, C. Faloutsos (1999) On Power Law Relationships of the Internet Topology. ACM SIGCOMM Computer Communication Review 29(4), pp: 251-262.

Ferber J., O. Gutknecht (1998) AALAADIN: A Meta-Model for the Analysis and Design of Organizations in Multi-Agent Systems. In Proceedings of the Third International Conference on Multi-Agent Systems (ICMAS), Cite des Sciences - La Villette, Paris, France, July 1998.

Feurzeig, W., S. Papert (1968) Programming Languages as a Conceptual Framework for Teaching Mathematics. In Proceedings of NATO Science Conference on Computers and Learning, pp: 37-42.

Filipe, J., K. Liu, (2000) The EDA Model: An Organisational Semiotics Perspective to Norm based Agent Design. Proceedings of Workshop on Norms and Institutions in Multi-Agent Systems, Barcelona, Spain.

Fikes, R.E., N.J. Nilsson (1971) STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving. Artificial Intelligence 2(3-4), pp: 189-208

Finger, J. (1986) Exploiting Constraints in Design Synthesis. PhD. Thesis, Stanford University, Stanford, CA, USA.

FIPA (2003) Agent Communicative Act Library Specification. http://www.fipa.org

Floridi, L. (2004) Open Problems in the Philosophy of Information, Metaphilosophy 35, pp:554-582,

Floyd, S. (2007) Adaptive Web Cache, http://www.icir.org/floyd/web.html (Accessed April, 2007)

Francis, P., S. Saxena (1998) Optimal Distributed Leader Election Algorithm for Synchronous Complete Network. In Proceeding of the International Conference on Global Connectivity in Energy, Computer, Communication and Control (TENCON '98), pp: 86-88.

Foukia N. (2005) IDReAM: Intrusion Detection and Response executed with Agent Mobility. In the Proceedings of the International Conference on Autonomous Agents and

Multi-Agent Systems (AAMAS'05), pp: 264-270, Utrecht, The Netherlands.

Fox, J., N. Johns, A. Rahmanzadeh, R. Thompson (1996) Proforma: a Method and Language for Specifying Clinical Guidelines and Protocols. In J Brender, J P Christensen, J-R Scherrer P McNair (editors) Medical Informatics Europe '96 IOS Press, Amsterdam, pp: 516-520.

Fruchterman, T.M.J., E.M. Reingold (1991) Graph Drawing by Force-Directed Placement. Software-Practice and Experience 21(11), pp: 1129-1164.

G. T. Ltd. (2002) GigaSpaces Platform, White Paper (February 2002).

Galli, A., A. Signorini (2005) The Indexable Web is more than 11.5 Billion Pages. In proceedings of the 14th International World Wide Web Conference (WWW'05) pp: 902-904.

Gamma, E., R. Helm, R. Johnson, J. Vlissides (1994) Design Patterns: Elements of Reusable Object-Oriented Software. Addison Wesley.

Ganek, A.G., T.A. Corbi (2003) The Dawning of the Autonomic Computing Era. IBM Systems Journal, 42(1) pp: 5-18,

Garlan, D., S. Cheng, A. Huang, B. Schmerl, P. Stennkiste (2004) Rainbow: Architecture Based Adaptation with Reusable Infrastructure. IEEE Computer 37(10), pp: 46-54

Gasser, L. (1991) Social Conceptions of Knowledge and Action: Distributed Artificial Intelligence and Open Systems Semantics *Artificial Intelligence*, January/February 1991. Reprinted in Michael N. Huhns and Munindar P. Singh (eds). Readings in Agents, Morgan Kaufmann, 1997.

Gelernter D., N.Carriero (1992) Coordination Languages and Their Significance, Communications of the ACM 35 (2) pp: 96–107.

Genesereth, M. R., S.P. Ketchpel (1994) Software Agents. Communications of the ACM 37 pp: 48-53

Gilbert Nigel, Matthijs den Besten, Akos Bontovics, Bart G.W. Craenen, Federico Divina, A.E. Eiben, Robert Griffioen, Gyorgy Hévízi, Andras Lõrincz, Ben Paechter, Stephan Schuster, Martijn C. Schut, Christian Tzolov, Paul Vogt and Lu Yang (2006) Emerging Artificial Societies Through Learning. Journal of Artificial Societies and Social Simulation 9(2), pp: 9.

Gillett C. (2002) The Varieties of Emergence: Their Purposes, Obligations and Importance. Grazer Philosophische Studien 65 pp: 89-115.

Gilmore S., J. Hillston (1994) The PEPA Workbench: A Tool to Support a Process Algebra-based Approach to Performance Modelling. In Proceedings of the Seventh International Conference on Modelling Techniques and Tools for Computer Performance Evaluation, number 794 in Lecture Notes in Computer Science, Springer-Verlag pp: 353-

368.

Glansdorff, P., I. Prigogine (1978) Thermodynamic Study of Structure, Stability and Fluctuations Wiley New York 1978

Gleizes M.-P., V. Camps, and P. Glize (1999) A Theory of Emergent Computation Based on Cooperative Self- Organisation for Adaptive Artificial Systems. Fourth European Congress of Systems Science. Valencia, 1999.

Gödel, K. (1931) Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme, I. Monatshefte für Mathematik und Physik 38: 173-98. In On Formally Undecidable Propositions Of Principia Mathematica And Related Systems, tr. B. Meltzer, Basic Books New York 1962.

Gordon, M.J.C., T.F. Melham (1993) An Introduction to HOL-a theorem Proving Environment for Higher Order Logic. Cambridge University Press.

Grassé Pierre-Paul (1959) La Reconstruction du nid et les Coordinations Inter-Individuelles chez Bellicositermes Natalensis et Cubitermes sp. La théorie de la Stigmergie: Essai d'interpretation du Comportement des Termites Constructeurs. Insectes Sociaux, 6, pp: 41–81.

Green C. (1969) Application of Theorem Proving to Problem Solving. In Proceedings of the First International Joint Conference on Artificial Intelligence (IJCAI-69) pp: 219-239

Grosz, B., C. Sidner (1990) Plans for Discourse. In P.R. Cohen, J. Morgan and M.E. Pollack (Editors) Intentions in Communication, MIT Press, Cambridge, MA, USA

HP World (2003) Adaptive Infrastructure, Atlanta, Georgia, 11-15th August, 2003

Hadeli, P. Valckenaers, M. Kollingbaum, H. van Brussel (2004) Multi-Agent Coordination and Control using Stigmergy Computers in Industry 53(1) pp 75-96.

Hadeli, P. Valckenaers, B. Saint-Germain, P. Verstraete, C. B. Zamfirescu, H. Van Brussels (2005) Emergent Forecasting using a Stigmergy Approach in Manufacturing Coordination and Control. Engineering Self-Organising Systems. S. Brueckner et al. (Eds), Lecture Notes in Artificial Intelligence, volume 3464, pp: 210-226, Springer-Verlag, Berlin, 2005.

Hales D., B. Edmonds (2003) Evolving Social Rationality for MAS using 'Tags', In Proceedings of the 2nd Int. Joint Conf. on Autonomous Agents and Multi-Agent Systems (AAMAS 2003). Melbourne, Australia: ACM Press, pp: 497-503.

Halpern, J.Y., Y. Moses (1990) Knowledge and Common Knowledge in a Distributed Environment Journal of the ACM 37(3) pp: 549-587

Halpern, J.Y., Y.Moses (1994) A Guide to Completeness and Complexity for Modal Logics of Knowledge and Belief Reprinted from Artificial Intelligence 54 1992 pp:311-379

Halpern, J.Y., L.C. Rego (2006) Reasoning about Knowledge of Unawareness. In Proceedings of Tenth International Conference on Principles of Knowledge Representation and Reasoning (KR 2006), pp:14-24

Harel D. (1979) First-Order Dynamic Logic. Lecture Notes in Computer Science, 68, *Springer-Verlag, New York.*

Harnad, S. (1990) The Symbol Grounding Problem. Physica D, pp: 335-346.

Hewitt, C. (1990) Towards Open Information Systems Semantics. In the proceedings of 10th International Workshop on Distributed Artificial Intelligence. October 23–27, 1990. Bandera, Texas.

Hewitt, C. (1991) Open Information Systems Semantics. Journal of Artificial Intelligence. January 1991.

Hintikka, J. (1962). Knowledge and Belief. Ithaca, N.Y.: Cornell University Press.

Hoare, C. A. R. (1978). Communicating Sequential Processes. Communications of the ACM 21 (8), pp: 666–677.

Hoare C. A. R. (1985). Communicating Sequential Processes. Prentice-Hall.

Holland J.H. (1995) Hidden Order: How Adaptation Builds Complexity. Addison-Wesley, Redwood City, CA, USA

Horn P. (2001) Autonomic Computing: IBM Perspective on the State of Information Technology. Presented at AGENDA 2001, Scottsdale. IBM, T.J. Watson Labs, New York. Available from: http://www.research.ibm.com/autonomic/manifesto/autonomic_computing.pdf (Accessed April, 2007)

IBM (2003) An Architectural Blueprint for Autonomic Computing. http://www-03.ibm.com/autonomic/pdfs/ACBP2_2004-10-04.pdf (Accessed April, 2007)

IBM and Cisco Systems (2003) Adaptive Services Framework. White Paper, October, 2003

Jenning, N. R. (1995) Controlling Cooperative Problem Solving in Industrial Multi-Agent Systems using Joint Intentions Artificial Intelligence. 75(2) pp: 195-240.

Jennings, N. R. (2001) An Agent-Based Approach for Building Complex Software Systems. Communications of the ACM 44(4) pp: 35-41.

Jennings, N.R., M. Wooldridge Editors (1998) Agent Technology: Foundations, Applications and Markets. Springer Verlag, Berlin

Jeong, H., B. Tombor, R. Albert, Z.N. Oltvai, A.L. Barabasi (2000). The Large-Scale Organization of Metabolic Networks. Nature 407, pp: 651–654.

Jones A., M. Sergot (1993) On the Characterisation of Law and Computer Systems: the Normative Systems Perspective. In Deontic Logic in Computer Science: Normative System Specification, pp: 275–307. J. Wiley and Sons.

Kawachi, Y., K. Murata, S. Yoshii, Y. Kakazu (2004) The Structural Phase Transition Among Fixed Cardinal Networks. Proceedings of the 7th. Asia-Pacific Conference on Complex Systems, Cairns Australia, pp: 247-255.

Kephart. J.O., D .M Chess (2003), The Vision of Autonomic Computing. Computing Vol. 36 No. 1, 2003 pp.41-52.

Kiczales, G., J. Lamping, A. Mendhekar, C. Maeda, C. Videira Lopes, J.-M.Loingtier, and J. Irwin (1997), Aspect-Oriented Programming In Proc. of ECOOP 1997.

Klemm, K., V.M. Eguíluz (2002) Highly Clustered Scale-Free Networks. Physical Review E65, 036123, arXiv:cond-mat/0107606v1.

Kowalski, R. A., F. Sadri (1997). Reconciling the Event Calculus with the Situation Calculus. Journal of Logic Programming 31 (1–3), pp: 39–58.

Kowalski, R.A., M. Sergot (1986) A Logic Based Calculus of Events. New Generation Computing 4(4), pp: 319-340

Kripke, S. (1963) Semantical Analysis of Modal Logic I: Normal Modal Propositional Calculi, Zeitschrift für Mathematische Logik und Grundlagen der Mathematik 9 pp: 67–96

Kuo, B.C., F. Golnaraghi (2003) Automatic Control Systems. 8th Edition, Wiley New York.

Labrou, Y., T. Finin (1997) A semantics Approach for KQML: A General Purpose Communication Language for Software Agents. Proceeding of the Third International Conference on Information and Knowledge Management (CIKM'94) Gaithersburg, Maryland, USA.

Lawrence, S. and C. L. Giles (1999) Accessibility and Distribution of Information on the Web Nature 400, pp: 107-109.

Laws, A., A. Taleb-Bendiab, S. Wade, D. Reilly (2003) From Wetware to Software: A Cybernetic Perspective of Self-adaptive Software. Lecture Notes in Computer Science 2614, pp: 341-357.

Lesperance, Y., H. Levesque, F. Lin, D. Marcu, R. Reiter, R. Scherl (1994) A Logical Approach to High-Level Robot Programming-A Progress Report. In B. Kuipers (Ed.) Control Of the Physical World by Intelligent Systems: Papers from the 1994 AAAI Fall Symposium, pp: 79-85, AAAI Press, Menlo Park, CA, USA.

Lesperance, Y., H. Levesque, R. Reiter (1997) A Situation Calculus Approach to Modeling and Programming Agents. In A. Rao, M. Wooldridge (Eds.) Foundations and Theories of Rational Agency, Kluwer Academic Press, New York, USA.

Levesque, H. (1986), Making Believers out of Computers, Artificial Intelligence 30 pp: 81-108

Levesque, H. J., P.R. Cohen, J.H.T. Nunes (1990). On acting together. In Proceedings of the Eighth National Conference on Artificial Intelligence (AAAI-90), pages 94–99, Boston, MA.

Levesque, H. J., F. Pirri and R. Reiter (1998) 'Foundations for the Situation Calculus'. Linköping Electronic Articles in Computer and Information Science, http://www.ep.liu.se/ea/cis/1998/018/

Levy, F., Quantz, J. J. (1998). Representing Beliefs in a Situated Event Calculus. In H. Prade (Ed.), Proceedings of the Thirteenth European Conference on Artificial Intelligence pp: 547–551, John Wiley, Chichester, UK.

Li, L., D. Alderson, J.C. Doyle, W. Willinger (2005) Towards a Theory of Scale-Free Graphs: Definition, Properties and Implications. Internet Mathematics 2(4), pp: 431-523.

Lin, F., R. Reiter (1994) How to Progress a Database (and Why) I: Formal Foundations. In J. Doyle, E. Sandwall, P. Torasso (Eds.) Proceedings of 4th International Conference on Principles of Knowledge Representation and Reasoning (KR'94), pp: 425-436.

Lindahl, L. (1992) Stig Kanger's Theory of Rights. In 9th International Congress of Logic, Methodology and Philosophy of Science. Stig Kanger Memorial Symposium on the Logic of Rights and Choices, Uppsala.

Litzkow, M.J., M. Livny, M. W. Mutka (1988) Condor: A Hunter of Idle Workstations, In: Proceedings of the 8th International Conference on Distributed Computing Systems,

Maes, P. (1987) Concepts and Experiments in Computational Reflection. In Proceedings of the ACM Conference on Object-Oriented Languages (OOPSLA)

Magnani, L. (2005) "Chance Discovery and the Disembodiment of Mind", Knowledge-Based Intelligent Information and Engineering Systems: 9th International Conference, KES 2005, pp: 547-553 Melbourne, Australia.

Mamei, Marco, Ronaldo Menezes, Robert Tolksdorf, Franco Zambonelli (2006) Case Studies for Self-Organization in Computer Science. Journal of Systems Architecture 52(8), pp: 443-460

Mamei M., F. Zambonelli (2004) Programming Pervasive and Mobile Computing Applications with the TOTA Middleware, In Proceedings of the International Conference On Pervasive Computing (Percom), IEEE CS Press,

Mamei M., F. Zambonelli, L. Leonardi (2004) Co- Fields: A Physically Inspired Approach to Distributed Motion Coordination, IEEE Pervasive Computing 3 (2) pp: 52–61.

Mano J-P., C. Bourjot, G. Leopardo, P. Glize (2006) Bio- inspired Mechanisms for Artificial Self-Organised Systems. Informatica 30(1) pp: 55-62.

Mayfield, J., Y. Labrou, T. Finin (1995) Evaluating KQML as an Agent Communication Language. In M. Wooldridge, J.P. Muller and M. Tambe (Editors) Intelligent Agents II, Springer pp: 347-360

McCarthy, J. (1959) Programs with Common Sense. In Proceedings of the Teddington Conference on the Mechanization of Thought Processes. HMSO, London, pp: 75-91

McCarthy, J. (1963) Situations, Actions and Causal Laws. Technical report, Stanford University, 1963. Reprinted in Semantic Information Processing, ed: M. Minsky, pp 410-417, MIT Press, Cambridge, Massachusetts, 1968

McCarthy, J. and P. Hayes (1968) Some Philosophical Problems from the Standpoint of Artificial Intelligence. Machine Intelligence, 4(1): pp: 463-502.

Menezes, R. R. Tolksdorff (2004) Adaptiveness in Linda-based Coordination Models, In Proceedings of the First International Workshop on Engineering Self-Organising Applications (ESOA 2003) LNCS 2977, Springer, 2004.

Meyer J.-J., R. Wieringa (1993) Deontic Logic in Computer Science: Normative System Specification. J. Wiley and Sons.

Microsoft Corporation (2007) (COM) Component Object Model Technologies. http://www.microsoft.com/com/default.mspx (Accessed April 2007)

Microsoft Corporation (2004) Dynamic Systems Initiative Overview, White paper, 31st March, 2004

Miller, R., & M. Shanahan (2002). Some Alternative Formulations of the Event Calculus. In A. C. Kakas & F. Sadri (Eds.), Computational logic: Logic programming and beyond: Essays in honour of Robert A. Kowalski, part II Vol. 2408, pp: 452–490. Berlin: Springer.

Milner, R. (1980) A Calculus of Communicating Systems. Volume 92 of Lecture Notes in Computer Science. Springer-Verlag.

Milner, R. (1983) Calculi for Synchrony and Asynchrony. Theoretical Computer Science 25(3), pp: 267-310

Milner, R., J. Parrow and D. Walker (1992). "A Calculus of Mobile Processes". Information and Computation (100) pp: 1--40.

Miseldine P., A. Taleb-Bendiab, D. England, M. Randles (2006) Addressing the Need for Adaptable Decision Processes within Healthcare Software. In the Proceedings of the 23rd Annual Conference on Health Informatics in the UK (HC2006), 2006, pp.: 117-124.

Miseldine, P., A. Taleb-Bendiab, (2005a) A Programmatic Approach to Applying

Sympathetic and Parasympathetic Autonomic Systems to Software Design, in Self-Organisation and Autonomic Informatics (1) (Ed: H. Czap et al) pp: 293-303, IOS Press, Amsterdam,

Miseldine P., A. Taleb-Bendiab (2005b) An Empirical Study into Governance Requirements for Autonomic E-Health Clinical Care Path Systems. In Proceedings of the 1st International Workshop on Requirements Engineering for Information Systems in Digital Economy (REISDE 2005) within ICETE05, pp: 171-178.

Miseldine, P., A. Taleb-Bendiab (2006) CA-SPA: Balancing the Crosscutting Concerns of Governance Autonomy in Trusted Software. In the Proceeding of the IEEE Workshop on Trusted and Autonomic Computing Systems 2006. Vienna, Austria, pp: 471-475.

Montoya, J. M., R.V. Solé (2000) Small World Patterns in Food Webs. arXiv.org:cond-mat/0011195.

Moore, R.C. (1985) A Formal Theory of Knowledge and Action. In J.B. Hobb, R.C. Moore (Eds.) Formal Theories of the Commonsense World, pp: 319-358, Ablex Publishing Corporation, Norwood, NJ.

Moreno, A. (2003) Agents Applied in Healthcare. AI Communications 16(3), pp: 135-137.

Moses Y., M. Tennenholtz (1995) Artificial Social Systems. Computers and Artificial Intelligence, 14(6), pp: 533–562.

Mueller, E. T. (2006) Commonsense Reasoning. Morgan Kaufmann, San Francisco, USA.

Mueller, E. T. (2007) Discrete Event Calculus with Branching Time. In Eyal Amir, Vladimir Lifschitz, & Rob Miller (Eds.), Logical Formalizations of Commonsense Reasoning: Papers from the 2007 AAAI Spring Symposium (pp. 126-131). Technical Report SS-07-05. AAAI Press Menlo Park, CA.

Naur, P., B. Randell (eds.) (1968) Software Engineering, Report on a Conference sponsored by the NATO Science Committee, Garmisch, Germany, October 1968.

Netcraft (2005) August 2005 Web server Survey. http://survey.netcraft.com/Reports/0508/

Netcraft (2007) The April 2007 Web Server Survey. http://news.netcraft.com/archives/2007/04/02/april_2007_web_server_survey.html
Object Management Group (1998), The Common Object Request Broker: Architecture and Specification Revision 2.2. OMG, 492 Old Connecticut Path, Framingham, MA 01701, USA.

Newman, M.E.J. (2001) The Structure of Scientific Collaboration Networks. In the Proceedings of National Academy of Sciences of the USA 98, pp: 404-409.

Nilsson, N.J. (1984) Shakey the Robot. SRI Technical Note No. 323, Stanford Research

Institute, Menlo Park, California.

Odelstad, J., L. Lindahl (2002). The Role of Connections as Minimal Norms in Normative Systems. Legal Know ledge and Information Systems. Eds. T. Bench- Capon, A. Daskalopulu and R. Winkels. Amsterdam: IOS Press.

Ohsawa Y. (editor) (2001) "Proceeding of the 1st International Workshop on Chance Discovery", Japanese Society for Artificial Intelligence.

Onody, R.N., P.A. de Castro (2004) Complex Network Study of Brazilian Soccer Players. Phys. Rev. E, 70, 037103: arXiv:cond-mat/0409609v2

Paczuski, M., D. Hughes (2004) A Heavenly Example of Scale Free Networks and Self-Organized Criticality. Physica A 342(1-2) pp: 158-163.

Padget, J. (2003) The Role of Norms in Autonomic Organisations. In Proceedings of IJCAI Workshop on AI and Autonomic Computing: Developing a Research Agenda for Self-Managing Computer Systems, Acapulco, Mexico, 10th August, 2003.

Parunak, H. (1997) Go to the Ant: Engineering Principles from Natural Multi-Agent Systems, Annals of Operations Research 75,pp: 69– 101.

Petri, C.A. (1962) Kommunikation mit Automaten. PhD thesis, Institut fuer Instrumentelle Mathematik, Bonn.

Poole, D. (1997) The Independent Choice Logic for Modelling Multiple Agents Under Uncertainty. Artificial Intelligence 94(1–2) pp: 7– 56.

Power, R. (1984) Mutual Intention. Journal for the Theory of Social Behaviour 14(1), pp:85-102

Randles M., A. Taleb-Bendiab, P. Miseldine, A. Laws (2005) Adjustable Deliberation of Self-Managing Systems. In proceedings of the 12th IEEE International Conference and Workshops on the Engineering of Computer-Based Systems (ECBS2005), pp: 449-456.

Randles, M., A. Taleb-Bendiab, P. Miseldine (2006a) Mind out of Programmable Matter: Exploring Unified Models of Emergent Autonomy in LNCS Volume 3825, pp: 65-73, Springer Berlin / Heidelberg.

Randles, M., A. Taleb-Bendiab, P. Miseldine (2006b) Addressing the Signal Grounding Problem for Autonomic Systems. In Proceedings of International Conference on Autonomic and Autonomous Systems (ICAS06), pp: 21,Santa Clara, USA, July 19-21, 2006.

Rao, A. S., M.P. Georgeff (1991). Modeling Rational Agents within a BDI-Architecture In Proceedings of Knowledge Representation and Reasoning (KR&R-91). Morgan Kaufman Publishers, San Matteo, CA. pp: 473-484.

Ratner, C. and H. Lumei (2003) Theoretical and Methodological Problems in Cross-Cultural Psychology. Journal for the Theory of Social Behavior, 2003, 33, pp. 67-94.

Redner, S. (1998) How Popular is your Paper? An Empirical Study of the Citation Distribution. The European Physical Journal B 4, pp: 31.

Reiter R. (1991) The Frame Problem in the Situation Calculus: A Simple Solution (sometimes) and a Complete Result for Goal Regression. Artificial Intelligence and Mathematical Theory of Computation: Papers in Honor of John McCarthy, ed: V. Lifschitz, pp: 359-380, Academic Press, San Diego, California.

Reiter, R. (1996) Natural Actions, Concurrency and Continuous Time in the Situation Calculus. In A.C. Aiello, J. Doyle, S.C. Shapiro (Eds.) Principles of Knowledge Representation and Reasoning: Proceedings of 6th International Conference (KR'96), pp: 2-13. Morgan Kaufman, San Francisco, CA, USA.

Reiter, R. (2001) Knowledge in Action. MIT Press, Cambridge, MA, USA.

Resnick, M. (1994) Turtles, Termites, and Traffic Jams, MIT Press, 1994.

Reynolds, C. (1987) Flocks, herds and schools: A distributed behavioral model, Computer Graphics 21 pp: 25–34.

Rouff, C., M.G. Hinchey, J. L. Rash, W. F. Truszkowski (2005) Towards a Hybrid Formal Method for Swarm-Based Exploration Missions. *In* Proceedings of the 29th Annual IEEE/NASA Software Engineering Workshop *sew*, pp. 253-264.

Rouff, C., A. Vanderbilt, W. Truszkowski, J. Rash, M. Hinchey (2004) Verification of NASA Emergent Systems. In Proceeding of the 9th International Conference on Engineering Complex Computer Systems (ICECCS2004), pp. 231-238.

Rouff, C., A. Vanderbilt, W. Truszkowski, J. Rash, M. Hinchey (2004) Verification of NASA Emergent Systems. In Proceedings of the 9th International Conference on Engineering of Complex Computer Systems, pp:231-238.

Sanner S,, C. Boutilier (2006) Practical Linear Value-approximation Techniques for First-order MDPs. In Proceedings of the 22nd Conference on Uncertainty in Artificial Intelligence. AUAI Press, Arlington Virginia, USA.

Schelling, T.C. (1971) Dynamic Models of Segragation. Journal of Mathematical Sociology 1(2), pp: 143-186

Scherl, R., H.J. Levesque (1993) The Frame Problem and Knowledge Producing Actions. In the Proceeding of the 11th National Conference on Artificial Intelligence (AAAI-93), Washington DC, USA, pp: 689-695

Schiffel S., M. Thielscher (2006) Reconciling Situation Calculus and Fluent Calculus. In Proceedings of the 21st National Conference on Artificial Intelligence and the 18th Innovative Applications of Artificial Intelligence Conference (AAAI06).

Schmidt D., M. Stal, H. Rohnert, and F. Buschmann (2001) Pattern-Oriented Software Architecture, vol. 2. John Wiley, 2001.

Schubert, L.K. (1990) Monotonic Solution of the Frame Problem in Situation Calculus: An Efficient Method for Worlds with Fully Specified Actions. In H.E. Kyberg, R.P. Loui, G.N. Carlson (Eds.) Knowledge, Representation and Defeasible Reasoning, pp: 23-67, Kluwer Academic Press.

Searle, J.R. (1980) Mind, Brains and Programs, Behavioural and Brain Sciences 3, pp: 417-457,

Sergot, M.J. (2001) A Computational Theory of Normative Positions. ACM Transactions on Computational Logic, 2(4) pp: 581—622.

Seyeed-allaei, H., G Bianconi, M Marsili (2005) Scale-Free Networks with an Exponent less than two, Condensed Matter Eprints, arXiv:cond-mat/0505588.

Shanahan, M. (1996). Robotics and the Common Sense Informatic Situation. In W. Wahlster (Ed.), Proceedings of the Twelfth European Conference on Artificial Intelligence, pp: 684–688 Chichester, UK: John Wiley.

Shoham, Y. (1993) Agent-Oriented Programming. Artificial Intelligence, 60 pp: 51-92

Simon, H.A. (1996) The Sciences of the Artificial. MIT Press, Cambridge, MA, USA.

Singh, A., M. Haahr (2006) Creating an Adaptive Network of Hubs Using Schelling's Model. Communications of the ACM 49(3), pp: 69-73

Smith B. C. (1982) Reflection and Semantics in a Procedural Language PhD. Thesis, MIT, Cambridge, Mass., USA.

Stamper, R.K. (2000) Information Systems as a Social Science: an Alternative to the FRISCO Formalism. In E.D.Falkenberg, K.Lyytinen, A.A.Verrijn-Stuart, (eds.), Information Systems Concepts: An Integrated Discipine Emerging, pp: 1-51 Kluwer Academic Publishers, Boston,.

Steiner, J., M. Hagner (2007) Runtime Analysis of a Self-Adaptive Hard Real-Time Robotic Control System. In proceedings Fourth IEEE International Workshop on Engineering of Autonomic and Autonomous Systems (EASe'07) pp: 53-60.

Strogatz, S. (2003) Sync: The Emerging Science of Spontaneous Order, Hyperion Press.

Sumpter. D. J. T. (2000) From Bee to Society: An Agent-Based Investigation of Honey Bee Colonies. PhD. Thesis, The University of Manchester, UK.

Sun Microsystems (2002) N1- Introducing Just-in-time Computing. White paper

Szyperski C. (1999) Component Software: Beyond Object-Oriented Programming. Addison-Wesley, 1999.

Taddeo, M., L. Floridi, (2005) Solving the Symbol grounding Problem: A Critical Review of Fifteen Years of Research, Journal of Experimental & Theoretical Artificial Intelligence, 17(4), pp: 419-445.

Taleb-Bendiab, A., D. England, M. Randles, P. L. Miseldine, K. Murphy (2006) A Principled Approach to the Design of Healthcare Systems: Autonomy vs Governance. Reliability Engineering and System Safety Journal 91(12), pp: 1576-1585.

Thielscher, M. (1998) Introduction to the Fluent Calculus. Linköping Electronic Articles in Computer and Information Science, 3(14) http://www.ep.liu.se/ea/cis/1998/'1'/

Tianfield, H., R. Unland (2004) Towards Autonomic Computing Systems. Engineering Applications of Artificial Intelligence 17(7), pp: 689-699.

Tofts, C., (1993) Processes with Probabilities, Priority and Time. Formal Aspects of Computing 6(5) pp: 536-564

Toner, J., Y. Tu (1999) Flocks, Herds and Schools: A Quantitative Theory of Flocking, Physical Review E. 58 4828–4858.

Travers, J., S. Milgram (1969) An Experimental Study of the Small World Problem. Sociometry, Vol. 32, No. 4, pp: 425-443.

Traversat, B., A. Arora, M. Abdelaziz (2002) Project JXTA 2.0 Super-Peer Virtual Network; www.jxta.org/project/www/docs/JXTA2.0protocols1.pdf

Tuomela, R., K. Miller (1988) We-Intentions. Philosophical Studies 53, pp: 367-389

Turkkan, J. (1989) Classical Conditioning: the New Hegemony. Behavioural and Brain Sciences 12, pp: 121-179

Vassos, S., H. Levesque (2007) Progression of Situation Calculus Action Theories with Incomplete Information. In Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI-07) pp: 2029-2035.

Want, R., T. Pering, D. Tennenhouse (2003) Comparing Autonomic and Proactive Computing. IBM Systems Journal 42(1) pp:129-135.

Watts, D.J., S.H. Strogatz (1998) Collective Dynamics of 'Small-World' Networks. Nature 393 pp: 440-442.

Weyns D., K. Schelfthout, T. Holvoet, and O. Glorieux (2004) Role based model for adaptive Agents. Fourth Symposium on Adaptive Agents and Multi-agent Systems at the AISB '04 Convention, 2004.

Widom, J. and S. Ceri (1995) Active Database Systems. Morgan-Kaufmann, San Mateo, California.

Wilensky, U. (2005). NetLogo *Preferential* Attachment Model. http://ccl.northwestern.edu/netlogo/models/PreferentialAttachment. Center for *Connected* Learning and Computer-Based Modeling, Northwestern University, Evanston, IL.

Wilensky, U. (2007) NetLogo Simulation Software Version 4beta1 http://ccl.northwestern.edu/netlogo Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL.

Wooldridge, M., N.R. Jennings (1995) Intelligent Agents: Theory and Practice. The Knowledge Engineering Review 10(2) pp: 115-152.

Wu, K., A. Liu (1995) Rearrangement Inequality. Mathematics Competitions 8(1), pp: 53–60.

Yahoo (2005) Our Blog has Grown and so has our Index. http://www.ysearchblog.com/archives/000172.html

Zambonelli, F., N. R Jennings, M. J Wooldridge (2003) Developing Multi-agent Systems: the Gaia Methodology. ACM Transactions on Software Engineering and Methodology, 12(3), September 2003.

Zambonelli F., M.-P. Gleizes, M. Mamei, and R. Tolksdorf (2004) Spray Computers: Frontiers of Self-Organisation for Pervasive Computing. Second International Workshop on Theory and Practice of Open Computational Systems (TAPOCS 2004) in 13th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE'04), pp: 397-402. Los Alamitos, USA.

Zhu, H. (2005) Formal Reasoning about Emergent Behaviours of Multi-Agent Systems. In Proceedings of 17th International Conference on Software Engineering and Knowledge Engineering (SEKE'05), pp: 280-285

Appendix 1

Adaptive Middleware

Adaptive middleware becomes more necessary as the complexity of systems increases. Emerging distributed applications often involve multimedia communication, mobility, embedded computing, group communications, and high availability. Addressing these issues means that systems must adapt to changing conditions, such as unexpected security attacks, hardware failures, and dynamic environments. Existing technologies are available to provide extensive middleware services, through Object Request Brokers (ORB), Tuple-Spaces, Peer-to-Peer software and Message and Event Oriented systems. The Microsoft.NET framework, based on DCOM (Microsoft Corporation, 2007) and the Common Object Request Broker Architecture (CORBA) (Object Management Group, 1998) are typical of current standards in middleware. It is not the intention here, however, to provide a review of middleware systems/architectures. The problem of specifying and implementing self-governing systems is a middleware concern, as the self-governance is treated as a separate concern within the middleware. Thus the usual middleware software technologies of: Computational Reflection (Maes, 1987), Component-based design (Szyperski, 1999), Aspect-Oriented Programming (Kiczales et al, 1997) and software design patterns (Schmidt et al, 2001) can be utilized. In this work these concepts will be utilized in a different manner and emphasis then current practice permits, by considering a system as collective of components with adjustable autonomy.

- Computational Reflection sees a system as a controller of itself. Thus reasoning and analysis of its own actions and operation can proceed at runtime, as an integral part of the systems operation emanating from a logical specification that provides for deliberation and adaptation at global level, through an observation system, and at local level with the component/agents autonomous behaviour.
- Component-based design is used as a means to model the system with agent like entities/components. In this way these active components can be arranged with autonomous functionality to operate in their local environment and provide the basis for a bottom-up engineering of function.
- Aspect Oriented Programming gives the necessary separation of concerns allowing the self-governance of the system to be considered apart from the

many other middleware concerns, but crucially enabling the middleware, as well as the application system construction, to be adapted at runtime.

- Software design patterns become even more useful when applied to autonomous components or agents. The whole subsystems and flexible interactions can be reused: Agent designs and implementations can be reused within and between applications.

There are a number of middleware-resident self-governance initiatives currently proposed, including: Hewlett Packard's Adaptive Infrastructure (HP World, 2003), Sun's N1 (Sun Microsystems, 2002), Microsoft's Dynamic Services Initiative (Microsoft Corporation, 2004), Cisco's Adaptive Network Care (IBM and Cisco Systems, 2003) and Intel's Proactive Computing (Want et al, 2003). The most widely studied and most well known is IBM's Autonomic initiative (IBM, 2003). It is this that currently provides the state of the art in research regarding systems self-governance and implementations.

Appendix 3

Publications by the Author

N. Badr, A. Taleb-Bendiab, M. Randles, D. Reilly, "A Deliberative Model for Self-Adaptation Middleware Using Architectural Dependency". In the Proceedings of DEXA Workshops 2004 pp: 752-756

M. Randles, A. Taleb-Bendiab, P. Miseldine, "A Stochastic Situation Calculus Modelling Approach for Autonomic Middleware", In the Proceedings of the 5th Annual Conference on the Convergence of Telecommunications, Networking and Broadcasting (PGNET2004) pp: 317-322, Liverpool, UK, 2004

M. Randles, A. Taleb-Bendiab, P. Miseldine, A. Laws "Adjustable Deliberation of Self-Managing Systems." Proceedings of IEEE International Conference on the Engineering of Computer Based Systems (ECBS 2005) pp: 449-456, Maryland, USA, 2005

D. W. Bustard, R. Sterritt, A. Taleb-Bendiab, A. Laws, M. Randles, Frank Keenan, "Towards a Systemic Approach to Autonomic Systems Engineering". In Proceeding of ECBS 2005 pp: 465-472

A. Taleb-Bendiab, D.England, P. Misedine, M. Randles, K. Murphy, "Sources of Complexity in the Design of Healthcare Systems: Autonomy vs. Governance". Workshop on the Complexity in Design and Engineering University of Glasgow, March 2005.

M. Randles, A. Taleb-Bendiab, P. Miseldine "Using Stochastic Situation Calculus to Formalise Danger Signals for Autonomic Computing" Proceedings of the 6th Annual Conference on the Convergence of Telecommunications, Networking and Broadcasting (PGNET2005) pp: 241-246, Liverpool, UK, 2005

A. Taleb-Bendiab et al, "Model-Based Self-Managing Systems Engineering", In Proceeding of 3rd Intl. Workshop on Self-Adaptive Autonomic Computing in DEXA2005, Copenhagen, Denmark, 2005

M. Randles, A. Taleb-Bendiab, P. Miseldine, "Mind out of Programmable Matter: Exploring Unified Models of Emergent System Autonomy for Collective Self-

Regenerative Systems" Workshop on Radical Agent Technology (WRAC2005), NASA Goddard Space Flight Centre Greenbelt, MD, USA, 2005

M. Randles, A. Taleb-Bendiab, P. Miseldine, "A Logical Treatment for the Emergence of Control in Complex Self-Organising Systems" In Self-Organisation and Autonomic Informatics (ISBN I-58603-577-0) (Editors: Hans Czap, Rainer Unland, Cherif Branki, Huaglory Tianfield),pp: 3-17, IOS Press, Amsterdam, 2005.

M. Randles, A. Taleb-Bendiab, P. Miseldine, "Mind out of Programmable Matter: Exploring Unified Models of Emergent Autonomy" in LNCS Volume 3825, pp: 65-73, Springer Berlin / Heidelberg, 2006

P. Miseldine, A. Taleb-Bendiab, D. England, M. Randles, "Addressing the Need for Adaptable Decision Processes within Healthcare Software". In the Proceedings of the 23rd Annual Conference on the Health Informatics in the UK (HC2006), pp: 117-124, Harrogate, UK. March 2006.

M. Randles, A. Taleb-Bendiab, "Analysing Infrastructure and Emergent System Character for Ubiquitous Computing Software Engineering" In the Proceedings of the International Workshop on Software Engineering Challenges for Ubiquitous Computing (ed. G. Kortuem), pp: 9-10, Lancaster University, June, 2006

P. Miseldine, M. Randles, D. Lamb, A. Taleb-Bendiab, "Towards the Automated Engineering of Autonomic Systems" Proceedings of the 7th Annual Conference on the Convergence of Telecommunications, Networking and Broadcasting (PGNET2006) pp: 393-398, Liverpool, UK, 26th/27th June 2006

M. Randles, A. Taleb-Bendiab, P. Miseldine, " Addressing the Signal Grounding Problem for Autonomic Systems". Proceedings of International Conference on Autonomic and Autonomous Systems, 2006 (ICAS06), pp: 21,Santa Clara, USA, July 19-21, 2006 ** **AWARDED BEST PAPER IN CONFERENCE****

A. Taleb-Bendiab, David England, Martin Randles, Philip Miseldine and Karen Murphy,"A Principled Approach to the Design of Healthcare Systems: Autonomy vs Governance", Reliability Engineering & System Safety, Volume 91, Issue 12, pp: 1576-1585, December 2006

Martin Randles, A. Taleb-Bendiab, Philip Miseldine, "Towards Scaleable Self-Governance through Situated Cognitive Systems: Utilising Deliberative Logic to Provide an Encompassing Epistemic Layer", In Proceeding of 4th Intl. Workshop on Self-Adaptive Autonomic Computing in DEXA2006, Krakow, Poland, 6th September, 2006, pp: 114-118.

Martin Randles, A. Taleb-Bendiab, Philip Miseldine, "Harnessing Complexity: A Logical Approach to Engineering and Controlling Self-Organizing Systems", International Transactions on Systems Science and Applications Volume 2, Number 1 (2006), pp: 11-20

Ali Obied, A. Taleb-Bendiab, Martin Randles, "Self Regulation in Situated Agents", In Proceedings of 1st Intl. Workshop on Agent Technology and Autonomic Computing, Erfurt, Germany, Sepember 2006.

P. Miseldine, A. Taleb-Bendiab, D. England, M. Randles,"Addressing the Need for Adaptable Decision Processes within Healthcare Software", Medical Informatics & The Internet in Medicine, Vol. 32 (1), 2007, pp: 35-41

Martin Randles, A. Taleb-Bendiab, Philip Miseldine, David Lamb, "Using Signatures of Self-Organisation for Monitoring and Influencing Large Scale Autonomic Systems", In proceedings of fourth IEEE Workshop on Engineering of Autonomic and Autonomous Systems (EASe'07), 2007 pp: 24-26

Martin Randles, Hong Zhu, A. Taleb-Bendiab, A Formal Approach to the Engineering of Emergence and its Recurrence In the Proceedings of The Second International Workshop on Engineering Emergence in Decentralised Autonomic Systems (EEDAS 2007) at the IEEE International Conference on Autonomic Computing (ICAC07) Jacksonville, Florida, USA, June 11, 2007.