

Integration of System Requirements and Multi-Agent Software Architecture

Lúcia Rosângela David Bastos

Supervisor

Dr. Jaelson Freire Brelaz de Castro

Centro de Informática
Universidade Federal de Pernambuco
Recife, Pernambuco, Brazil

Submitted in partial fulfilment of the requirements for the Degree of Doctor of Philosophy.

August 2005

ABSTRACT

Requirements Engineering and Software Architecture have become established areas of research, education and practice within the software engineering community. However, the two disciplines have evolved independently from each other and promising areas of mutual interest remain to be explored.

Requirement Engineering is concerned with identifying the purpose of the system and the context in which it will be used. Software Architecture is related to the definition of software components and their properties and relationships. Additionally, Software Architecture has been recognised to have a deep impact on the achievement of non-functional requirements such as availability, reliability, maintainability, safety, confidentiality, evolvability, and so forth.

There is a clear relationship between requirements and architectures. However, evolving and elaborating system requirements into software architectures satisfying those requirements is still a difficult task, mainly based on intuition. Understanding and supporting the interaction among software requirements and architectures remains one of the most challenging problems in software engineering research. To narrow these problems, we are proposing techniques for deriving architectural model in concert with Multi-Agent System (MAS) requirement specifications.

This thesis proposes a framework to identify and map key architectural elements and the dependencies among these elements, based on the stated system requirements and organisational concepts. Our approach advocates that Multi-Agent System (MAS) corresponds to an organisational structure in which actors are members of a group in order to accomplish a goal. An organisation comprises groups, goals, members, roles and interactions. The basic idea behind the notion of groups is to provide the means to collectively refer to a set of roles performing a collaborative set of tasks to achieve their goals. A member assigned to roles does not work in isolation but interacts and cooperates with other roles.

Our framework is based on two complementary models that separate the concerns of the social organisation from those of the architectural organisation. In the Organisational Model, the main goals are identified by understanding a requirement model as the functionality requested for the system. The organisation of the social system consists roles and interactions, as intended by the system and its environment. Additionally, the goals and softgoals define the selection of an architectural style. In the Assignment Model, roles are clustered into sub-groups. These sub-groups are related to components, based on their similarity with the architectural components. The result is an architectural configuration, which is the allocation of sub-groups to architectural components.

ACKNOWLEDGEMENTS

TABLE OF CONTENTS

1	INTRODUCTION	1
1.1	INTRODUCTION	2
1.2	MOTIVATION	2
1.2.1	The Structural gap between Requirements and Architectures.....	3
1.2.2	Goal-based techniques in Requirements Engineering	4
1.2.3	Organisational Concepts in Multi-Agent Systems	5
1.2.4	Agent-Based techniques in Business Modelling	6
1.3	RESEARCH APPROACH	6
1.3.1	Research Context.....	7
1.4	MAIN CONTRIBUTIONS	8
1.5	THESIS STRUCTURE.....	8
2	BACKGROUND	10
2.1	INTRODUCTION	11
2.2	REQUIREMENTS ENGINEERING.....	11
2.2.1	Functional Requirements.....	12
2.2.2	Non-functional Requirements	13
2.3	ORGANISATIONAL CONCEPTS IN REQUIREMENTS ENGINEERING	14
2.3.1	Organisational Goals	15
2.3.2	Social Groups	17
2.4	THE MULTI-AGENT SYSTEMS	20
2.4.1	Agents	21
2.4.2	Non-Functional Requirements for Agents.....	21
2.4.3	Multi-Agent Organisation	22
2.5	ORGANISATIONAL CONCEPTS IN MULTI-AGENT SYSTEM METHODOLOGIES	24
2.5.1	MaSE.....	24
2.5.2	GAIA.....	25
2.5.3	AALAADIN Model	26
2.6	THE TROPOS PROJECT	27
2.6.1	Early and Late Requirement in Tropos Project	27
2.6.2	Multi-Agent Architectural Styles	32
2.6.3	NFR Framework and Architectural Selection	34
1.7	PERSPECTIVES OF MULTI-AGENT SYSTEMS APPROACHES.....	36
1.8	SUMMARY	37
3	SYSTEMATIC INTEGRATION BETWEEN REQUIREMENTS AND ARCHITECTURE USING ORGANISATIONAL CONCEPTS	39
3.1	INTRODUCTION	40
3.2	THE SIRA FRAMEWORK IN TROPOS.....	40
3.3	THE SIRA FRAMEWORK	41
3.4	THE SIRA PROCESS.....	42
3.4.1	The Organisational Model Specification	43
3.4.2	Assignment Model Definition	46
3.4.3	Architectural Configuration	49

3.5	SUMMARY	49
4	THE ORGANISATIONAL MODEL	51
4.1	INTRODUCTION	52
4.2	THE BANK EXAMPLE	52
4.3	THE ORGANISATIONAL MODEL.....	54
1.1.1	Group and Sub-groups	56
1.1.2	Goal Refinement	57
1.1.3	Role Identification and Interactions	59
1.1.4	Norms.....	64
1.4	ANALYSIS OF ARCHITECTURAL ALTERNATIVES	65
1.4.1	Architectural Elements in Multi-Agent Systems.....	66
1.4.2	Non-Functional Properties of MAS.....	66
1.4.3	Analysis of Architectural Alternatives for the Bank Example	68
1.5	SUMMARY	68
5	THE ASSIGNMENT MODEL.....	70
5.1	INTRODUCTION	71
5.2	SOCIAL NETWORK ANALYSIS	71
5.2.1	The Centrality of Actors.....	72
5.2.2	The Structural Equivalence between Actors	78
5.3	THE ASSIGNMENT MODEL	79
5.3.1	Clustering Roles into Sub-Groups.....	80
5.3.2	Correlation Analysis between Groups and Architecture	83
5.4	MAPPING SUB-GROUPS TO COMPONENTS.....	87
5.4.1	The Bank Architecture in Structure-in-5 Style.....	88
5.5	SUMMARY	89
6	THE E-COMMERCE CASE STUDY	90
6.1	INTRODUCTION	91
6.2	THE E-COMMERCE EXAMPLE	91
6.3	THE ORGANISATIONAL MODEL OF THE MEDI@ EXAMPLE	93
6.3.1	Task Refinement	93
6.3.2	Role Identification.....	94
6.3.3	Architectural Selection.....	96
6.4	ASSIGNMENT MODEL OF THE MEDI@ EXAMPLE.....	97
6.4.1	Cluster Analysis	98
6.4.2	Correlation Analysis between Group and Components.....	100
6.5	ARCHITECTURAL CONFIGURATION.....	103
6.6	SUMMARY	104
7	CONCLUSION.....	105
7.1	FINAL CONSIDERATIONS	106
7.2	CONTRIBUTIONS.....	107
7.3	DISCUSSION OF RESULTS.....	107
7.3.1	Agents and Requirements.....	108
7.3.2	Agents and Architectures	108

7.4	FUTURE WORKS.....	109
7.5	SUMMARY	110

LIST OF FIGURES

<i>Figure 1 – The context in the Tropos Project</i>	<i>7</i>
<i>Figure 2 - Early and late requirements.....</i>	<i>14</i>
<i>Figure 3 - Elements from i* framework.....</i>	<i>28</i>
<i>Figure 4 - SD model for Media Shop.....</i>	<i>29</i>
<i>Figure 5 - Types of links in Tropos.....</i>	<i>31</i>
<i>Figure 6 - The Join Venture Style</i>	<i>33</i>
<i>Figure 7 - The Structure-in-5 Style</i>	<i>33</i>
<i>Figure 8 - Selecting architectural style in Tropos</i>	<i>36</i>
<i>Figure 9 – The SIRA Framework in Tropos.....</i>	<i>41</i>
<i>Figure 10 – The SIRA Framework.....</i>	<i>42</i>
<i>Figure 11 - The SIRA Process Activities.....</i>	<i>43</i>
<i>Figure 12 – The Organisational Model Specification.....</i>	<i>44</i>
<i>Figure 13 - Tropos representation of goal refinement with means-end and contribution analysis.</i>	<i>44</i>
<i>Figure 14 - Assignment Model Definition.....</i>	<i>47</i>
<i>Figure 15 - The organisational chart of a bank office example.....</i>	<i>52</i>
<i>Figure 16 - Bank work example in i* notation.....</i>	<i>53</i>
<i>Figure 17 - A partial schema for the i* Requirement Model.</i>	<i>54</i>
<i>Figure 18 - Organisational Model in Telos</i>	<i>55</i>
<i>Figure 19 - Cooperation between two actors to fulfil two goals.....</i>	<i>57</i>
<i>Figure 20 - Means-ends Analysis for the task Lending Money.....</i>	<i>58</i>
<i>Figure 21 - Means-ends Analysis for the task Account Management</i>	<i>59</i>
<i>Figure 22 - Role Interaction Graph of the goal Get Loan</i>	<i>62</i>
<i>Figure 23 - Role interaction graph of the goal Deposit Money.....</i>	<i>63</i>
<i>Figure 24 - Star Network</i>	<i>72</i>
<i>Figure 25 - Interaction Graph of the Structure-in-5 Architecture</i>	<i>76</i>
<i>Figure 26 - Interaction Graph after the cluster analysis</i>	<i>82</i>
<i>Figure 27 - Bank Architecture using the Structure-in-5 Style.....</i>	<i>89</i>
<i>Figure 28 - Strategic Rationale model for Medi@ actor.....</i>	<i>92</i>
<i>Figure 29 - Task refinement for the goal Process Internet Orders</i>	<i>94</i>
<i>Figure 30 - Role interaction of the goal Process Internet Orders</i>	<i>95</i>
<i>Figure 31 - Interaction Graph of the Join Venture Style</i>	<i>97</i>
<i>Figure 32 - Medi@ system architecture model.....</i>	<i>104</i>

LIST OF TABLES

<i>Table 1 - The Correlation catalogue.....</i>	<i>46</i>
<i>Table 2 - Bank Initial Group specification.....</i>	<i>56</i>
<i>Table 3 - Role categorisation in the Bank domain.....</i>	<i>59</i>
<i>Table 4 – The Lender role specification.....</i>	<i>60</i>
<i>Table 5 - Interaction Protocols.....</i>	<i>61</i>
<i>Table 6 - Interaction matrix of the goal Get Loan.....</i>	<i>64</i>
<i>Table 7 - Interaction matrix of the goal Deposit Money.....</i>	<i>64</i>
<i>Table 8 - The Correlation Catalogue for the Bank Example.....</i>	<i>68</i>
<i>Table 9 - Degree Centrality Measures of the goal Get Loan (g=6).....</i>	<i>74</i>
<i>Table 10 - Degree Centrality Measures of the goal Deposit Money (g=6).....</i>	<i>75</i>
<i>Table 11 - Closeness Centrality of the goal Get Loan (g=6).....</i>	<i>75</i>
<i>Table 12 - Closeness Centrality of the goal Deposit Money (g=6).....</i>	<i>76</i>
<i>Table 13 - Interaction matrix of the Structure-in-five architecture.....</i>	<i>77</i>
<i>Table 14 - Degree Centrality of the Structure-in-five Style (g=5).....</i>	<i>77</i>
<i>Table 15 - Closeness Measures of the Structure-in-five Style.....</i>	<i>77</i>
<i>Table 16 - Interpretation of the Correlation Coefficients.....</i>	<i>78</i>
<i>Table 17 – The similarity coefficients of the bank example.....</i>	<i>81</i>
<i>Table 18 - Interaction matrix after the cluster analysis.....</i>	<i>82</i>
<i>Table 19 - Centrality Measures (Degree and Closeness) (g=5).....</i>	<i>83</i>
<i>Table 20 - Centrality Correlation Analysis.....</i>	<i>84</i>
<i>Table 21 - Centrality Correlation in the Bank example.....</i>	<i>85</i>
<i>Table 22 - The structural similarity based on In-degree.....</i>	<i>85</i>
<i>Table 23 - The structural similarity based on Out-degree.....</i>	<i>86</i>
<i>Table 24 – Similarity Correlation between Bank sub-groups and Structure-in-Five components.....</i>	<i>87</i>
<i>Table 25 – Correlation Analysis of the Bank Example.....</i>	<i>87</i>
<i>Table 26 - Interaction matrix of the task “Internet Shop Managed”.....</i>	<i>95</i>
<i>Table 27 - The Correlation catalogue for Medi@.....</i>	<i>96</i>
<i>Table 28 - Interaction matrix of the Join Venture Style.....</i>	<i>97</i>
<i>Table 29 - Degree Centrality of the task Internet Shop Managed (g=4).....</i>	<i>98</i>
<i>Table 30 - Closeness Centrality for the task Internet Shop Managed (g=4).....</i>	<i>98</i>
<i>Table 31 - Role similarity of the e-commerce example.....</i>	<i>99</i>
<i>Table 32 - Degree Centrality Measures of the Joint Venture Style (g=4).....</i>	<i>100</i>
<i>Table 33 - Closeness Centrality for the Joint Venture Architecture (g=4).....</i>	<i>100</i>
<i>Table 34 - Centrality Correlation between Medi@ Sub-groups and Joint Venture components.....</i>	<i>101</i>
<i>Table 35 - Similarity based on Out-degree for the e-commerce example.....</i>	<i>102</i>
<i>Table 36 - Matrix based on In-degree for the e-commerce example.....</i>	<i>102</i>
<i>Table 37 - Similarity Correlation Table of the Medi@ e-commerce example.....</i>	<i>103</i>
<i>Table 38 - Correlation Analysis of the Medi@ e-commerce example.....</i>	<i>103</i>

Chapter 1

Introduction

This chapter provides an introduction to the research work of this thesis. In addition, we describe the research objectives and provide an overview of the approach taken.

*"No part of the delivery process is as critical, nor as difficult -
because requirements map the human world to the technological world."
Jim Highsmith*

1.1 INTRODUCTION

Requirements Engineering and Software Architecture have become established areas of research, education and practice within the software engineering community. Unfortunately, the terminology and concepts used for architectural description are quite different from those used for requirement specification. However, there is a clear relationship between requirements and architectures. Understanding and supporting this relationship remains one of the most challenging problems in software engineering research. In this thesis we present an approach for the integration of system requirements and software architectures. We propose a model to support organisational concepts in ways that enrich the system requirements and take into account architectural requirements in the context of multi-agent systems.

The remainder of this chapter is organised as follows. Section 1.2 presents the rationales for this research. Section 1.3 introduces the research approach. Section 1.4 describes the scope and contributions of our thesis to different research areas. Finally, section 1.5 provides an outline of the following chapters.

1.2 MOTIVATION

Requirement engineering is concerned with identifying the purpose of a system and the context in which it will be used. Requirements are related to concepts such as goals, conflicts, options and agreements [Boehm et al, 1998], [Boehm et al, 2001], [Lamsweerde, 2001]. Moreover, systems characteristics and properties (functional and non-functional) are also described in terms of requirements [Chung et al, 2000]. Software architectures are important because they represent the particular abstraction for understanding the structure of a system [Bass et al, 1998], [Garlan, 1993]. Software architecture has long been recognised to have a profound impact on the achievement of non-functional goals ("ilities") such as availability, reliability, maintainability, safety, confidentiality, evolvability, and so forth.

Since the mid-1970s when Requirement Engineering was established as a distinct field of investigation and practice, the definition has evolved. Initially it was defined as the early stage of software engineering concerned with the definition of software requirements to be used for constraining and measuring the eventual software implementation. While this description again focuses on aspects of software systems, it extends earlier definitions to also include aspects of the real world. As a discipline, Requirement Engineering thus brings software engineering concerns closer to the problems experienced in organisational settings.

The foundation and concepts of software architecture have come into being since the 1960's with the development of software. However, the term 'Software Architecture' only began to attract substantial attention both from the research community and from the industry in the 1990's. The most important reason for this is the growth of software systems: hundreds of thousands of lines of code were commonplace. The challenge to create, evaluate and maintain these large and complex software systems has greatly stimulated the growth of the importance of software architecture [Kazman, 2001], [Shaw, 2001].

As with most other important concepts in computer science, no definition of software architecture is commonly agreed upon [Bass et al, 1998], [Perry and Wolf, 1992], [Shaw and Garlan, 1996]. However, it is usually agreed that software architecture is concerned with the components of the system and their interactions. Most researchers agree that the main concern of software architecture is the high-level structure, in contrast with the detailed design of software. Architecture models a solution for the problem described in the requirements and provides high-level abstractions for the representation of the structure, behaviour and main properties of a software system. Moreover, the architecture should capture and reflect the quality properties of the system under construction (e.g. safety, performance, reliability).

A fundamental problem faced when relating requirements and architecture is the large conceptual distance and the so-called structural gap between requirements and architectural models. This thesis adopts organisational-based analysis for the architectural modelling of multi-agent systems. Multi-Agent Systems (MAS) are promising as models of organisation because they are built on the idea that most work in human organisations is done based on beliefs, communication, cooperation, and distribution [Weiss, 2000], [Wooldridge et al, 2002].

This thesis describes multidisciplinary research that combines aspects from Requirement Engineering, Agent Theory, and Organisational Studies. In the following subsections, we give an overview of some fundamental aspects of these different fields: the structural gap between requirements and architectures, the goal-based techniques in requirement engineering, the organisational concepts in multi-agent system, and the agent-based techniques in business modelling.

1.2.1 The Structural gap between Requirements and Architectures

The relationship between requirements and effective software architecture for a desired system is not readily obvious. In general, the inter-dependencies and constraints between requirements elements and architectural elements are not well understood and subsequently only limited guidance is available for matching requirements and architecture. For example, it is often unclear whether and how a statement of stakeholder goals should affect the desired system's architecture. Similarly, it is not easy to decide how to effectively address the functional and non-functional requirements.

The existence of conceptual differences between what to do (requirements) versus how to do it (architecture, design and code) constitutes a semantic gap. Some critical challenges emerge when trying to fill this gap and to reconcile the vocabulary between requirements and architectures. Some issues are listed below [Grunbacher et al, 2004]:

- Requirements are frequently captured informally in a natural language. On the other hand, entities in a software architecture description are usually specified in a semi-formal manner;
- System properties described in non-functional requirements are difficult to specify in an architecture model;

- The iterative and concurrent evolution of requirements and architectures demand that the development of architecture have to be based on incomplete requirements. In addition, some requirements can only be understood after modelling the system architecture;
- Mapping requirements onto architectures and maintaining consistency between them is a difficult task. A single requirement may address multiple architectural concerns and a single architectural element may have numerous non-trivial relations to various requirements;
- Complex and large-scale systems have to satisfy hundreds, possibly thousands of requirements. Consequently, it is difficult to identify and to refine architecturally relevant information contained in the requirements;
- Requirements and software architecture emerge in a process involving heterogeneous stakeholders with conflicting goals, expectations, and terminology. Supporting different stakeholders demands finding the right balance for these divergent interests.

The above-described differences between requirements and architectures make it difficult to build a bridge that spans the two.

In this thesis we advocate that a Multi-Agent System (MAS) corresponds to the organisational structure in which actors are members of a group in order to accomplish a goal. An organisation comprises groups, goals, members, roles and interactions. A member assigned to roles does not work in isolation but interacts and cooperates with other roles. We believe that such an organisational model will provide a stepping-stone to the integration between requirements and architectures.

1.2.2 Goal-based techniques in Requirements Engineering

The current thinking in the area of Requirement Engineering (RE) is that requirements need to be articulated within the framework of ‘real-world’ knowledge. Research in this area focuses on conceptual techniques and tools for explicitly capturing and representing, in a structured way, the domain knowledge subsequently used to drive the system development phases [Loucopoulos and Karakostas, 1995].

Organisational models represent (implicitly or explicitly) the goals of individuals, groups, or organisations, in which a goal is a desired condition potentially attained at the end of an action (or process). Stakeholder goals and their role in defining and solving design problems are topics of longstanding interest in the field of requirements engineering.

Goals are the objectives and targets for a system. Goal analysis approaches emphasise the use of the goal notion in order to understand or describe aspects of the real world, in an attempt to *rationally* find better ways of coping with the complexity of human affairs.

Research in the areas of business planning, business administration and control, and workflow analysis endorses a goal-directed view. This view is based on the premise that in collaborative work situations, people do not strictly follow rules or procedures; rather, they are aware of

personal and group goals and act accordingly. This is especially true when people are not faced with well-structured, repetitive processes. In such situations, they tackle ill-structured problems where both the intended outcome and the possible routes need be specified. This scenario is common in RE [Bubenko 1983], [Kavakli, 1999&2004].

Goals are considered a potential motivator to action, and are distinct from the plans, procedures or other means of attaining the goal. Goal-driven approaches focus on why systems are constructed. Furthermore, since goals are typically more stable than requirements, they are a beneficial source for requirements derivation [Anton, 1997], [Dardenne et al, 1993], [Yu, 1995], [Kavakli, 1999], [Lamsweerde, 2001]. Goals are operationalised and refined into requirements and point to new, previously unconsidered scenarios. Similarly, scenarios also help in the discovery of goals. Although the merits and benefits of scenario-based and goal-based analysis in requirements engineering are well understood, researchers are now faced with the question of how to use scenarios and goals in a complementary fashion [Cockburn, 1995], [Anton and Potts, 1998], [Cysneiro and Leite, 1999], [Rolland et al, 1998].

This thesis uses the organisational concepts to model multi-agent systems as a social organisation of coordinated autonomous individuals that interacts in order to achieve specific, possibly common goals.

1.2.3 Organisational Concepts in Multi-Agent Systems

According to Yoshino (1995), “An organisation is a consciously coordinated social entity, with a relatively identifiable boundary, that functions on a relatively continuous basis to achieve a common goal or a set of goals”. Organisation theory is the discipline that studies both *structure* and *design* in such social entities.

Classical organisational theories define an organisation as a whole consisting of agents performing tasks [Fayol, 1916], [Morgan, 1996]. There is a fixed structure of agent tasks and agent communication. Organisation theory describes how practical organisations are actually structured, offers suggestions on how new ones can be constructed, and how old ones can change to improve effectiveness [Fincham and Rhodes, 1999]. The management principles from organisational theory can be applied in multi-agent systems [Gazendam and Jorna, 1996].

Multi-agent systems (MAS) present a useful software engineering paradigm where systems are described as individual agents pursuing high-level goals. Although having such an abstraction seems promising, its widespread adoption among system designers has not materialised yet. Efforts are challenged not only by known distributed system issues, but also by the complexities associated with supporting agent collaboration.

Our approach to develop agent-based solutions emphasises the importance of organisational modelling as a critical initial step in producing business applications.

1.2.4 Agent-Based techniques in Business Modelling

There are at least three distinct approaches to domain modelling, such as: business process reengineering (BPR), object-oriented (OO), and artificial intelligence (AI), the foundation of agent-based technology. All three approaches are model-based and offer techniques for describing problem domains. The convergence of OO, BPR, and AI technology has results in a significant breakthrough in building models of the organisation that are capable of end-to-end integration of business analysis and software systems. Although the fusion of these disciplines is not yet complete to the point of standardisation, some researchers have recognised and launched major agent-oriented development initiatives [Fox et al, 1996], [Wagner, 2000], [Wooldridge et al, 2000], [Wood and DeLoach, 2001], [Yu, 2001], [Zambonelli et al, 2001].

Traditional BPR methods are expressive and easily understood when describing business process and organisation. However, these approaches produce separate models of data, processes and organisation that soon become unmanageable. Moreover, the models are difficult to integrate and validate for the development of software systems. Information is captured using informal techniques, and incomplete lifecycle models lack a clear implementation path.

Object-oriented methods are well suited to software engineering and have potential for reuse. However, they are neither inherently business oriented nor sufficiently expressive for most problem domains. Moreover, they often foster premature commitments to design and implementation strategies. Consequently, they do not capture a sufficiently rich domain model for significant reuse. In practice, most OO methods are informal and not repeatable.

Business models must make a distinction between passive and active goal seeking objects (such as agents) to close the gap between model and reality. Business object models need to contain behavioural declarations, role definitions, awareness, and constraints.

Agent-based technology can be useful to enhance organisational modelling as well as offer new techniques for developing business applications. An agent-oriented perspective allows us to develop expressive models of the organisation as a foundation for adaptive and reusable business software.

1.3 RESEARCH APPROACH

The concept of desirable social behaviour is of utmost importance when Multi-Agent Systems (MAS) are considered from an organisational point of view. The organisational analogy that putting actor (humans or agent) together does not form efficient organisations unless sufficient processes, regulations, infrastructure and organisation goals are also in place. This leads to a rising awareness that MAS can best be understood, modelled and developed if they are inspired by human social organisation [Castelfranchi, 2000], [Artikis et al, 2001], [Zambonelli et al, 2001]. This is, in many ways, a novel concept within requirement engineering and agent researches, even if sociability has always been considered an important characteristic of actor and agents.

Organisational structure will then result from the efforts to reach the organisational goals. Structure may be achieved through the division of work, definition of roles and positions, and vertical and horizontal relationships among them. This structure needs to be understandable through modelling techniques. Such techniques describe the organisational or business environment as co-operation among different organisational actors, (e.g., human individuals, software systems, workgroups, etc.). Co-operation among actors presupposes that these actors share common goals and act towards their fulfilment.

The above considerations lead to the following high-level research questions:

1. How can organisations and other types of societies be modelled in a way that integrates global aims and requirement taking in account the autonomy of their participants?
2. Is the agent paradigm adequate to model organisational systems?
3. Can such an organisational model be used to define architectural patterns in an abstract organisation (such as multi-agent system)?

In order to use the agent paradigm, our approach advocates that the organisational concepts can be useful to model agents' software architectures. In other words, our approach needs to provide means for modelling the social level of roles and interactions, in which organisational goals are assigned to sub-groups of actors that perform roles. Roles do not work in isolation, but interacts to perform specific tasks.

1.3.1 Research Context

Our approach aim at extends the Tropos project to deal with organisational group, roles and interactions in the late requirements phase. This organisational model provides a conceptual view of the multi-agent system as an organisational architecture. This allows relating the functional and non-functional goals elaborated in requirements models to the architectural components.

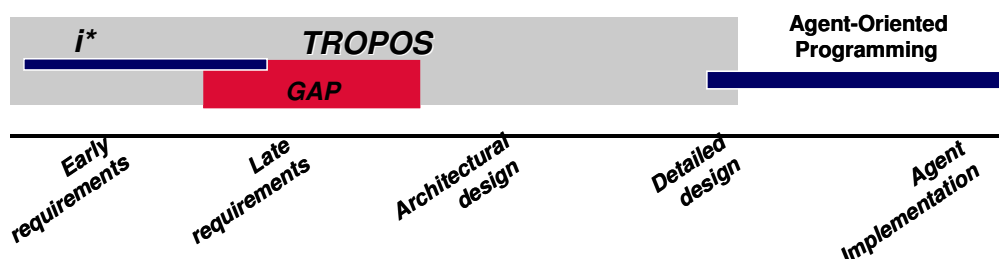


Figure 1 – The context in the Tropos Project

The Tropos project has made some preliminary efforts to narrow the semantic gap between requirements and architectures: the definition of a set organisational architectural styles and

the selection of architectural style using the NFR framework. However, Tropos does not explicitly cover the correlation between requirement elements and architectural elements. The information available in the requirement models is not enough to derive architectural information, as seen in Figure 1.

The social model of the Tropos project provides a conceptual view of the software system and its organisational structure, as it represents a localised organisation or society of agent. Its attributes model social aspects, most notably the organisational architecture, social goals, social roles and social constraint of that society (group of agents).

In brief, our approach consists of extending the requirement models with organisational concepts that allows the establishment of relationships among the requirements elements and the architectural components.

1.4 MAIN CONTRIBUTIONS

The interaction between software requirements and architectures remains one of the challenging problems in software engineering research. In this thesis, we present theoretical foundations as well as some guidelines to point out the feasibility for systematic integration of two phases of a requirement-driven software development methodology using the organizational concepts in requirement and architectural models.

The main contribution of our integration approach is to assure that the architecture components represent or are associated to the main organisational requirements, which in turn will be fulfilled by the architecture components.

This work offers a framework to guide the integration process of requirements and architecture using organisational models. The key concepts addressed in this integration framework are the following:

- Definition of the Organizational Model to extend the requirements with identification of roles, responsibilities and interactions;
- Definition of the Assignment Model using social analysis to cluster roles into sub-groups, and to measure similarity between sub-groups and architectural components.
- Definition of the Architectural Model using same concepts from requirements model.
- A process with a set of activities to carry out the transition from requirements to architecture.

1.5 THESIS STRUCTURE

This thesis is conceptually divided into seven individual chapters.

- Chapter 2 gives an overview of basic theories and related research areas. This chapter provides some background on the different disciplines, which aim to provide a common understanding of the work in the remainder of this thesis.
- Chapter 3 introduces our framework to narrow the integration between requirements and architecture. Our approach is based on two complementary models and a process. The process focuses on a systematic way to carry out the transition from requirement model to architectural configuration.
- Chapter 4 presents our organisational approach for agent communities. The focus of this organisational modelling is on designed social groups, with explicit objectives and structure, in opposition to emergent societies, that result from the ‘ad hoc’ interaction of agents. This organisational model can be thought of as a kind of abstract code of behaviour that governs how members of MAS work according to organisational requirements, such as goals, roles and interactions.
- Chapter 5 details the Assignment Model. The assignment model clusters roles into sub-groups and examines the degree of similarity between sub-groups and components. The purpose is to map organisational elements into architectural components based on their interactions. The result is the first architectural configuration of the multi-agent systems.
- Chapter 6 presents an e-commerce case to track the transition from an organisational model to an architectural configuration.
- Chapter 7 concludes the thesis by discussing the overall contribution of the research in the context of related work in the area. In addition, the chapter discusses the limitations of the approach and points to future research directions.

Chapter 2

Background

This chapter overviews an overviews basic theories in order to provide a better understanding of the concepts and ideas described in the thesis. It intends to outline the foundations upon which our research is built, rather than a complete overview.

2.1 INTRODUCTION

In the last few years the relation of Requirement Engineering (RE) to the organisational context has attracted much interest from the software engineering community. Research in this area is based on the premise that in designing software systems, requirement engineers aim to ‘improve’ organisational situations which are seen as problematic – or, at least, as less than perfect. As a discipline, RE thus brings the software engineering concerns closer to problems experienced in organisational settings. More specifically, RE addresses the problems associated with the business goals, plans, processes and systems to be developed in order to achieve organisational objectives [Loucopoulos and Karakostas, 1995], [Lamsweerde, 2001].

Furthermore, organisational approaches to the modelling of Multi-Agent Systems (MAS) are gaining ground in agent research, and several research groups have presented models for agent societies. Many researchers are dealing with new development techniques suitable for MAS that conceive multi-agent systems as organisations of agents that interact to achieve common goals. Goals, groups and roles have been used in such methodologies as an intuitive and natural concept for defining agent organisations [Ferber and Gutknecht, 1998], [Partsakoulakis and Vouros, 2002]. Concepts such as organisational rules [Zambonelli, 2002], norms and institutions [Dignum and Dignum, 2001], [Esteva et al, 2001], and social structures [Parunak and Odell, 2002] arise from the idea that the effective engineering of MAS needs high-level, agent-independent concepts and abstractions that explicitly define the architecture of the organisation in which agents operate.

This chapter introduces the notions of system requirements, multi-agent systems, software architecture and their relations with organisational concepts. Section 2.2 introduces the traditional definition of requirement engineering. Section 2.3 provides a general vision of the organisational concepts in new researches of requirements engineering. Section 2.4 provides the background notion of multi-agent systems. Section 2.5 overviews the use of organisational concepts in multi-agent systems. Section 2.6 presents the Tropos methodology. Section 2.7 discusses the approaches presented in this chapter. Finally, Section 2.8 presents a summary of this chapter.

2.2 REQUIREMENTS ENGINEERING

Since the mid-1970s, when RE was established as a distinct field of investigation and practice, the RE definition has evolved from software systems [IEEE, 1998] to a broader perspective that incorporate aspects of systems and organisational environment [Loucopoulos and Karakostas, 1995], [Pohl, 1996], [Yu and Mylopoulos, 1997], [Zave, 1997].

Despite the fact that there is no common definition of Requirement Engineering, five activities to be performed have been identified [Pohl, 1996]:

1. Requirement *elicitation* is about understanding the organisational situation that the system under consideration aims to improve, and about describing the needs and

constraints concerning the system under development. The relevant knowledge about the problem (system) is distributed among many stakeholders;

2. Requirement *negotiation* aims to establish an agreement on the requirements of the system among the various stakeholders involved in the process;
3. Requirement *specification* involves a mapping of real-world needs onto requirement models. The wider view on RE suggests that requirement specification goes beyond the traditional approach (which concentrates on functional requirements modelling), and involves the modelling of the enterprise context providing the purpose of the intended system;
4. Requirement *validation* intends to ensure that the derived specification corresponds to the original stakeholder needs and conforms to the internal and/or external constraints set by the enterprise and its environment;
5. Requirement *management* is the fundamental activity in requirements engineering and is concerned with changes to a system requirement. The goal of requirement management is to make sure that the project is able to succeed in delivering what the client wants, with the least effort in terms of time and cost. The world moves on, expectations change, and people change their minds about what they want. Requirements are added in response to changing stakeholder needs, or because they were missed in the initial analysis. Changes to requirement documentation need to be managed. Typical changes to requirement specifications include adding or deleting requirements, and fixing errors.

Different authors have different definitions, but there are clearly non-functional requirements as well as functional requirements. Functional requirements describe the behaviour (functions or services) of a system as it supports user goals, tasks or activities. Non-functional requirements include constraints and qualities. These differ from functional requirements in that they refer to features that systems must possess [Chung et al, 2000]. Informally, functional requirements describe *what* the system must do, while non-functional requirements describe *how well* these functional requirements will be satisfied.

2.2.1 Functional Requirements

Functional requirements express the intended behaviour of a system *what* a system will do. This behaviour may be expressed in terms of the services, tasks or functions that a system is required to perform [Loucopoulos and Karakostas, 1995].

Functional requirement specification is concerned with the definition of functional, static and dynamic aspects under which the system information needs to operate. During software development, functional requirements are usually incorporated into software artefacts. At the end of the process, all functional requirements must have been implemented in such a way that the software satisfies the requirements defined at the early stages.

Functional requirements usually have localised effects, i.e., they affect only the part of the software that addresses the functionality defined by the requirement. By contrast, non-functional requirements are usually widespread throughout distinct parts of the software.

2.2.2 Non-functional Requirements

Non-functional requirements (NFR or system qualities) have significant effects upon software product to define the final quality and justifying the design decisions that constrain the implementation of functional requirements. They capture required properties of the system, such as performance, security and maintainability. In other words, they address *how well* some behavioural or structural aspects of the system should be accomplished [Chung et al, 2000]. As such, NFRs should be addressed as early as possible in the software lifecycle and properly built into a software architecture before a detailed design proceeds on an undesirable path [Chung and Yu, 1998], [Chung, 1998], [Gross and Yu, 2001].

Non-functional requirements can be perceived as instances of generic categories that play the role of quality factors to be achieved in the development of the software system [ISO/IEC, 1991], [Chung et al, 2000]. For example:

- *Efficiency* refers to system performance under certain conditions. Efficiency can be described in the specification phase and successively refined in the design phase. However, it can be measured only in the implementation phase;
- *Portability* refers to the ability of the system to be moved around its environment (hardware or software). Portability has attributes that depend upon the ability to be adapted, installed, conformed and replaced;
- *Maintainability* refers to the ability of the system to react to changes in its environment or in its requirement specification. The attributes of this category are: simplicity of analysis, conciseness, stability and modularity;
- *Expandability* refers to the ability of the system to react to the inclusion of new components, caused by revised customer demands. Expandability and maintainability are quality factors that are strongly dependent on the architecture.
- *Robustness* refers to the ability of the system to react to incorrect inputs or to component flaw.
- *Safety* is related to the protection of the system and of the data that it manages.

Other non-functional requirements are the so-called constraint requirements of the environment or of the architecture [Ambriola and Gervasi, 1996]. *Constraints* are characteristics of the “information system” or the development organisation that limit the development in some way. These are true architectural requirements determined by precise hardware and software needs, for instance, a requirement of a real time system that must be implemented on pre-existing hardware. Quality requirements determine “*how*” desirable (or *how good*) a given implementation may be, while constraint requirements determine if the implementation is acceptable at all.

Requirements do not only consist of functions. This misconception occurred, in part, because the structured analysis techniques focus on articulating functional requirements. Requirements also need to be understood within an organisational context.

2.3 ORGANISATIONAL CONCEPTS IN REQUIREMENTS ENGINEERING

Enterprise and organisational models have long been developed and recognised in Requirement Engineering. The need for greater precision, completeness and consistency of requirements has led to the proposal of many tools and techniques. However, when developing system truly fulfils the real needs of an organisation, it is required to have a deeper knowledge of the intentional and strategic aspects of system actors. Many requirement models cannot cope with the questioning of the reasons and end up dealing only with the functions of the system.

Several works in requirements engineering make an important distinction between the capture of initial requirements (early requirements) and the capture of final requirements (late requirements) [Yu and Mylopoulos, 1997], [Bresciani et al, 2001], [Fuxman et al, 2001b], [Castro et al, 2002]. There is a need to distinguish the initial phases of requirement engineering, when the requirements engineer is trying to understand an organisational scenario, from later phases in which a system solution is formulated.

Much of requirements engineering research has taken as a starting point the initial requirements statements, which express the customer's wishes about what the system should do. This early phase aims to model and analyse stakeholder interests and how they might be addressed, or compromised, by various system-and-environment alternatives. These requirements are often ambiguous, incomplete, inconsistent, and usually expressed informally.

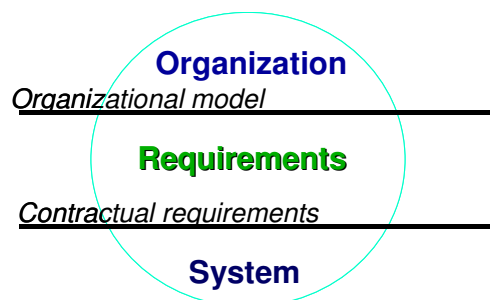


Figure 2 - Early and late requirements

By contrast, requirement activities focus on the completeness, consistency and verification of the requirements. Early-phase requirements engineering activities have objectives and assumptions that are different from those of the later phase. Early requirements are typically informal and relate to organisational and non-functional issues. The emphasis during this

phase is on the understanding the motivations and reasons associated with system requirements. The “early-phase” requirements engineering includes activities that consider how the intended system meet organisational goals, why the system is needed, what alternatives exist, what the implications of the alternatives, and how the stakeholders’ interests and concerns might be addressed [Dardenne et al, 1993], [Wieringa et al, 1997], [Alencar et al, 2001]. Figure 2 shows these boundaries between organisation model (early requirements) and contractual requirements (late requirements) analysis.

This thesis understand that goals and requirements are close related since the latter represent a target to be attained and a wish to be fulfilled. Organisational goals are concerned with the objectives of the enterprise, the strategic politics adopted, and finally, the entire organisational structure. The structure of an organisation defines the roles of the various actors, and the ways in which they influence the organisation to achieve a goal. The following sub-sections will provide a background notion about these organisational concepts of goals, social groups, actors, roles, and social positions.

2.3.1 Organisational Goals

A *goal* is an objective the system under consideration should achieve. Goal formulations thus refer to intended properties to be ensured. Goals may be formulated at different levels of abstraction, ranging from high-level and strategic concerns to cover core concerns: functional concerns associated with the services to be provided, and non-functional concerns associated with quality of service – such as safety, security, accuracy, performance, and so forth.

Requirement modelling techniques can be used to help deal with the knowledge and reasoning needed in the earlier phase of requirements engineering. As a result, understanding the organisational context and rationales (the “why”) that generate systems requirements can be just as important for the ongoing success of the system. This extensive applicability of organisational concepts in RE suggests that the elucidation and manipulation of organisational goals is a natural and inherent part of RE, which can be expressed as follows:

- Goals can provide a criterion for *sufficient completeness* of requirements specification; the specification is complete with respect to a set of goals if all the goals can be proved to be achieved from the specification and the properties known about the domain considered [Yu, 1995].
- Goals can provide a criterion for requirements *pertinence*; a requirement is pertinent with respect to a set of goals in the domain considered if its specification is used in the proof of one goal at least [Yu, 1995].
- Goals provide the rationale for requirements, in a way similar to design goals in design processes. A requirement appears because of some underlying goal, which provides a base for it. In particular, for business application systems, goals may be used to relate the software-to-be to the organisational or business context [Yu, 1993], [Castro et al, 2002].
- Requirement engineers are faced with many alternatives to be considered during the requirement elaboration process. Alternative goal refinements provide the right level

of abstraction at which decision-makers can be involved for validating choices being made or suggesting other alternatives overlooked so far. Alternative goal refinements allow alternative system proposals to be explored [Lamsweerde, 2001].

- By focusing on goals rather than on interactions, it is also easier to investigate different ways of achieving the stated goals and to detect and solve conflicts between them. Goals have been recognised to provide the roots for detecting conflicts among requirements and for eventually resolving them [Lamsweerde and Letier, 2000], [Letier and Lamsweerde, 2002].
- In terms of information, requirements are more volatile than goals. A requirement represents one particular way of achieving some specific goal; the requirement is therefore more likely to evolve towards another way of achieving that same goal than the goal itself;
- Goals drive the identification of requirements to attain them; they have been shown to be among the basic driving forces (together with scenarios), for a systematic requirement elaboration process [Anton and Potts, 1998], [Dubois and Heymans, 1998], [Lamsweerde, 2001].

Current goal-oriented approaches have demonstrated that the use of goal analysis offers benefits and advantages at almost every stage in the RE process:

1. In goal elaboration, the modelling of goals has been proposed during requirements elicitation in order to describe organisational behaviour. The i* approach [Yu, 1995], [Yu and Mylopoulos, 1997], [Yu, 2001] provides a description of work organisation in terms of dependency relationships among actors. This approach will be detailed in section 2.6. The KAOS methodology [Dardenne et al, 1993], [Letier and Lamsweerde, 2002] highlights the importance of explicitly representing and modelling organisational goals, and their relations to operational system components are explicitly stated. KAOS is aimed at supporting the processing of requirements – from the high level goals that should be achieved by the composite system to the operations, objects and constraints to be implemented by the software. GBRAM [Anton, 1997] offers prescriptive guidelines on how to extract goals from different sources into one ordered goal set.
2. In scenario approaches, goals are considered as the contextual property of a scenario i.e., a property that relates the scenario to its organisational context [Leite et al, 1997]. Goals can be used to structure scenarios by connecting every action in a scenario to a goal assigned to an actor [Cockburn, 1995]. Goals can organise scenarios in a goal hierarchy [Rolland et al, 1998]. A goal is defined as something a stakeholder hopes to achieve in the future, whilst a scenario expresses a possible way in which the goal can be achieved. By assigning goals to scenarios and organising the goals using three types of relations (refine, AND, OR) a structure for managing scenarios is also established.
3. In non-functional requirements (NFR) framework [Chung et al 2000] the representation of non-functional requirements is provided in terms of interrelated goals. The NFR model consists of goals that represent non-functionality. This

framework will be used to select architectural styles in the Tropos Methodology (Section 2.6).

Organisations can be characterised by social groups endowed with macro-level goals. Actor's behaviour is influenced by macro-level goals. Social groups will be discussed in the sequel.

2.3.2 Social Groups

The purpose of any society is to allow its members to coexist in a shared environment and pursue their respective goals in co-operation with others. Societies usually specify mechanisms of social order in terms of common norms and rules that members are expected to adhere to [Davidsson 2001]. The main aspects in the definition of 'society' are goal, structure, rules and norms. The social structure is determined by roles, interaction rules and communication. Rules and norms describe the desirable behaviour of members and are established and enforced by institutions that often have a legal standing and thus lend legitimacy and security to members.

An organisation, like social group, consists of a structure, roles, and the functioning of that structure in terms of social actions and interactions. For an organisation to show reliable and stable courses of activity, some actor relationships are needed to define the roles of mutual dependence and hierarchy. These member actors distinguish organisations from other forms of communities.

In sum, an organisation is as a set of actors and their interactions, which are regulated by mechanisms of social order and created by more or less autonomous actors in order to achieve common goals [Dignum et al, 2002].

2.3.2.1 Actor

An *actor* is a *self-controlling participant* in an organisation. Examples are human beings in social organisations or a car with its driver in a transport system. Self-control is an essential quality of an actor. A car without a driver is not an actor but a passive object. The car is a tool used by the human actor and is seen as part of the actor.

An actor communicates and collaborates with other actors within the system and with external actors in the system environment. Through this communication and collaboration, the actors create, organise and maintain the system.

From an organisational perspective, the main function of an individual actor is to perform a role that contributes to the global aims of the society. An actor may have roles in different systems. For instance an actor can be an employee in a company, father in his family and member of a club. A *role* is a combination of responsibility, authority and competencies that allows actions as initiator or executor in certain transactions. The responsibilities and authorities of a role are the result of commitments between the actors in the system and may be laid down in contracts or agreements.

Here, an actor models an entity that has strategic goals and intentionality within the system or the organisational setting. An actor represents a physical, social or software agent as well as a role or position.

2.3.2.2 Roles

In sociological organisational theory the phenomenon of organisations is well discussed. Roles are not independent of the sociological structure or environment where they evolve. Some authors have advocated role-oriented approaches to agent society development, especially when it is manifest to take an organisational view on the application scenario [Fox et al, 1996], [Zambonelli et al, 2001], [DeLoach, 2002].

The role theory is widely applied for enterprise modelling, postulating that individuals occupy positions in an organisation. In every organisation in the society, people play various roles to implement functions of the system. The roles might be “president”, financial manager or secretary. Each role in the organisation communicates with each other in order to perform their responsibilities.

The term *Role* varies in meaning depending on the context, such as:

- **Sociology:** In sociology, a role is considered as a set of expectations (e.g. attitudes, beliefs and assumptions) that are associated with a certain position. Roles are defined independently from the actual role keeper and prescribe in particular accepted behaviour and action [Weber, 1976].
- **Organisation Theory:** In traditional organisation theory the situation of members within an organisation is specified by roles. Roles are thereby groups of people at the same hierarchical level or in the same organisational unit. Every company with its social entities can be regarded as a complex structure of functionally and hierarchically defined roles. In this context, roles are used to represent hierarchical structures, to define rules of substitution, to allocate tasks and to determine the social position of an employee [Biddle and Thomas, 1979].
- **Functional Orientation:** From a functional perspective every role is described by a set of tasks to be performed. A role can be held by different actors, while an actor can fulfil different roles at the same time. Roles are thereby used to standardise range of operations and to ensure the controlled completion of tasks.
- **Skill Management:** Within the scope of competence-specific classification each role is defined by a set of skills, abilities and know-how (e.g. industry skill standards or language abilities that are necessary to execute tasks). On the basis of these qualification requirements every task is assigned to suitable roles.
- **Information Security:** With regard to security matters users with equal rights are subsumed into groups called roles [Lupu, 1998]. Rights and permissions are assigned to roles rather than to individual users (role-based access control). Each role consists of a set of allowed transactions (e.g. documents, data or applications).

Roles have been used in multi-agent systems (MAS) in order to achieve coherence in teams of cooperative agents in domains where well-coordinated activity is required [Mao and Yu, 2004], [Partsakoulakis and Vouros, 2004], [Sudeikat et al, 2004]. Roles in specific contexts of action provide abstract specifications of distributed behavioural patterns. When these roles are performed in a coordinated fashion, they can accomplish a specific objective. The explicit specification of roles and their relations and interdependencies provides a shared context for group of agents to track the tasks performed. Different approaches are used to identify role concepts for modelling social actors in organisational structures [Fox et al, 1996], for modelling social agents in multi-agent systems [Lupu and Sloman, 1997], [Trzebiatowski and Münch, 2001], [Wood and DeLoach, 2001], [Zambonelli et al, 2003], [MESSAGE, 2004], and for modelling authority in role-based access control (RBAC) [Crook et al, 2002].

However, there is no agent framework or methodology to model MAS that accomplishes the abstractions provided by social groups (such as roles and interactions) and the concepts of architectural styles. An integrated approach is needed which combines these different aspects.

In this thesis, roles are defined as an abstract representation of an actor function or service within a social group. Roles are local to a group. Position represents the sphere of influence of an actor within a group, while roles encapsulate the way an actor should act within a group.

2.3.2.3 Social Position

Every social actor has a status or social position. Actors often hold more than one social position and status at a time. In the latter case, they possess what is known as a status set. For example, an actor can be a son, brother and teacher. The actor status set grows and expands throughout our lifetime. Every status has roles attached to it. Roles are the behaviours, responsibilities and privileges that pertain to individuals occupying a specific status [Biddle and Thomas, 1979].

Since an organisational position serves as a placeholder for a specific actor, the competencies related to a position should be assigned to the position itself instead of to the person occupying it (although it is possible to assign competencies to an actor directly as well). However, in real life organisations there is a difference between qualifications and competencies. While the former refers to attributes of a specific actor and cannot be removed (i.e. “speaks Portuguese”), the latter depends on the position the actor occupies in a given organisation (i.e. “may sign orders over \$ 50.000”) and can be assigned and removed.

With the increasing size of a company, the number of similar positions is likely to increase. In this case *social position* serves as a template for organisational identity (e.g. “secretary”), which can be refined in a position instance (e.g. “Prof. Jaelson’s secretary”).

In this thesis, *social position* is defined as an abstract representation of actor organisational identity within a group. Social position encapsulates the hierarchy relations and roles of an actor within an organisational group. Social positions are local to group.

In short, the concept of social behaviour is of utmost importance when Multi-Agent Systems (MAS) are considered from an organisational point of view. This leads to a rising awareness that MAS can best be understood, modelled and developed if they are inspired by social organisation. Some basic concepts of MAS will be outlined in the next section.

2.4 THE MULTI-AGENT SYSTEMS

Several definitions from different disciplines have been proposed for the term Multi-Agent Systems (MAS). MAS can be defined as a loosely coupled network of problem-solver entities that work together to find answers to problems that are beyond the individual capabilities or knowledge of each entity [Durfee et al, 1989].

More recently, the term Multi-Agent System has been given a more general meaning, and it is now used for all types of systems composed of multiple autonomous components. Agents are often deployed in environments in which they interact with other agents (including both people and software) that may have conflict aims. Such environments are known as *multi-agent systems* [Luck et al, 2003]. MAS have the following general characteristics [Jennings et al, 1998], [Wooldridge, 2002]:

- Each agent has incomplete capabilities to solve the problem;
- There is no global control;
- Data is decentralised;
- Computation is asynchronous;

The Multi-Agent Systems deal with breaking down the problem and assigning particular sub-problems to those agents with the best abilities to solve them. Although each agent has its own goals and interests, the assigning of the sub-problem is done in such a way as to solve the global problem in the most appropriate and efficient manner. A benefit of multi-agent systems is their scalability. That is, a new agent can easily be added to the multi-agent systems, because it is inherently modular. Therefore, to solve the problem, an agent in MAS interact in one of the following ways:

- Cooperation: the agent reveals its goals and structure to the other party, and both agents have the same goals.
- Coordination: the agent reveals its goal to the other party, but the agents have different goals;
- Competition: This is divided into loose and strict competition. In loose competition, the agent reveals its goal to the other agent, but encapsulates its symbol structure. In strict competition, the agent encapsulates both its goal and symbol structure.

MAS that consist of human agents as well as software agents have been studied as environments for computer supported cooperative work, as expert systems, and as decision support systems. Instead of hierarchically structured computer programs, multi-agent simulation models of organisation offer a bottom-up perspective: based on the capabilities of agents and resulting communication and cooperation. The characteristics of the organisation as a whole can be defined, discovered, or explained [Gazendam and Jorna, 1996].

In this work, MAS is a system consisting of agents that communicate and cooperate on the basis of organisational architectures. Multi-agent systems are described on the basis of organisational structures built from groups and roles, independently from the actual nature of the agent. A group is a set of agents. An agent is an active component that interacts with its environment and has the ability to play one or more roles. Some agent concepts will be discussed in more detail below.

2.4.1 Agents

Studies of Multi-Agent Systems (MAS) focus on systems in which many agents interact with each other. The agents are considered to be autonomous entities, such as software programs or robots. Their interactions can be either cooperative or selfish. That is, the agents can share a common goal (e.g. an ant colony), or they can pursue their own interests (as in the free market economy).

Agents are autonomous or semi-autonomous hardware or software that carry out tasks in complex, continuously changing environments [Wooldridge, 2002]. In order to cope with the tasks for which they are designed, agents have the following basic capabilities:

- Reactive: agents must react in a timely and appropriately way to unplanned events and to changes in the environment;
- Goal oriented: agents act in a purposeful manner;
- Communicative: agents should be able to communicate with the environment and other agents;
- Adaptive: agents should be able to change their behaviour according to previous experience;
- Autonomous: agents must exercise control over their actions;
- Temporally continuous: agents must possess the ability to executing continuously.

In this thesis, MAS consist of a group of agents that can take specific roles within an organisational structure. Agents are represented as actors. Agents are persistent and have relatively long-lived goals representing the functional and non-functional aspects of what they are doing.

2.4.2 Non-Functional Requirements for Agents

Based on bibliographical research [Wooldridge and Jennings, 1995], [Weiss, 2000], [Yu, 2001], [Sturm and Shehory, 2003], the following set of agent properties (and their relationships) present in most agent-oriented systems are established as non-functional requirements for MAS [Silva et al, 2004]:

- ◆ *Autonomy*: ability of the software to act independently without direct intervention from humans or other agents. Active autonomous entities are not necessarily compliant with external demands or desires.
 - *Deliberativity*: A deliberative agent's actions are decided by considering both information from its environment and information about previous experiences. Such agents are able to generate goals and to act rationally in order to achieve them.
 - *Reactivity*: agents perceive their environment, and respond in a timely fashion to changes that occur.
- ◆ *Sociability*: the ability to participate in multiple relationships, interacting with a number of other agents, at the same time or at different times.
 - *Organisation*: the arrangement of relationships among agents that produces a unit endowed with qualities not perceived at the individual level.
 - *Interaction*: the ability to communicate with the environment as well as with other agents.
 - *Communication*: the ability to exchange messages with other agents
- ◆ *Competition*: the ability to interact with other agents where the success of one agent implies the failure of others (opposite of cooperation).
- ◆ *Cooperation*: the ability to interact with other agents to achieve a common purpose; this happens among non-antagonistic agents that either succeed or fail together.
 - *Coordination*: the ability to perform some activity in a shared environment with other agents such as common goals and tasks, avoiding unnecessary conflicts, and pooling knowledge and evidence.
 - *Negotiation*: the ability to interact with other agents in order to reach an agreement about some matter. It involves the exchange of information, the relaxation of initial goals, mutual concessions, lies or threats.

MAS represent an exciting new means of analysing, designing and building complex software systems. Some approaches to structuring multi-agent systems will be overviewed in next subsection.

2.4.3 Multi-Agent Organisation

With the increasing use of MAS, it is necessary to analyse and study MAS from the viewpoint of Software Architecture. The concepts of software architecture are important for modelling MAS in practice. However, few efforts have been directed toward a definition of an acceptable MAS architecture. This lack of definition may be due to the common

misconception in research cycles that MAS architectures and frameworks need to be designed from first principles to match project requirements [Wooldridge and Jennings, 1998].

Therefore, one might expect that the widespread adoption of MAS technology will only begin after the formalisation and standardisation of the architectures, mechanisms, and protocols that support the distributed interoperation of agents. Recently, research groups have started to pursue the standardisation of multi-agent technology, such as those of the Object Manager Group (OMG) and the Foundation for Physical Agents (FIPA):

- ◆ The OMG Model - The OMG group proposes a reference model as a guideline for the development of agent technologies [OMG, 2004]. This model outlines the characteristics of an agent environment composed of agents (i.e., components) and agencies (i.e., places) as entities that collaborate using general patterns and policies of interaction. Under this model, agents are characterised by their capabilities (e.g., inferencing, planning, and so on), type of interactions (e.g., synchronous, asynchronous), and mobility (e.g., static, movable with or without state). Agencies, on the other hand, support concurrent agent execution, security and agent mobility, among other functions.
- ◆ The FIPA Model - The Foundation for Intelligent Physical Agents (FIPA) is a multi-disciplinary group pursuing the standardisation of agent technology [FIPA, 2004]. This organisation has made available a series of specifications to direct the development of multi-agent systems. FIPA's approach to MAS development is based on a "minimal framework for the management of agents in an open environment." This framework is described using a reference model (which specifies the normative environment within which agents exist and operate), and an agent platform (which specifies an infrastructure for the deployment and interaction of agents).

Multi-agent models of organisation represent a way of looking at software architecture. Besides actors or agents, communication channels are distinguished as basic elements of the organisation. The structure of an organisation defines the roles of the various internal actors, their responsibilities, and the resources they have been allocated. Additionally, an organisation defines how to coordinate the activities of the various components and how they depend on each other. These dependencies may involve both actors of the organisation and its environment, such as partners, competitors or clients. Furthermore, there are dynamic processes of task allocation, cooperation, and communication. Based on these considerations, the following aspects of the architecture of multi-agent systems can be distinguished [Gazendam and Jorna, 1996]: *the topology of the components, the system function decomposition, and the way of coordination and synchronisation of processes*:

1. The *topology* of a multi-agent system is the way in which the basic components of an organisation are ordered in space and time. The basic components of an organisation are agents and resources. Their ordering in space and time takes the form of architectural styles.
2. The *system function decomposition* aspect of the architecture of an organisation is the way the organisation is composed of sub-organisations fulfilling a specific function. These sub-organisations can be decomposed into smaller sub-organisations, and so on. The main alternatives for system function decomposition found in the organisation

and management literature is functional system decomposition, check and balance decomposition, and subsystem decomposition. The 'functional' decomposition of organisations into smaller units is based on specialisation like production, marketing, finance, personnel, research, material management, and so on. An alternative to functional decomposition based on specialisation is the decomposition of functions in a web of actors that have to maintain a system of *checks and balances*, as in a social and political arena.

3. *The coordination and synchronisation of agent processes* is the next architecture aspect to discuss. Agent processes can be seen as being related to the wandering around of autonomous agents in a natural and agent-made environment, in which they cooperate in an occasional or regular way. Agent processes also can relate to perception, communication and learning.

The next sub-section will outline some multi-agent methods that support at least one organisational concept.

2.5 ORGANISATIONAL CONCEPTS IN MULTI-AGENT SYSTEM METHODOLOGIES

Over the last few years, the need for applicable and broadly accepted development methodologies for multi-agent system (MAS) has resulted in a large number of proposals. Although many agent-oriented methodologies have been developed, it is difficult to select a specific one or even to compare them since they usually address different agent properties and organisational concepts. This section will examine three approaches that support organisational concepts in MAS development: GAIA [Wooldridge et al, 2000], MaSE [Wood and DeLoach, 2001] and AALAADIM [Ferber and Gutknecht, 1998].

2.5.1 MaSE

MaSE is a methodology for analysing and designing heterogeneous multi-agent systems. The ultimate goal of MaSE is the automatic generation of code with respect to the original system specification [Wood and DeLoach, 2001]. The MaSE methodology comprises seven steps:

1. Capturing goals: The designer takes the initial system specification and transforms it into a structured set of system goals.
2. Applying use cases: Use cases drawn from system requirements are structured as a sequence of events and are used in translating goals into roles. This step results in an initial set of roles. A role in MaSE is an abstract description of an entity's expected function.
3. Refining roles: The designer ensures that all the necessary roles have been identified and develops the tasks that define role behaviour and communication patterns. Roles are captured in a Role Model that specifies roles and their interrelations. The

specification of roles comprises their identity and the tasks that roles must perform to accomplish their goals.

4. Creating agent classes: The designer specifies the agent classes that are identified from role specifications. Agent classes are defined in terms of roles they play and conversations in which they must participate. These classes are documented in an Agent Class Diagram.
5. Constructing conversations: Coordination protocols between pairs of agents are constructed.
6. Assembling agent classes: This is the step where the internal elements of agent classes are created.
7. System deployment: The designer defines the configuration of the current system to be implemented.

A role in MaSE is an abstract specification of an entity's expected function that comprises the tasks that should be performed for the role to accomplish its goals. The specification and interrelations of roles are defined during the design phase and do not change. Since each agent class is defined by the set of roles it plays, the roles' lifespan is equal to the lifespan of the agents in the system. Furthermore, the assignment of agent classes to roles is performed during the design phase. Although there is a one-to-one mapping between roles and agent classes, the designer may combine multiple roles in a single agent class or map a single role into multiple agent classes. Agents, as instances of agent classes, do not deliberate on role assignments. They are not able to revise the devised organisation.

In sum, roles in MaSE are used in the analysis phase and are substituted by the agent classes in the design phase.

2.5.2 GAIA

Gaia is intended to allow an analyst to go systematically from a statement of requirements to a design, which is sufficiently detailed to be implemented directly. Gaia, as stated in [Zambonelli et al, 2003], encourages developers to think of building agent-based systems as a process of organisational design. An organisation is considered to be a collection of roles that stand in certain relationships to each other.

A role in Gaia is defined by four attributes: *responsibilities*, *permissions*, *activities*, and *protocols*. Responsibilities determine the functionality of the role and are divided into *liveness* properties and *safety* properties. Liveness properties describe those states of affairs that must be brought about by the agent that has been assigned to the role. Safety properties describe those states of affairs that the agent must maintain across all states of execution of the role. In order to realise responsibilities, a role has a set of permissions that identify the resources that are available to that role. Activities are computations associated with a role and may be carried out by the agent without interacting with other agents. Finally, protocols define role interactions. The analysis stage of Gaia is based on the following steps:

1. Identification of the roles in the system. This gives a prototypical (informal and unelaborated) role model, defined as a set of role schemata. Each schema comprises protocols, activities, permissions and responsibilities.
2. Protocols of interactions among roles are identified and documented. This results in an interaction model, which captures the recurring patterns of inter-role interaction for task performance.
3. Full elaboration of roles model, using the protocol model.

During the design phase, role specifications guide the definition of agent types in the system. Roles are static and long-lived, because they define a fixed organisational structure and are statically associated with specific agent types. However, one or more instances of an agent type (which plays an associated role) can be created according to instance qualifiers introduced in the agent model. Instance qualifiers define the cardinality of roles, i.e., the number of agents of the same class that can be assigned in a specific role. However, agents cannot deliberate on the organisational structure and on the number of agents that should play a specific role.

Hence, systems developed with Gaia methodology have a fixed organisation structure that manages interaction and distributivity in a static fashion.

2.5.3 AALAADIN Model

AALAADIN is not a specific agent methodology, but a meta-model for describing organisations of agents using the core concepts of group, agent and role [Ferber and Gutknecht, 1998]. Multi-agent systems can be described in terms of different forms of organisations such as market-like or hierarchical organisations. An organisation in AALAADIN is a framework for activity and interaction through the definition of groups, roles, and their relationships. A recent work has extended the AALAADIN model: the organisation description consists of two aspects: a structural aspect and a dynamic aspect [Ferber et al, 2003].

In AALAADIN, a group is defined as an atomic set of agent aggregations and has a finite set of roles. Groups and roles are specified during systems development. A group can be created by any agent that then automatically takes the special role of group manager. The group manager has the responsibility for handling requests for group admission or role requests. A role in this model is an abstract representation of an agent function, service or identification within a group. Each agent can take several roles, and each role assigned to an agent is local to a group. The most interesting feature of AALAADIN is that an agent can create new groups with roles. Therefore, the structure of the system organisation can be constructed dynamically. Agents can create roles with transient lifetimes in a group structure. AALAADIN does not aim at cooperative behaviour. It just provides the designer with the above-mentioned facilities in order to build groups of agents.

To sum up, the model provides developers with the ability to define systems that can cope with task-environment dynamics in terms of agent groups and roles. The actual properties of roles depend on the final MAS design.

In this thesis, group model defines an organisational structure, and offers a set of modelling parameters that can influence the division of labour and the coordination mechanisms, thereby affecting how the organisations functions. Modelling parameters include role interactions, organisational styles, and task assignment.

Our framework is developed in the context of the Tropos project, an information system development that is requirement-driven in the sense that it adopts organisational concepts used during early requirements analysis. Tropos will be outlined next.

2.6 THE TROPOS PROJECT

Tropos is an information system development methodology that is requirements-driven [Mylopoulos et al, 2000 & 2001], [Castro et al, 2002], [Bresciani et al, 2004]. The methodology adopts the view of information systems as social structures. By social structures, we mean a collection of social actors, human or software, which act as agents, positions, or roles and have social dependencies among themselves. Tropos is intended as a seamless methodology tailored to describe both the organisational environments of a system and the system itself in terms of the same concepts.

The Tropos project offers a modelling framework for requirement analysis and a set of organisational styles inspired by organisation theory. The requirement models and the architectural styles are explained next.

2.6.1 Early and Late Requirement in Tropos Project

The Tropos ontology adopts concepts offered by the i* (i-star) organisational modelling framework [Yu, 1993 & 1995], such as actor, agent, position, role, and social dependency.

The i* models provide a framework for understanding the organisational environment and goals. The participants of the organisational setting are actors with intentional properties such as goals, beliefs, abilities and commitments. The term actor was used to refer generically to any unit to which intentional dependencies could be ascribed. An intentional actor does not simply carry out activities and produce entities, but has motivations, intents, and rationales behind its actions. Actors are strategic in the sense that they are not merely focused on meeting their immediate goals, but are also concerned with opportunities and vulnerabilities, and seek rearrangements of their environments that would better serve their interests [Yu, 1993 & 2001].

Modelling a finer group of dependencies will help to identify more precisely how one dependency might lead to other dependencies. Actors depend on each other for goals to be achieved, tasks to be performed, and resources to be furnished. They also depend upon each other in order to fulfil their objectives and have their tasks performed. In sum, Tropos has the following concepts, as seen in Figure 3:

- Actor: An actor is an active entity that carries out actions to achieve goals by exercising its know-how. Actors are represented as circles;

- **Dependency:** A dependency describes an intentional relationship between two actors, i.e., an “agreement” between two actors: the depender and the dependee, where one actor (depender) depends on another actor (dependee) on something (dependum). The dependum is the type of the dependency and describes the nature of the agreement. Dependencies have the form *depender* → *dependum* → *dependee*;
- **Goal:** A goal is a condition or state of affairs in the world that the stakeholders would like to achieve. How the goal is to be achieved is not specified, which allow alternatives to be considered. Goal dependencies are used to represent delegations of responsibility for fulfilling a goal. Goals are represented as ovals;
- **Softgoal:** A softgoal is a condition or state of affairs in the world that the actor would like to achieve. But unlike with the concept of (hard) goal, there are no clear-cut criteria for whether the condition is achieved, and it is up to the subjective judgement and interpretation of the developer to judge whether a particular state of affairs achieves the stated softgoal sufficiently. Softgoal dependencies are similar to goal dependencies, but their fulfilment cannot be defined precisely (for instance, the appreciation is subjective, or the fulfilment can occur only to a given extent). Softgoals are represented as clouds;
- **Resource:** A resource is an entity (physical or informational) about which the main concern is availability. Resource dependencies require the dependee to provide a resource to the depender. Resources are represented as rectangles;
- **Task:** A task specifies a particular way of doing something. Tasks can also be seen as the solutions in the target system. These solutions provide operations, processes, data representations, structuring, constraints and agents to meet the needs stated in the goals and softgoals. Task dependencies are used in situations where the dependee is required to perform a given activity. Tasks are represented as hexagons.

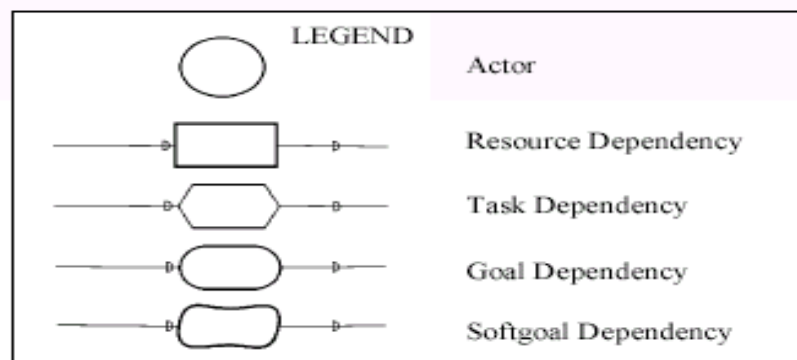


Figure 3 - Elements from i* framework

The i* technique consists of two models: The Strategic Dependency Model (SD) and Strategic Rationale Model (SR). SD includes a set of nodes and links connecting them, where nodes represent actors and each link indicates a dependency between two actors. SR is used

to: (i) describe the interests, concerns and motivations of participants in the process; (ii) enable the assessment of the possible alternatives in the definition of the process; and (iii) research in more detail the existing reasons behind the dependencies among the actors.

2.6.1.1 The Strategic Dependency model

The Strategic Dependency Model (SD) includes a set of nodes and the links connecting them, where nodes represent actors and each link indicates a dependency between two actors. Hence, a model is described in terms of a network of dependency relationships among various actors, capturing the motivation and why of activities.

SD encourages a deeper understanding of a business process by focusing on intentional dependencies among actors, beyond the usual understanding based on activities and entity flows. It helps identify what is at stake, for whom, and what impacts are likely if a dependency fails. Although a strategic dependency model provide hints about why a process is structured in a certain way, it does not sufficiently support the process of suggesting, exploring, and evaluating alternative solutions. That is the role of the Strategic Rationale (SR) model.

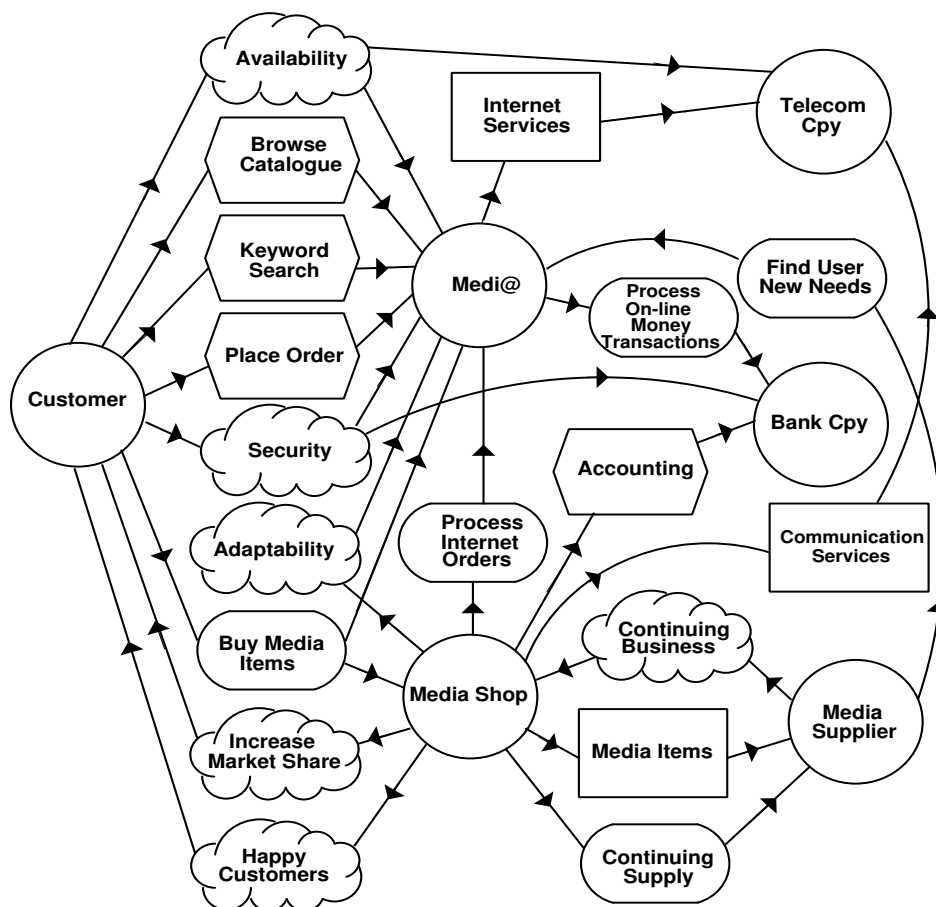


Figure 4 - SD model for Media Shop

Figure 4 displays the SD model of an e-commerce example. Media Shop is a store selling and shipping different kinds of media items such as books, newspapers, magazines, audio CDs, videotapes, and the like. Media Shop customers (on-site or remote) can use a periodically updated catalogue describing available media items to specify their order. Media Shop is supplied with the latest releases from Media Producer and in-catalogue items by Media Supplier. To increase market share, Media Shop has decided to open up a B2C retail sales front on the internet. With the new setup, a customer can order Media Shop items in person, by phone, or through the internet. The system has been named Medi@ and is available on the world-wide-web using communication facilities provided by Telecom Cpy. It also uses financial services supplied by Bank Cpy, which specialises in on-line transactions. The basic objective for the new system is to allow an on-line customer to examine the items in the Medi@ internet catalogue and place orders (The full e-commerce definition can be found in [Castro et al, 2002]).

For our example, the Medi@ system is viewed as a fully-fledged actor in the strategic dependency model. With respect to the actors previously identified, Customer depends on Media Shop to buy media items while Media Shop depends on Customer to increase market share and make them happy (with Media Shop service). Media Supplier is expected to supply Media Shop with media items in a continuous way because the latter depends on the former for continuing business. It can also use Medi@ to determine new needs from customers, such as media items not available in the catalogue, while expecting Media Producer to provide her with quality packages. As indicated earlier, Media Shop depends on Medi@ for processing internet orders and on Bank Cpy to process business transactions. Customer, in turn, depends on Medi@ to place orders through the internet, to search the database for keywords, or simply to browse the on-line catalogue. With respect to relevant qualities, Customer requires that transaction services be secure and available, while Media Shop expects Medi@ to be easily adaptable (e.g., catalogue enhancing, item database evolution, user interface update,y). Finally, Medi@ relies on internet services provided by Telecom Cpy and on secure on-line financial transactions handled by Bank Cpy.

2.6.1.2 The Strategic Rationale model

The Strategic Rationale (SR) model is a supplementary model to the Strategic Dependency (SD) model. By explicitly representing means-ends relationships, it provides a systematic way for exploring the space of possible new process designs. This model allows modelling of the reasons associated with each actor and their dependencies. Each actor possesses external dependencies with other actors, but internally each possesses goals and tasks that impel and justify each dependency. A good way to begin the decomposition is to observe how the *dependee* actor can satisfy the *dependum* associated with the same, and then to investigate and decompose intentions and strategic organisational reasons as a whole.

A strategic rationale model is a graph with four main types of nodes -- goal, task, resource, and softgoal -- and two main types of links -- means-ends links and task decomposition links (see Figure 5 - pg.31). These links are described as:

- Means-ends link: This link is associated with the obtaining of a certain end, which can be a goal, resource, softgoal or task. The means to obtain the end is defined as tasks that are necessary to reach the end.

- Task-decomposition link: A node task is linked to its component tasks through a decomposition connection. The four types of nodes can be linked. A task may be decomposed into smaller units of a node to represent the way and the reasons associated with the accomplishment of the task.

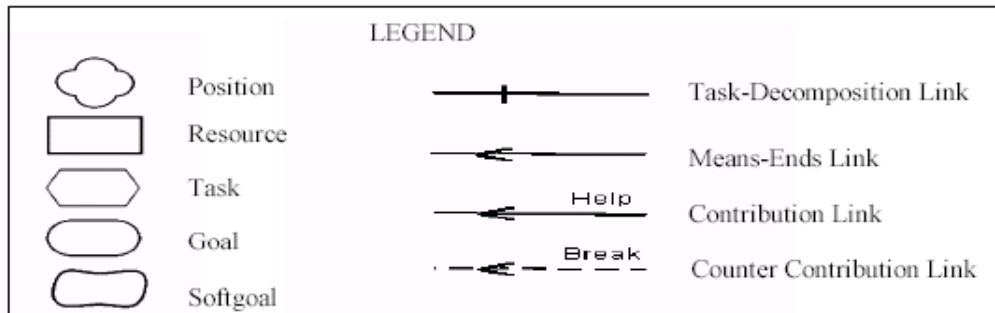


Figure 5 - Types of links in Tropos

When a process element is expressed as a goal in the Strategic Rationale model, it means that there might be different possible ways of accomplishing it. A task specifies one particular way of doing things (of accomplishing a goal), in terms of decomposition into sub-tasks, sub-goals, resources, and softgoals.

When redesign a business process, goals offer potential places to look for the process improvement. Higher goals are discovered by asking “why” questions. Once sufficiently high-level goals have been identified, alternatives may be sought by asking “how else” the goals can be accomplished. Each alternative may have different implications for a number of quality goals, or “softgoals”, such as adaptability, performance, and profitability. A softgoal is one that does not have a priori, clear-cut criteria of satisfaction. Although some of these can be measured and quantified, a qualitative approach can be used at the stage of exploring the space of alternatives. Contributions can be positive or negative, and are judged to be adequate or not. The treatment of softgoals is based on a framework developed for dealing with non-functional requirements in software engineering [Chung et al, 1995&2000].

Usually, when we try to understand an organisation, the information captured by standard modelling techniques (DFD, ER, Statechart, etc.) focuses on entities, functions, data flows, states and the like. These techniques are not capable of expressing the reasons (or “why”) for the process in form of motivations, intentions and rationales. The i* models cater for some of these advanced concepts. In cases where a particular organisation or system has to be improved, the analyst’s task is simplified by describing the *what* and *why* of a given organisation or system.

In architectural modelling, Tropos offers a set of organisational styles inspired by organisation theory. These organisational styles are useful to guide the architectural design of the Multi Agent Systems.

2.6.2 Multi-Agent Architectural Styles

Designers rely on styles, patterns or idioms to describe the architectures of their choice. An architectural style determines the vocabulary of components and connectors that can be used in instances of that style, together with a set of constraints on how they can be combined [Garlan and Shaw 1993], [Shaw and Clements, 1997]. Tropos has an organisational architectural catalogue defined as *organisations of agent* that cooperate to achieve common goals for cooperative, dynamic and distributed applications [Do et al, 2003], [Kolp et al, 2001 & 2003].

The architectural catalogue (*pyramid, joint venture, structure in 5, takeover, arm's length, vertical integration, co-optation, bidding*) is based on concepts and design alternatives coming from research on organisation management [Mintzberg, 1992]. The proposal is to use human organisations as a metaphor to suggest a set of generic styles for agent systems, with a preference for organisational design theories over social emergence theories.

The Tropos architectural styles are represented in i^* and are formalised in terms of *Formal Tropos specification language* [Fuxman et al, 2001a & 2003]. The Formal Tropos offers a textual notation for i^* (i-star) and describes the dynamic constraints among the different elements of the specification in a first order linear-time temporal logic.

In the next sub-sections, we only detail the Joint venture and the Structure-in-five styles here. These two styles will be used as examples in this thesis.

2.6.2.1 The joint venture style

The joint venture style is a more decentralised style based on an agreement between two or more principal partners who benefit from operating at a larger scale and reuse the experience and knowledge of their partners.

As seen in Figure 6, each principal partner is autonomous in a local dimension and interacts directly with other principal partners to exchange services, data and knowledge. However, the strategic operation and coordination of the joint venture is delegated to a Joint Management actor, who coordinates tasks and manages the sharing of knowledge and resources. Outside the joint venture, secondary partners supply services or support tasks for the organisation core.

In the joint venture style, the manipulation of partners can be done by registering new components to joint manager. Some constraints complement the basic style. 1) Partners depend on each other for providing and receiving resources. 2) Operation coordination is ensured by the joint manager actor, which depends on partners for the accomplishment of these assigned tasks. 3) The joint manager actor must assume two roles: a private interface role to coordinate alliance partners and a public interface role to take strategic decisions, define policy for the private interface, and represents the interests of the whole partnership with respect to external stakeholders

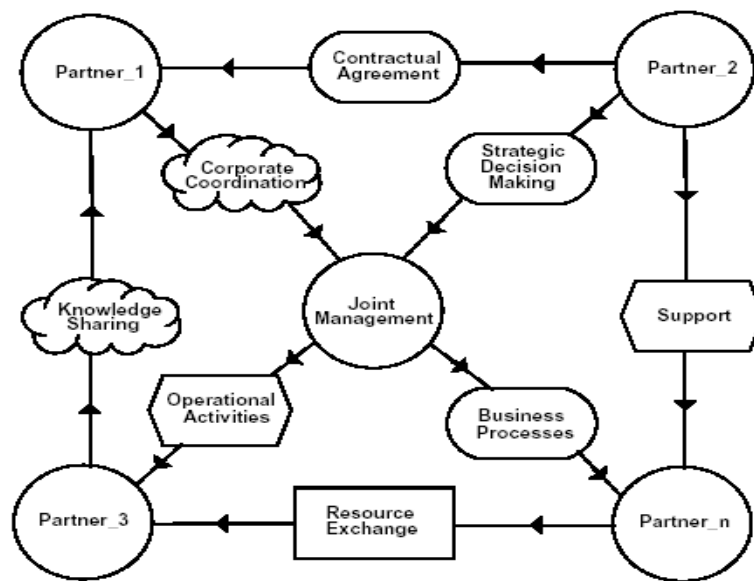


Figure 6 - The Join Venture Style

2.6.2.2 Structure-in-5 Style

The *structure-in-5* (Figure 7) is a typical organisational style based on five sub-units [Mintzberg, 1992].

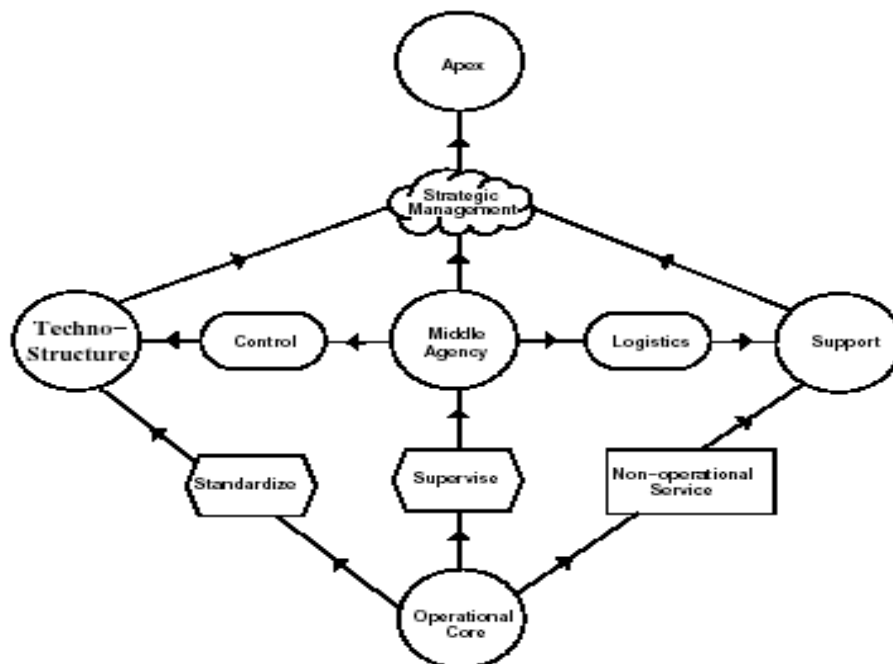


Figure 7 - The Structure-in-5 Style

This decomposition allows one to apply alternative coordination mechanisms (such as mutual adjustment, direct supervision, standardisation of skills, outputs and work processes) and design parameters (such as job specialisation, behaviour formalisation, decentralisation, unit size, and unit grouping) in order to analyse the different behaviours of the organisation. Basically, this structure defines a hierarchy of roles inside the organisation, the responsibilities associated with each sub-units, and inter-dependencies among them. At the base level, the Operational Core takes care of basic tasks — the input, processing, output and direct support procedures — associated with running the organisation. At the top lies the Apex, composed of executive actors. Below it, sit the Technostructure, Middle Agency and Support actors, who are in charge of control/standardisation, management and logistics, respectively. The Technostructure component carries out the tasks of standardising the behaviour of other components, in addition to applying analytical procedures to help the organisation adapt to its environment. Actors joining the apex to the operational core make up the Middle Agency. The Support component assists the operational core for non-operational services that are outside the basic flow of operational tasks and procedures.

The research on software architectures overlaps with work in requirements engineering. However, the need to treat non-functional properties explicitly is a critical issue when software architecture is built. Architectural definitions should provide the required functionality for the software system. Moreover, the architecture should be in conformity with the system quality requirements (non-functional requirements). These requirements include among others performance, safety and trust requirements.

In [Giorgini et al, 2002] there is an evaluation of conventional architectural styles (layered, control loop and task tree) and organisational architectural styles (structure-in-5 and joint venture) with respect to a set of agent software qualities, including predictability, adaptability, and openness. In fact, organisational architectures integrate non-functional requirements with architectural design, since certain non-functional requirements can be associated with these styles.

2.6.3NFR Framework and Architectural Selection

Unlike functional requirements which define what a software is expected to do, non-functional requirements specify global constraints on how the software operates or how the functionality is exhibited. Non-functional requirements are as important as the functional ones. They are not simply desirable qualities, but also critical aspects of dynamic systems without which the applications cannot work and evolve properly.

Because quality requirements are subjective by nature, they are usually achieved not in an absolute sense, but to a sufficient or satisfactory extent (the notion of *satisficing*). Accordingly, the Non-Functional Requirements (NFR) Framework introduces the concept of *softgoals*, whose achievement is judged by the sufficiency of contributions from other (sub-) softgoals. Throughout the development process, the consideration of design alternatives, the analysis of design tradeoffs and the rationalisation of design decisions are all carried out in relation to the stated softgoals and their refinements [Chung et al, 1995 & 2000], [Chung 1998b].

The NFR Framework will guide the selection process of the appropriate architectural style using as criteria the desired qualities (NFR) identified by the requirements models. The analysis involves refining these qualities, represented as softgoals, to sub-goals that are more specific and more precise and then evaluating alternative architectural styles against them. Each style contributes positively/negatively to some of the identified subgoals. The overall evaluation of a pattern with respect to a quality is arrived at by propagating contributions from bottom towards the top of a softgoal dependency graph.

In the example in Figure 4 (the SD model for Media Shop - page 29), the software quality attributes Security, Availability and Adaptability have been included in the late requirement models. They guide the selection process of the appropriate architectural style. The analysis involves refining these qualities, represented as softgoals, into sub-goals that are more specific and then evaluating alternative architectural styles against them.

The resulting dependency graph is shown in Figure 8: The organisational patterns are represented as operational attributes fulfilled by the joint-venture pattern. Design rationale is represented by claim softgoals drawn as dashed clouds. They make it possible for domain characteristics such as priorities to be considered and properly reflected into the decision making process. Exclamation marks (! and !!) are used to mark priority softgoals. A check mark “√” indicates a fulfilled softgoal, while a cross “X” labels an unfulfillable one. The evaluation results in contribution relationships from the social structures to the quality attributes, labelled “+”, “++”, “-”, “--” that mean respectively *partiallysatisfied*, *satisfied*, *partially denied* and *denied*.

In Figure 8, Adaptability is AND-decomposed into Dynamicity and Updatability. For the e-commerce example, dynamicity deals with the way the system can be designed using generic mechanisms to allow web pages and user interfaces to be dynamically and easily changed. Updatability should be treated as strategically important for the viability of the application, the stock management and the business itself since Media Shop employees have to update the catalogue for inventory consistency very often.

Availability is decomposed into Usability, Integrity and Response Time. Network communication may not be very reliable so causing sporadic loss of the server. There may be data integrity concerns with the capability of the e-business system to do what needs to be done, as quickly and efficiently as possible: in particular with the ability of the system to respond in time to client requests for its services. It is also important to provide the customer with a usable application, i.e., comprehensible at first glimpse, intuitive and ergonomic. Equally strategic to usability concern is the portability of the application across browser implementations, and the quality of the interface. Security is decomposed into Authorisation, Confidentiality and External Consistency. Clients, exposed to the Internet are, like servers, at risk in web applications.

When the contribution relationships for each architectural style to the different non-functional requirements of the system have been assigned, the best-suited architectural style is chosen. This decision involves the categorisation of the softgoals according to their importance to the system and the identification of the architectural style that best satisfies the most important non-functional requirements. Eventually, non-functional requirements analysis allows us to choose the joint venture architectural style for our e-commerce example (the operationalised attribute is marked with a “√”).

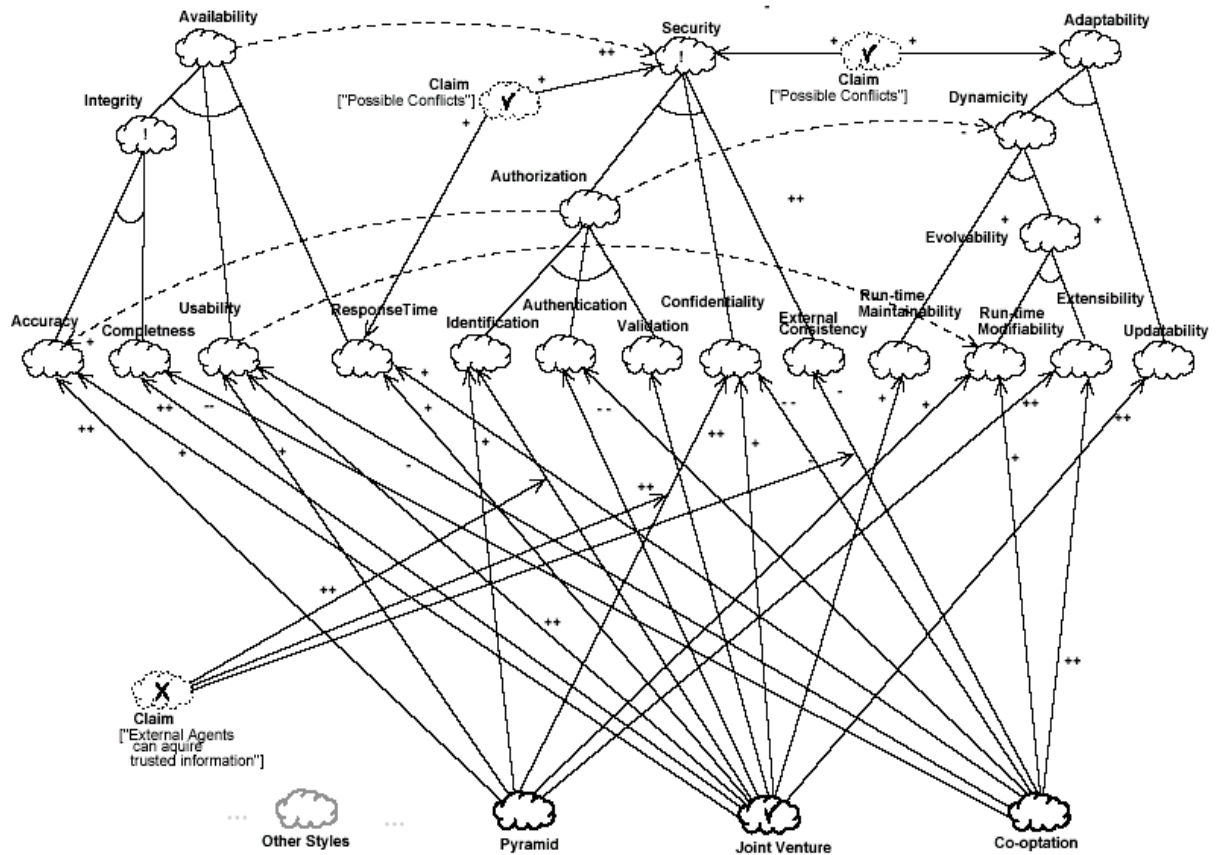


Figure 8 - Selecting architectural style in Tropos

Approaches to bridge requirements and architecture models have to find the right balance between flexibility and formality to be beneficial for all stakeholders involved in requirements engineering and system architecture.

The next sub-section discusses the main concepts and researches presented in this chapter.

2.7 PERSPECTIVES OF MULTI-AGENT SYSTEMS APPROACHES

Software systems demand special care in the requirement and architectural modelling phases. Requirements and system architecture emerge in an iterative process that involves heterogeneous stakeholders with conflicting goals, expectations, and objectives.

Additionally, information systems can be seen as built in order to help organisations achieve their goals. A number of goal-based requirement approaches, most notably KAOS [Lamsweerde, 2001] and the Tropos [Castro et al, 2002] have proposed the explicit use of the notion of 'goals' to structure system requirements and architecture. Kaos and Tropos are both

goal-based methods and focus on the main concepts of organisational modelling. The KAOS approach is essentially goal-driven and its method describes the use of goals to define requirements and considers how to deal with obstacles in a goal-modelling framework. In KAOS, it is not clear how to address the organisational context in early requirements analysis, how the non-functional proprieties can influence the selection of architectural artefacts, and how this selection can influence the agents' identification (for example, system components) and goals attributions.

Tropos has a different approach to goal modelling. Tropos is requirement-driven and its methodology covers the software development process as a whole, from the early steps, in which the software engineer models the requirements of the organisational setting (early requirements) and of the system-to-be (late requirements), up to detailed design. The methodology uses the same concepts, such as actor, goal, softgoal, and goal dependencies, in all phases. Moreover, Tropos can also be combined with non-agent (e.g., object-oriented or imperative) software development paradigms. For example, one may want to use Tropos for early development phases and then use UML for later phases [Alencar et al, 2001] [Santander and Castro, 2002a & 2002b].

In Multi-agent research, the concepts hold great promise for responding to the new realities of autonomous information systems. They offer higher level abstractions and mechanisms which address issues such as knowledge representation and reasoning, communication, coordination, cooperation among heterogeneous and autonomous parties, goals, beliefs and intentions. Their rich representational capabilities allow more faithful and flexible treatments of complex organisational processes, leading to more effective requirements analysis and architectural modelling.

Organisational approaches to the design of multi-agent systems are gaining ground in agent research and several research groups have presented models for agent societies. These include AALAADIN [Ferber and Gutknecht, 1998], MESSAGE [MESSAGE, 2004], MaSE [Wood and DeLoach, 2001] and GAIA [Zambonelli et al, 2003]. Research in MAS shows that roles can represent system goals and constrain agent behaviours [Partsakoulakis and Vouros, 2004]. Concepts such as organisational rules [Zambonelli, 2002], norms and institutions [Dignum and Dignum, 2001], [Esteva et al, 2001], and social structures [Parunak and Odell, 2002] arise from the idea that the effective engineering of MAS needs high-level, agent-independent concepts and abstractions that explicitly define the organisational structure in which agents act [Zambonelli et al, 2001].

In spite of the significant progress accomplished in the areas of requirement specification and architectural model, we still need frameworks, techniques and tools to support the systematic achievements of multi-agent architecture in the context of requirement models.

2.8 SUMMARY

This chapter has provided an overview of the basic theories and related research areas to help gain a better understanding of the concepts and ideas described in following chapters. In particular, it has examined the use of organisational analysis techniques within Requirement Engineering and Software Architecture in the context of Multi-Agent Systems. The main

aspects of each research area were described and ways to concern the integration among these areas were discussed.

In addition, the Tropos Project was presented. Tropos has a requirement driven methodology within which our framework is located. Tropos advocates that MAS can benefit from new organisational architectural styles, which are based on concepts and design alternatives coming from research on organisation management. These architectural styles support the design of cooperative, dynamic and distributed MAS applications.

Following chapters will present our approach to enhance the requirement model with organisational concepts for the derivation of organisational architectures in context of the Tropos project.

Chapter 3

The SIRA Framework - Systematic Integration between Requirements and Architecture Using Organisational Concepts

This chapter introduces the SIRA Framework. Our approach is based on two complementary models and a process. The process focuses on a systematic way to carry out the transition from requirement model to architectural configuration.

3.1 INTRODUCTION

Requirements engineering and software architecture are established areas of research, education and practice in the software engineering community. In spite of the considerable progress on these two fronts, we still need the solid basis, techniques and tools to support the systematic accomplishment of architectural objectives within the context of complex stakeholder relationships.

To narrow the integration between requirements and architecture, this chapter presents a framework to identify and map key components and interactions, based on the stated system requirements and organisational concepts. The *SIRA* Framework emphasises the organisational environment and helps to reduce the gap among Multi-Agent Systems (MAS) requirement models and architectural models. Our approach advocates that MAS correspond to organisational structure, in which actors are members of a group in order to perform specific tasks.

This chapter is organised as follows. Section 3.2 locates this work in the context of the Tropos Project. Section 3.3 describes the basic structure of the *SIRA* Framework. Section 3.4 introduces the *SIRA* Process. Finally, section 3.5 concludes the chapter.

3.2 THE *SIRA* FRAMEWORK IN TROPOS

The *SIRA* (Systematic Integration between Requirements and Architecture) framework describes a software system from the perspective of an organisation and is located in the context of the Tropos project, as show in Figure 9. The Tropos methodology spans four phases:

1. Early requirements - concerned with the understanding of a problem by studying an organisational setting; the output is an organisational model that includes relevant actors, their goals and dependencies.
2. Late requirements - the system-to-be is described within its operational environment, along with relevant functions and qualities.
3. Architectural design - the system's global architecture is defined in terms of sub-systems, interconnected through data, control and dependencies.
4. Detailed design – the behaviour of each architectural component is defined in further detail.

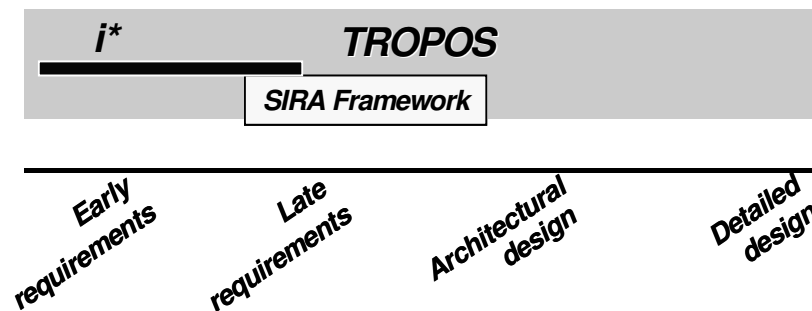


Figure 9 – The SIRA Framework in Tropos

The organisational view extracted from Strategic Dependency (SD) and Strategic Rationale (SR) models is used to capture system-related goals. Tropos does not explicitly cover the correlation between requirement models and architectural elements. Moreover, the information available in the late requirement models is not enough to derive architectural information. In this context, the SIRA framework provides a set of complementary models to specify properties of organisational groups as well as to help to derive of architectural properties.

The SIRA framework is described in the following.

3.3 THE SIRA FRAMEWORK

The SIRA (Systematic Integration between Requirements and Architecture) Framework provides an organisational model to specify the structure and global behaviour of the members that populate the multi-agent systems. The framework is meant to describe systems at the conceptual level and represents the agents organisation in a way that:

1. Is independent of the internal design of the agents;
2. Distinguishes organisational characteristics from individual agent characteristics;
3. Creates relation links between organisational the model and architectural components (or agent populations).

The SIRA framework is based on two complementary models that separate the concerns of the social organisation from those of the architectural organisation, as seen in Figure 10:

1. In the Organisational Model, the main goals are identified by understanding a requirement model as the functionality requested for the system. A goal is refined into sub-goals or tasks so that each sub-goal requires the cooperation of fewer roles. The organisation of the social system consists roles and interactions, as intended by the system and its environment. Additionally, the goals and softgoals define the selection of an architectural style.

2. In the Assignment Model, roles are clustered into sub-groups. These sub-groups are related to components, based on their similarity with the architectural components. The result is an architectural configuration, which is the allocation of sub-groups to architectural components.

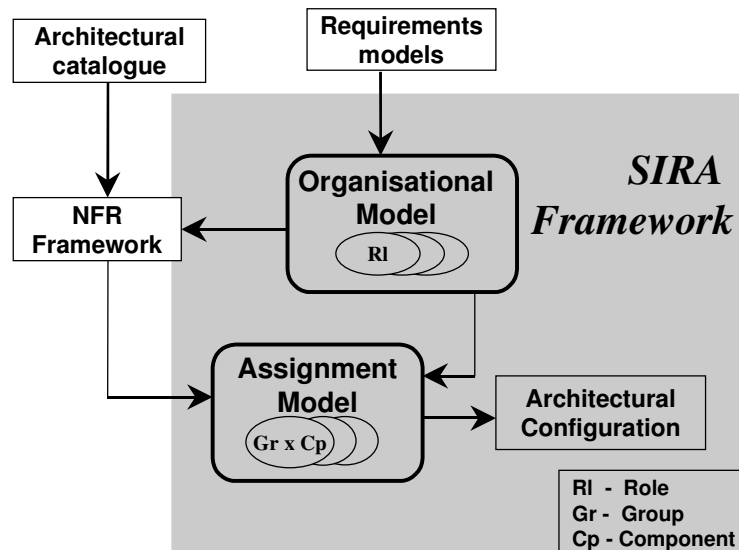


Figure 10 – The SIRA Framework

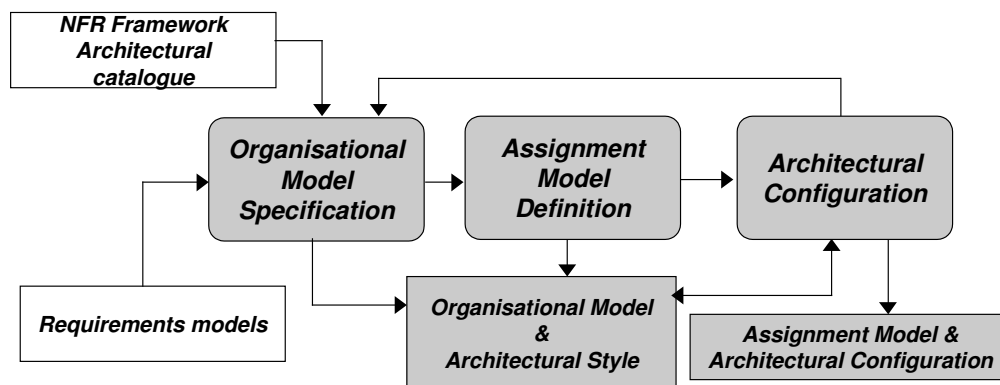
Our framework advocates that a system corresponds to an organisational structure, in which actors are members of a group, playing roles in order to perform specific tasks. Roles can be used both as an intuitive concept in order to analyse requirements in multi-agent systems as well as a behavioural structure in order to derive coherent software architectures.

Furthermore, the SIRA Framework provides the SIRA Process to accomplish the transition from requirements to architectural configuration. This process will be overviewed in the next section.

3.4 THE SIRA PROCESS

In software development, the transition from requirements to architecture is one of the most complex problems. This transition can be defined as a set of activities to analyse and to accomplish each decision involved. The *SIRA Process* focuses on a systematic way to assist the transition from requirement models to an architectural configuration. The aim is to develop a practical set of activities applicable to all the steps in modelling multi-agent systems based on the observation of the behaviour of an actor role in an organised group. The SIRA Process takes the organisational perspective as starting point and specifies some steps to derive multi-agent architectures for a particular domain. It consists of three activities (see Figure 11):

1. The first activity, *Organisational Model Specification* takes as input requirements models and an architectural catalogue to generate the *SIRA Organisational Model*.
2. The second activity is *Assignment Model Definition*. This activity clusters actors into sub-groups and matches sub-group and components;
3. The third activity, *Architectural Configuration*, links the *SIRA Organisational Model* to architectural style. This activity assigns sub-groups to architectural component and generates the architectural model of the multi-agent.

Figure 11 - The *SIRA* Process Activities

Each activity will be detailed in the next sub-sections.

3.4.1 The *Organisational Model Specification*

The first activity of the *SIRA* process is concerned with the specification of the Organisational Model. This activity supports the mapping from the *i** requirement model onto *SIRA* Organisational Model.

The organisational model refines the resulting system into a more precise model. Goals, softgoals and tasks of the requirement model determine the behaviour of the group and its relationship with the environment. These requirements are the basis for an organisational context, which is used to identify a social group (*in the form of sub-groups, roles and interactions*) that Multi-agent Systems have to provide. Additionally, some quality attributes for multi-agent architectures are identified from the perspective of a domain (e.g. e-commerce) and organisational styles.

As seen in Figure 12, the input for this activity is a requirements model and an architectural catalogue. As output we have the *SIRA* organisational model to complement the information of requirements models with roles and interactions. This activity includes three sub-activities: Goal and Task Refinement, Role Identification, and Architectural Selection. Each sub-activity will be outlined in the following sub-sections.

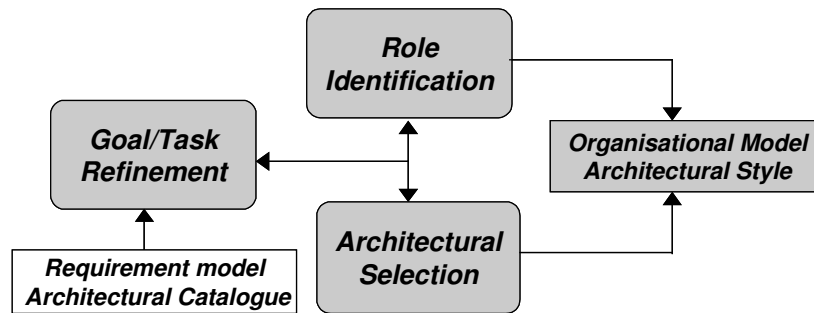


Figure 12 – The Organisational Model Specification

3.4.1.1 Goal and Task Refinement

One way to express the behaviour of an actor with respect to its environment is in terms of the expected functionality of the organisational group, that is, what the group is expected to do or produce. In the *Goal and Task Refinement* sub-activity, actors are *Organisational Groups* with responsibilities to be performed. The group responsibilities are identified from main dependencies among actors of the requirement models.

In Tropos, goal analysis is applied to early and late requirement models in order to refine them and to elicit new dependencies by using means-end analysis. Means-Ends Analysis is a problem solving strategy first introduced in GPS (General Problem Solver) [Newell and Simon, 1963]. The search process over the problem space combines aspects of both forward and backward reasoning in that both the condition and action portions of rules are looked at when considering which rule to apply. The basic idea that underlies Means-Ends Analysis is that the planner has the ability to compare two problem states and determine one or more ways in which these problem states differ from each other

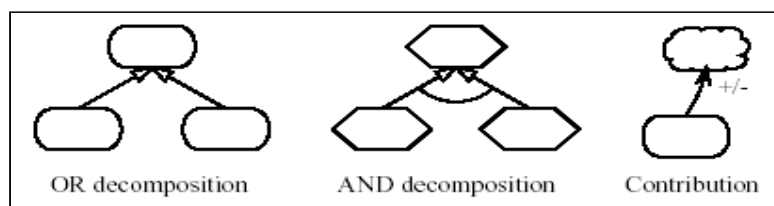


Figure 13 - Tropos representation of goal refinement with means-end and contribution analysis.

The goal analysis is conducted from the point of view of an actor, by using three basic reasoning techniques: means-end analysis, contribution analysis, and AND/OR decomposition (see Figure 13). In more detail, means-end analysis aims at identifying tasks, resources and softgoals that provide means for achieving a goal. Contribution analysis identifies goals that can contribute positively (+) or negatively (-) in the fulfilment of the goal to be analysed. In a sense, it can be considered as an extension of means-end analysis, with goals as means.

AND/OR decomposition-combines AND and OR decompositions of a root goal into sub-goals, modelling a finer goal structure [Bresciani et al, 2004].

3.4.1.2 Role Identification

The organisational structure defines the roles of various actors, their responsibilities, defined in terms of tasks and goals they have assigned and resources they have been allocated. In the Multi-Agent System, the situation can be seen in similar way, in that components are members of this MAS organisation and in a particular position may have multiple roles such manager, specialist, or subordinate depending on the task they are currently undertaking, and particularly with whom they interacting. MAS structure can be viewed as a collection of roles, that stand in certain relationships to one another, and that take part in systematic interactions with others roles.

The Role Theory can also be of some use as it is widely applied for group modelling, postulating that individuals occupy positions in an organisation [Biddle and Thomas, 1979]. Roles encapsulate actor responsibility and are identified by grouping “related” tasks. The analyst, together with users and other stakeholders, defines and allocates tasks for roles within the organisation. Role identification is done by means of group tasks. Grouping tasks is guided by the following principles:

1. Task cohesion: tasks with a high degree of dependence between them may be grouped together;
2. Low role coupling: there is a low degree of dependence between roles;
3. Goal coherence: roles have a limited number of defined goals.

In the *SIRA* Framework, the organisational group is redefined into a set of roles and interactions in order to represent the division of labour among members. *Role Identification* sub-activity is concerned with the identification of the sequence and types of tasks that can be performed by an actor in collaboration with others. Here, collaboration is a special type of coordinated activity in which actors work jointly with each other, to carry out the set of tasks needed to achieve a common goal. Every role communicates and interacts with other roles. So, role identification involves the distribution of task and interactions to perform each group goal in a coordinated way.

3.4.1.3 Architectural Selection

Software architectures have been the focus of considerable research, which has resulted in a collection of well-understood architectural styles and a methodology for evaluating their effectiveness with respect to particular software qualities.

An organisational architecture defines the organisational structure, and offers a set of design parameters that can influence the division of labour and interactions, thereby affecting how the organisation functions. Design parameters include, among other things, responsibility assignment, standardisation, supervision, and controls.

The organisational styles from Tropos (Section 2.6.2 - pg.32) are evaluated and compared with the non-functional properties of MAS architectures. Table 1 summarises the correlation catalogue for the organisational patterns and top-level quality attributes considered in [Kolp et al, 2001 & 2003]. The following notation used by the NFR (non-functional requirements) framework +, ++, -, --, models partial/positive, sufficient/positive, partial/negative and sufficient/negative contributions, respectively. When the contribution relationships for each architectural style to the different non-functional requirements of the system have been assigned, the best-suited architectural style is chosen. This decision involves the categorisation of the softgoals according to the importance to the system and identification of the architectural style that best satisfies the most important non-functional requirements in the business domain [Chung et al, 2000].

Table 1 - The Correlation catalogue

CORRELATION	PRED.	SEC	ADAPT.	COOP.	COMP.	AVAIL.	INTEG.	MOD.	AGGR.
Flat	--	--	-			+	+	++	-
Struct-5	+	+		+	-	+	++	++	++
Pyramid	++	++	+	++	--	+	--	-	
Joint-Vent	+	+	++	+	-	++		+	++
Bid	--	--	++	-	++	-	--	++	
Takeover	++	++	-	++	--	+		+	+
Arm's-Length	-	--	+	-	++	--	++	+	
Hierarchical Ctr			+	+	+	+		+	+
Vertical Integration	+	+	-	+	-	+	--	--	--
Cooptation	-	-	++	++	+	-	+		

Once the architectural style has been chosen, it has to conform to the functional and non-functional requirements of the application domain. Additionally, a specific refinement, resolution or assignment is selected on the basis of the qualitative preferences dictated by positive contributions to high-priority softgoals. In the NFR framework, qualities are represented as *softgoals*. Analysing them involves refining these softgoals into sub-goals that are more specific and more precise, and then evaluating alternative architectural styles against them. For each type of alternative, decisions have to be made which, in the end, will produce different architectures.

3.4.2 Assignment Model Definition

The Assignment Model Definition activity proposes the social network analysis as a method for framing and describing the effects of organisational characteristics to relate multi-agent systems and organisational architectures.

The basic idea of a social network is very simple. A social network is a set of actors (or roles, or agents) that may have relationships (or interactions) with one another [Hanemman, 2005], [Wasserman and Faust, 1994], [Zack, 2000]. Networks can have few or many actors, and one or more kinds of relations between pairs of actors. To build a useful understanding of a social network, a complete and rigorous description of a pattern of social relationships is a necessary starting point for analysis. That is, ideally is necessary to know about all of the relationships between each pair of actors.

Our Assignment Model describes the ways of clustering roles into sub-groups, which are mapped into organisational architecture using social network analysis. The structural models of the social analysis focus on the structure of a network, using different indices analysis [Hanemman, 2005], [Wasserman and Faust, 1994], [Zack, 2000]. Typical examples of structural indices are:

- **Centrality:** The centrality degrees indicate roughly the ability of an actor to reach the other actor of the organisational group. From the individual measure of centrality, a global measure of centralisation of the group can be evaluated. In SIRA Framework, centrality measures the strength of relations among roles. In other words, greater the number of relations, higher is the degree centrality of the role.
- **Structural equivalence:** Structural equivalence of two actors is the degree to which the two actors have the same profile of relations to all other actors in the group. This technique enables us to get a view of how the actors (in one group or in distinct groups) are "approximately equivalent".

Our objective is to develop ways to quantify the actor relations of the organisational group and to use these measures to test:

- Actors reflecting more influences or respond to more interventions;
- Differences in organisational sub-structures;
- Similarity with respect to relations of sending and receiving;
- Correlation of organisational group and architecture style measures,

The Assignment Model Definition activity includes two sub-activities (depicted in Figure 14): Cluster Analysis and Correlation Analysis. The input to this activity is an organisational model and an architectural style. Firstly, the cluster analysis groups the roles that have the same pattern of relations into sub-groups. Finally, the similarity analysis measures the linear relationship between the sub-groups and components.

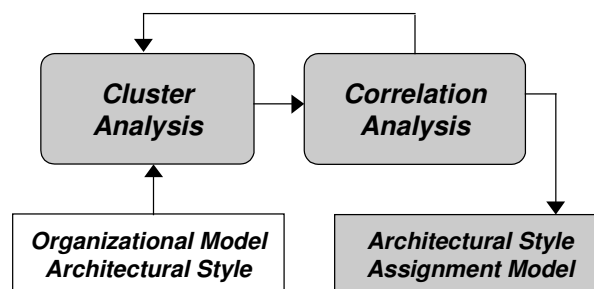


Figure 14 - Assignment Model Definition

3.4.2.1 Cluster Analysis

The *Cluster Analysis* sub-activity is one way of to look for sub-structures inside a group. The term *cluster analysis* encompasses a number of different methods for grouping objects of similar kind into respective categories. In our approach, cluster analysis is an exploratory analysis, which aims at sorting different roles into sub-groups in a way that the degree of association between two roles is maximal if they belong to the same sub-group and minimal otherwise.

In the Assignment Model, the cluster analysis is used to discover functional sub-structures of the Organizational Model. These functional sub-structures are identified as sub-groups. A concern is to cluster together roles that are the most similar, recalculates similarities, and iterates until all pair of roles is analysed. These sub-groups will be compared to architectural components with respect to the set of roles that they perform.

The *Cluster Analysis* measures the extent to which a role is connected to other roles in terms of centrality and structural equivalence. The centrality indices are used to measure how an actors are central (in terms of its relations) in the group and how close an actor is to the others in the group. The structural equivalence is used to measure the similarity coefficient of the relations for each pair of actors. The distributions of these two indicators among the whole group give insights about to which degree social group are clustered and how close two actors are on average.

3.4.2.2 Correlation Analysis

In the *SIRA* framework, the way of examining the correlation between two groups is to look for patterns of relations, and then comparing their behaviour.

Correlation analysis is concerned with measuring the degree of association between two actors. Our approach examines the correlation between sub-group and components in terms of centrality and similarity correlation.

In the centrality correlation, we compare the centrality degree and closeness measures of each pair of actors in the sub-group and architecture style. The first step is to analyse the measures of the most central actor in each group. After that, the centrality measures of the each actor in organisational group are compared with the centrality measures of each component in architecture. The result is the centrality correlation table.

In the similarity correlation, two actors that are structurally equivalent have the same pattern of relation to all other actors -- in this case, they are perfectly substitutable or exchangeable. In "real" data, exact similarity may be quite rare, and it may be meaningful to measure approximate similarity. There are a several approaches for examining the pattern of similarities in the relation-profiles of actors, and for forming structural equivalence groups. We measure the structural equivalence by calculating the Pearson correlation coefficient [Pearson, 2005]. The result is the similarity correlation table.

The correlation analysis is based on these two tables to match the *SIRA* Sub-groups and architectural components. Based on the centrality and similarity correlation tables, we analyse the correlation in terms of strong correlation (++), partial correlation (+) and no correlation ().

3.4.3 Architectural Configuration

This activity in the *SIRA Process* defines the map between requirement model and architectural style. The MAS architecture is the pattern in which agents, processes performed by agents, and resources produced by agents make up an organisational structure. The architectural configuration must support the MAS organisation, the independence of entities (hence reusability), the abstraction and analysis of properties ranging from functionality to security and reliability, and defines components and relationships.

A configuration is an instantiation of an architecture style with a chosen arrangement and an appropriate number of actors (or agents) of each type. One given architecture can lead to several configurations and a given configuration is tightly linked to the style and organisational structure and the application domain where the Multi-Agent System is deployed; thus, the architecture was modelled so that the set of possible configurations covers the different layouts foreseeable.

Our Architectural configuration describes the Multi-Agent System at a macroscopic level in terms of a manageable number of sub-groups using the organisational styles from Tropos. Multi-Agent Architectural configuration is a collection of components and a set of interactions between these components.

The architectural configuration is derived from the correlation analysis of the Assignment Model. The correlation analysis may results in degrees of similarity between sub-groups and components (in terms of partial, strong or no correlation). The correlation analysis should be refined in relation to the responsibilities that each sub-group and component should perform. The responsibilities of sub-groups are defined by the cluster analysis in the Assignment Model. The responsibilities of the architectural components are defined in the organisational catalogue. Sub-groups and components with the most similar correlation results are related into an architectural configuration of the Multi-Agent System.

3.5 SUMMARY

This chapter has presented our approach for modelling multi-agent systems, the *SIRA Framework*. Multi-Agent Systems (MAS) represent an exciting new means of analysing and designing complex software systems.

Additionally, we have outlined a process to assist the transition between requirements and architecture. The *SIRA Process* focuses on a systematic way to support the transition from requirement specification to architectural model. The *SIRA Process* consist of three main activities:

1. The first activity, *Organisational Model Specification* takes as input a Requirement model and an Architectural catalogue to generate the *SIRA Organisational Model*, in which actors are members of a group that performs roles in order to execute specific tasks.

2. The second activity is *Assignment Model Definition*. This relies on the use of an Architectural Catalogue and Non-functional Requirements (NFR) framework to select architectural styles. The Assignment Model addresses the clustering of roles into sub-groups and the correlation analysis between sub-groups and architectural components.
3. The third activity (*Architectural Configuration*) links the *SIRA Organisational Model* to an architectural style. This activity assigns sub-groups to each architectural component and generates the architectural configuration of the multi-agent.

The SIRA organisational and Assignment models will be detailed in the next chapters.

Chapter 4

The Organisational Model

This chapter introduces the organisational model for agent communities in the SIRA Framework. The organisational model can be thought of as a kind of abstract code of behaviour that governs how actor members should work according to organisational requirements.

4.1 INTRODUCTION

Organisational concepts can be thought of as a kind of abstract code of behaviour that governs how actor members should work according to organisational requirements, such as goals, norms and interactions. Organisational models, implicitly or explicitly, represent the goals of individuals, groups, or organisations, whereby a goal is a desired condition potentially attained at the end of an action or a process. Stakeholder goals and their role in defining and solving design problems are topics of longstanding interest in the field of requirements engineering. Our model emerges from the understanding, that in organisations interactions aim at achieving some desired global goals, and that members (actor and groups) are autonomous and heterogeneous.

This chapter is organised as follows. Section 4.2 introduces the bank example. The next sections give a description of two models: the Organisational Model is described in section 4.3, and the selection of the architectural style is introduced in section 4.4. Finally, section 4.5 concludes the chapter.

4.2 THE BANK EXAMPLE

A bank example will be used to present the different models of the SIRA framework. A look at the organisation of the bank system can illustrate the different components of an agent organisation. Banking is a business that deals with money and credit. Banking organisations have to provide the tools to handle all customer transactions. Banks hold money belonging to others, investing and lending it to get more money. They accept deposits from the public for safekeeping and pay interest on this money, and also lend this money to earn interest. More details about bank institutions and regulations can be found in BACEN (2004).

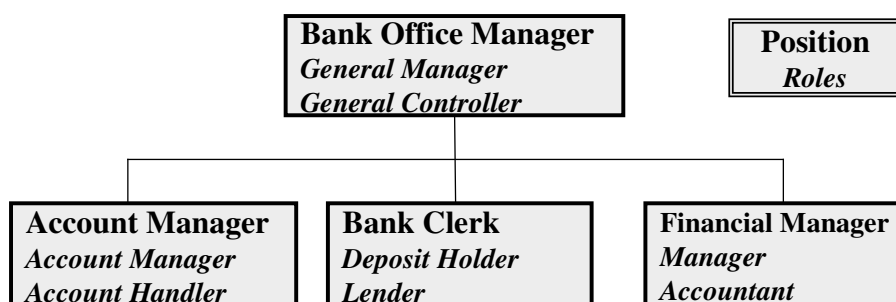


Figure 15 - The organisational chart of a bank office example

Figure 15 shows a traditional organisational structure (or organisational chart) of a bank domain. The positions in this organisation are Bank Office Manager, Account Manager, Bank

Clerk and Financial Manager. The objectives of the Bank Office Manager are to control the organisation and aim for higher profitability. The Account Manager has to monitor a set of customer accounts in order to offer new products and services. The Bank Clerk deals with deposits, withdrawals, and loans. The Financial Manager controls accounts and finances, and the customer hopes for a security and high quality services.

Banks can be described in terms of actors that administrate the organisation and actors that are responsible for monitoring customer accounts, and so on. The elements described so far are enough to produce the first model of an organisational environment. Figure 16 shows the i* organisational model of the bank example, in which Banks act as intermediaries between those who have money to hold and those who depend on money for business transactions. Bank Actor depends on the Customer's money (represented as the resource 'Money') for investment, and depends on customer to request loans (represented as the goal 'Get Loan') and to earn more money (represented as the softgoal 'Profitability'). Customer depends on Bank to hold his money in deposit or savings accounts (represented as the goal 'Deposit Money'), and to earn a certain amount of money called interest for depositing his money in a bank (represented as softgoal 'Deposit Profitability'). He also lets the bank use the money for new investments.

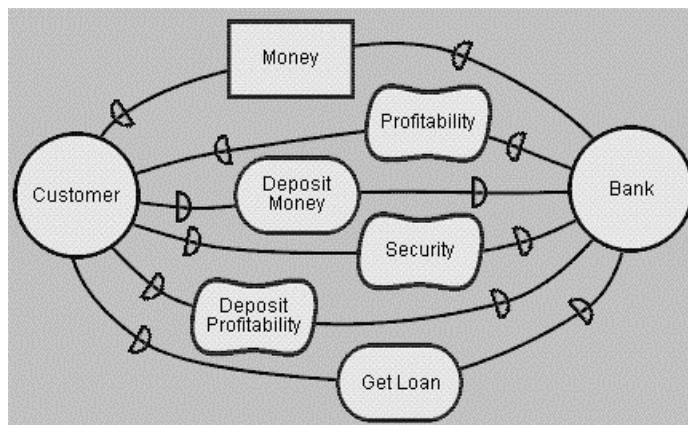


Figure 16 - Bank work example in i* notation

The bank domain defines the softgoal 'Security' for Bank Actor in terms of administrative, technical, and physical safeguards:

1. To insure the security and confidentiality of customer information;
2. To protect against any anticipated risks to the security and integrity of such information; and
3. To protect against unauthorised access to or use of such information, which could result in substantial harm or inconvenience to customers.

This bank example will be further detailed below, as the SIRA models are introduced.

4.3 THE ORGANISATIONAL MODEL

The SIRA organisational model considers a group as a collection of roles whose behaviour co-operatively determines the accomplishment of organisational goals. The meta-model of the SIRA organisational group is defined at the metaclass level in Telos, as a complementary set of i* (i-star) meta-model [Yu and Mylopoulos, 1994].

The Telos language adopts an object-oriented approach to knowledge representation. Classes can be classified under metaclasses, metaclasses under metametaclasses and so on. Furthermore, classes can be related through generalisation relations, which enforce the strict inheritance of attributes from superclasses to subclasses. A class may have more than one superclass, while generalisation relations may associate only classes at the same instantiation level (i.e. classes with classes, metaclasses with metaclasses and so on). These features make Telos an expressive language and justify its choice for describing the meta-concepts used to accommodate the knowledge relevant to a variety of information systems [Yu and Mylopoulos, 1994].

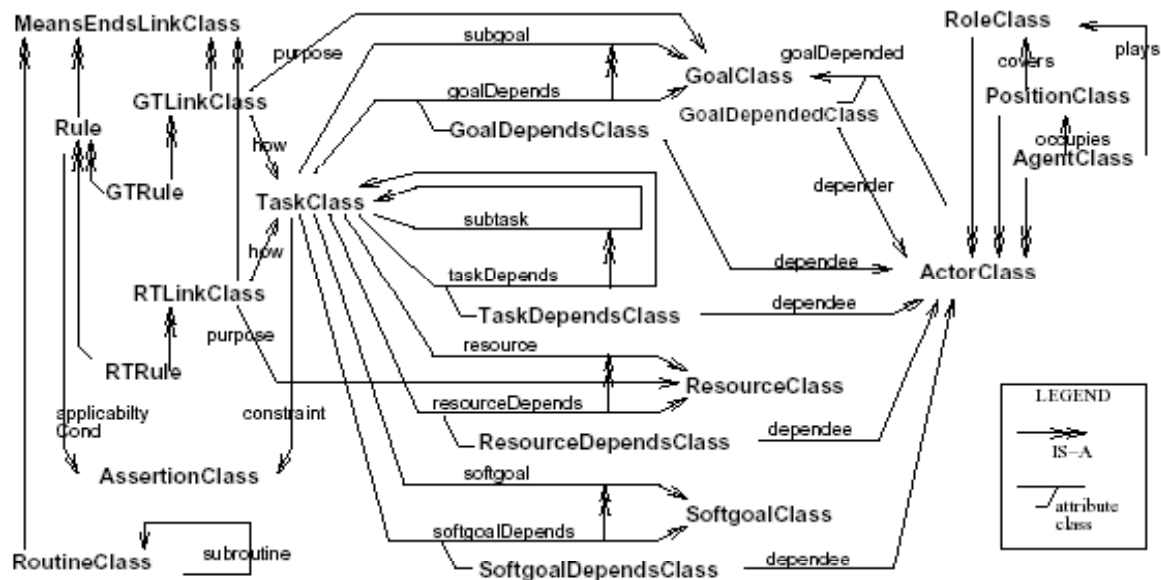


Figure 17 - A partial schema for the i* Requirement Model.

Figure 17 is a partial schema of the i* Requirement Model, and shows the task refinement links and some classes of dependency links. The middle section of the figure deals with task

decomposition links and their corresponding dependency links. A task can be refined into subgoals, subtasks, resources and softgoals. A rule is a means-end link that is not yet bound. It consists of an applicability condition, a means, and an end. A means-ends link is an application of a rule in a context in which the agent believes the applicability condition to hold. For means-ends links and rules, only the Goal-Task (GT) and Resource-Task (RT) cases are shown.

Telos treats entities and attributes uniformly. The attributes can be grouped into classes and have their own attributes. This allows the definition of different kinds of relations, without the need of supplying specific representation primitives for each of them. Telos is described in detail in [Mylopoulos et al, 1990].

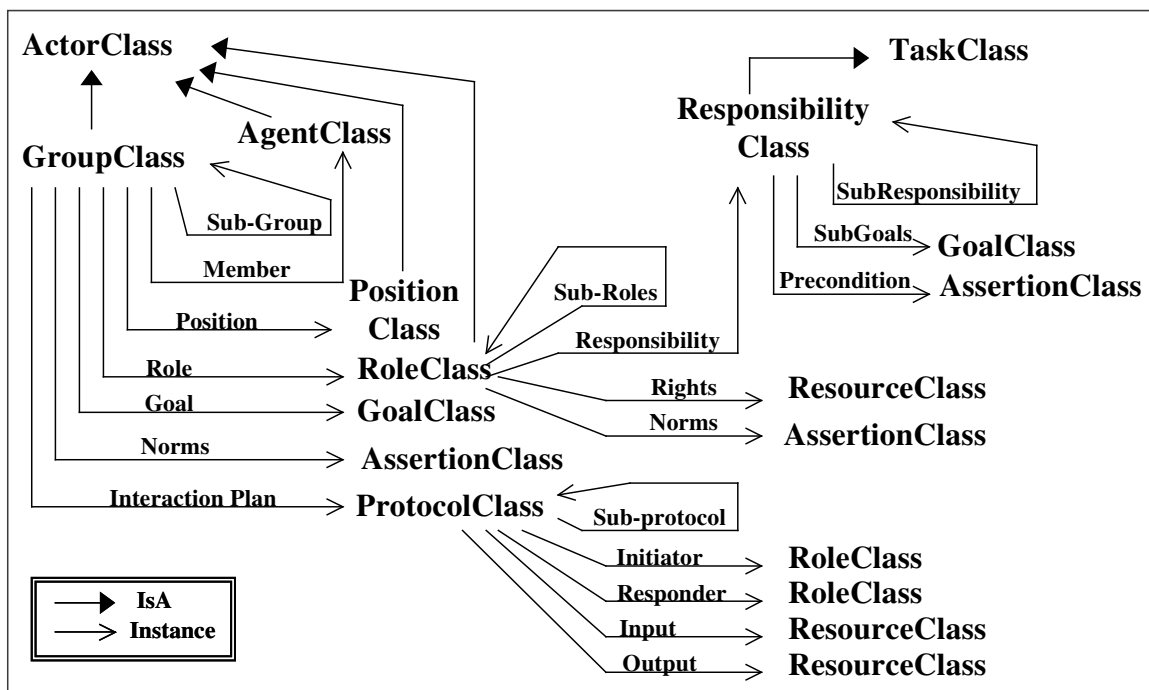


Figure 18 - Organisational Model in Telos

Figure 18 is a partial schema of the SIRA Organisational model. SIRA Group is meant to describe systems as organisational group structures at the conceptual level. Figure 18 shows Group Class as a sub-type of Actor Class of the i^* meta-model. Group Class can be refined into Sub-Groups, Members, Positions or Roles in order to represent the social structure of an agent organisation. The Group Class has a set of attributes:

- 'Member' corresponds to a set of agents in the group;
- 'Position' corresponds to a set of social identities in the group;
- 'Role' corresponds to the division of labour (or responsibilities) between the members;

- ‘Goal’ corresponds to a set of main goals that the group should accomplish;
- ‘Norms’ correspond to constraints (or rules) on group behaviour and interactions;
- ‘Interaction Plan’ corresponds to a sequence of interaction that each role should perform to fulfil the group goals.

In this organisational model, any group is the form $G = \{SG, P, R\}$, where SG is a set of sub-groups $SG = \{sg_1, sg_2, \dots, sg_m\}$ such that $i \neq j \Rightarrow sg_i \neq sg_j$ and $m \leq n$. P is a set of positions $P = \{p_1, p_2, \dots, p_m\}$ such that $i \neq j \Rightarrow p_i \neq p_j$, and R is a set of roles $R = \{r_1, r_2, \dots, r_n\}$ such that $i \neq j \Rightarrow r_i \neq r_j$.

For a set of n agent members $A = \{a_1, a_2, \dots, a_n\}$, the maps $A \rightarrow R$ and $R \rightarrow P$ are not fixed: roles can be performed by different members and positions can be filled by different roles. A position serves as a placeholder for a specific member. The role related to a position should be assigned to the position itself instead to the member occupying it (although it is possible to assign role to an member directly as well). A position can be occupied by one member, but it may be left empty, too. The map $A \rightarrow P$ is fixed.

Hence, the modelling of organisational groups is enabled by two elements. First, the strict organisational hierarchies that allows the modelling of multidimensional relationships among sub-group, positions and roles. Secondly, the relationship between member and group is separated by the introduction of *positions*, which form the static organisational structure of a group. An agent member can occupy several organisational positions and can therefore be a member of several organisational sub-groups.

4.3.1 Group and Sub-groups

The organisational group represents the social structure of an agent society. The collaboration between actor members is achieved through the introduction of *role interaction*.

The Bank Group textual specification is extracted from Figure 15 (on page 52) and Figure 16 (on page 53). Each Group should have a unique name. Moreover, groups are described in terms of the responsibilities, positions, and roles of their members, as shown in Table 2.

Table 2 - Bank Initial Group specification

Group	Bank Group
Responsibilities	Holding, investing and lending money
Positions	Bank Office Manager, Account Manager, Bank Clerk, Financial Manager
Roles	General Manager, General Controller, Account Manager, Account Handler, Deposit holder, Lender, Manager and Accountant.

In addition, groups can specify sub-groups that do not involve the whole organisation. Sub-groups consist of a subset of roles and are used to deal with local goals. Rather than involving the entire group in a goal, the roles that address it are organised into a sub-group. Any sub-

group is the form $\mathbf{SG} = \{\mathbf{R}\}$, where $\mathbf{R}$ is a set of roles $\mathbf{R} = \{r_1, r_2, \dots, r_n\}$ such that $i \neq j \Rightarrow r_i \neq r_j$.

In sum, the basic idea behind the notion of sub-groups is to provide a means to collectively refer to a set of roles, which perform a collaborative set of tasks to achieve a goal.

4.3.2 Goal Refinement

In *SIRA* Framework, the goal analysis of Tropos is used to redefine the system-to-be actor into a set of autonomous tasks. The essence of goal refinement is to decompose a goal into tasks and subtasks so that each sub-task requires the cooperation of fewer roles.

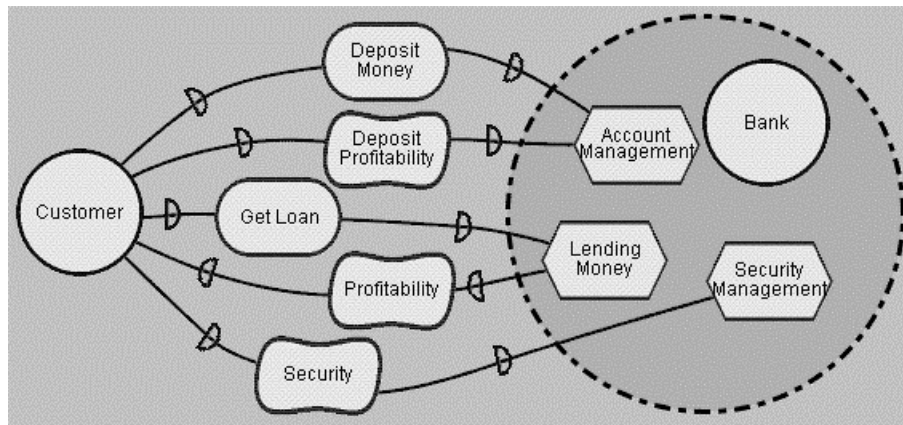


Figure 19 - Cooperation between two actors to fulfil two goals

Figure 19 shows the relationships of dependence and cooperation between two organisational actors that allows the achievement of two goals, in i^* notation. In the Bank example from Figure 16 (on page 53), the Customer actor depends on the Bank actor to achieve his goals *Get Loan* and *Deposit Money*:

1. A loan is the sum of money that has to be paid back with interest. The Bank actor has to perform the task *Lending Money* to fulfil the Customer's goal *Get Loan* and the Bank's softgoal *Profitability*.
2. A deposit is the sum of money belonging to customers that the Bank holds into individual accounts. A bank account is one of the ways to manage customer money. Here, the Bank actor has to perform the task *Account Management* to fulfil the Customer's goal *Deposit Money* and the softgoal *Deposit Profitability*.

Additionally, the Bank actor has to perform the task *Security Management* to accomplish the softgoal *Security*.

Means-end analysis is applied for the discovery of the means (task and interactions) that contribute to achieving the objective. The main goals are refined into task and sub-tasks. Task refinement is related to the identification of sub-tasks and interaction sequence. This results in the refinement of tasks into sub-tasks and in the identification of group responsibilities. In particular, means-end analysis aims at identifying tasks that provide the means for achieving a goal [Bresciani et al, 2004].

Figure 20 shows the refinement with i* means-ends analysis for the task *Lending Money*. It is one of the two major tasks of any Bank. The task *Lending Money* is refined in three sub-tasks, as defined below:

1. The sub-task *Customer Profile Management* to create, update and analyse information about customers;
2. The sub-task *Loan Request Analysis* to check if the customer has a positive credit historic (Credit Analysis) and his ability to repay the loan (Payment Analysis);
3. The sub-task *Loan Account Management* to provide an account for loan transactions (New Loan Account), to keep track on payment scheduler and balance (Loan Balance), and to support the finances of bank transactions (Accounting).

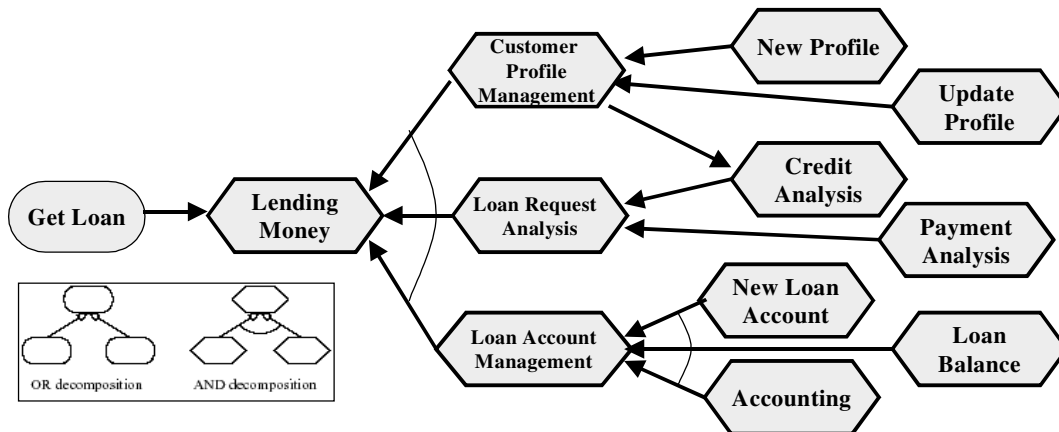


Figure 20 - Means-ends Analysis for the task Lending Money

The refinement for the task *Account Management* is shown in Figure 21. The task is refined in two sub-tasks, as defined below:

1. The sub-task *Customer Profile Management* to get or update information about customer;

2. The sub-task *Account Management* to provide transactions of account information and balance (Account Balance), and transactions to keep track of different kinds of accounts: checking accounts, saving accounts and investment accounts.

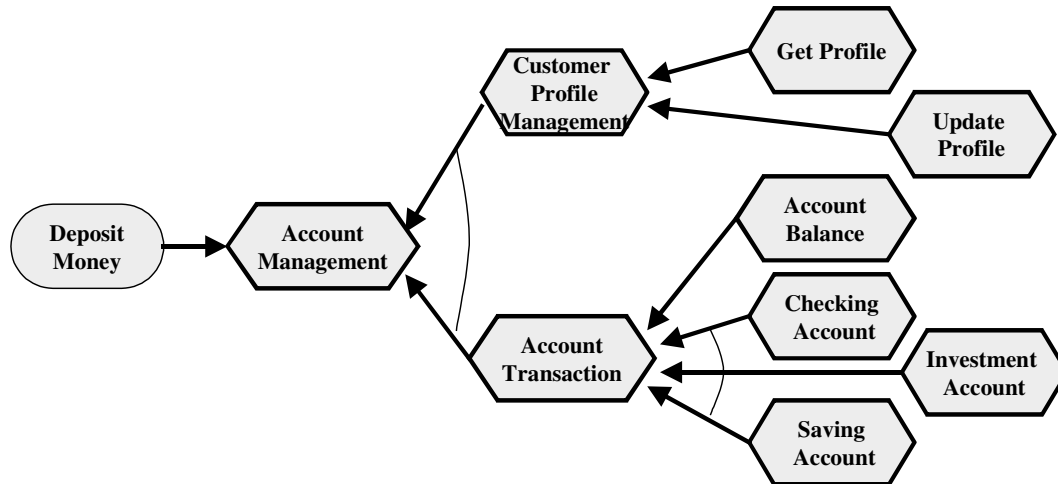


Figure 21 - Means-ends Analysis for the task Account Management

In this initial exploration, the means-end analysis diagram represents the functionality of the multi-agent systems. The next step focuses on the identification of roles and interactions.

4.3.3 Role Identification and Interactions

Most organisational models provide a role concept for the modelling of entities, as seen in Chapter 2. A role clusters types of behaviour into a meaningful responsibility to contribute to group goals.

Moreover, roles may be captured in a specific and bounded domain. For example, if we consider a bank domain, the following role categorisation in Table 3 can be used.

Table 3 - Role categorisation in the Bank domain

Depositor	Role of customers who place money in a bank.
Borrower	Role of customers who borrow money from a bank.
Lender	Role that lends money.
Manager	Role to keeps the control of the bank organisation or service.
Controller	Role in a business organisation who is responsible for managing its finances.
Handler	Role that has responsibility for specific bank service.
Holder	Role that has responsibility for executing the account transactions.
Accountant	Role that keeps or examines the records of money received, paid, and owed by a company or person

In the bank example, the task *Lending Money* is performed to fulfil the Customer goal *Get Loan*. The roles are identified as follows:

1. The *Loan Manager* is identified to perform the task *Lending Money*;
2. The *Lender role* is identified to perform the sub-task *loan request analysis*;
3. The *Loan Handler role* is identified to perform the sub-task *loan account management*;
4. The *Profile Handler role* is identified to perform the sub-task *customer profile management*;
5. The *Accountant role* is identified to perform the sub-task *accounting*.

Roles are identified by a set of tasks that fulfil each group goal. The roles identified can now be documented.

4.3.3.1 Role Specification

Consider the bank work example. The Lender Role specification is detailed in Table 4. Roles have a unique name and are described in terms of objectives to be fulfilled when performing the role. Responsibilities are a list of task and interactions extracted from the set of main goals to be fulfilled by actors, such as “Get Loan” and from interactions between roles Borrower and Lender, as seen in Figure 19 (on page 57). Collaborators are the set of interactions the role has with other roles. Skills are the set of expertise needed to perform the role responsibility. Rights are the set of privileges and competence (in term of access and recourse). Norms are the set of constraints and normative expressions restrict the actor should when performing a role. Norms can be defined in terms of business rules or conflict resolution.

Table 4 – The Lender role specification

Role	Lender
Objective	Lending money to a customer
Responsibilities	Loan Request Analysis; Loan Account Management
Collaborators	Borrower and Accountant
Skills	Knowledge about financial market
Norms	Bank operational limits and norms Customer must have investment or saving account

Every role communicates and interacts with other roles. Role relations and interaction protocols are expressed within a particular organisational level, i.e., roles are local to Groups or Sub-Groups. A group can define a set of interactions to be executed. The role interactions are the set of protocols $G_{ip}=\{Ip_1, Ip_2, \dots Ip_m\}$ $m \leq n$ that define the actions to be taken by the participants in the group-goal.

The protocol is a normative expression of the set of interactions that a role has with other roles. Protocols have unique names and are described in terms of the nature of the interaction. Initiator is the role(s) responsible for starting the interaction. Responder is the role(s) with which the initiator interacts. Inputs represent the information used by the role initiator while enacting the protocol. Outputs are information supplied by the protocol responder during the interactions. Table 5 shows the textual specification of an interaction protocol in the bank example. The protocol *Loan Request* is started when a Customer actor plays the *Borrower Role* to request money. The bank actor interacts by playing the Lender Role. The input is a 'Request Form' with the information used by 'Lender' to execute the task 'Lending Money'. The output is the response about the 'Loan Request' is the credit limit accepted (>0) or not (0).

Table 5 - Interaction Protocols

Protocol	Loan Request
Objective	Requests a loan
Initiator	Borrower
Responder	Lender
Inputs	Request form
Outputs	Credit limit

4.3.3.2 Role Dependencies and Relations

The interaction patterns describing social structure can be viewed as a network of relations. The SIRA Framework uses two kinds of tools from mathematics to represent information about interactions among roles: social network graphs and matrices.

A social network is a graph (sometimes called a sociogram) that consists of a set of nodes (or actors) connected by edges (or relations) [Hanneman, 2005]. It represents a method for guiding the analysis of groups when the focus is on patterns of interaction over time [Wasserman and Faust, 1994] [Zack, 2000].

A graph may represent a single type of relation among the actors (simplex), or more than one kind of relation (multiplex). Each tie or relation may be directed (i.e. originates with a source actor and reaches a target actor), or it may be a tie that represents co-occurrence, co-presence, or a bonded-tie between the pair of actors. Directed ties are represented with arrows, bonded-tie relations are represented with line segments. Directed ties may be reciprocated (A chooses B and B chooses A); such ties can be represented with a double-headed arrow. The strength of ties among actors in a graph may be binary (representing the presence or absence of a tie); signed (it represents negative, positive, or no tie); ordinal (it represents whether the tie is the strongest, next strongest, etc.); or valued (it measures an interval or ratio level).

In SIRA, role interactions are expressed within a particular organisational level, i.e., roles are local to Groups or Sub-Groups. An interaction graph is a set of relations performed and a set of roles played to fulfil a group goal. The nodes represent the actors (or set of roles in a sub-

group) and the arrows represent the relations among them. This representation makes it possible to distinguish between roles that interact and those that do not.

Figure 22 shows the interaction plan for the goal *Get Loan* in the bank example and uses a directed graph with the actor relations. The task *Lending Money* is performed by six roles:

1. Lender is the role played by the bank actor to interacts primarily with Customer (such as Borrower Role) to provide loan request interfaces;
2. Loan Handler is the role played by the bank actor to keep track of all information about loan account, balance and payment;
3. Security Manager is the role played by the bank actor to provide secure management of the bank services;
4. Loan Manager is the role played by the bank actor to manage and control all strategic data about lending money;
5. Profile Manager is the role played by the actor that keep track of all information about customer profiles;
6. Accountant is the role played by the actor that keeps track of all information about bank accounting and other financial data.

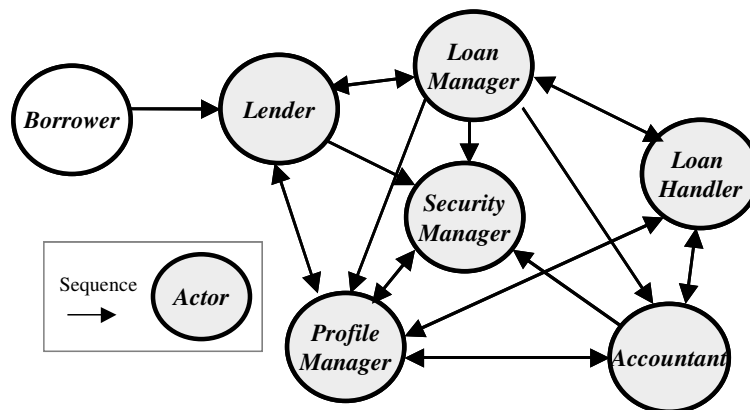


Figure 22 - Role Interaction Graph of the goal Get Loan

Figure 23 shows the interaction plan for the goal *Deposit Money* in the bank example and uses a directed graph with the actor relations. The task Account Management is performed by six roles:

1. Deposit Holder is the role played by the bank actor to interact primarily with Customer (Depositor Role) to provide deposit money transactions;

2. Account Handler is a role played by the bank actor to keep track of all information about accounts transactions and balances;
3. Security Manager is the role played by the bank actor to provide secure management of the bank account services;
4. Account Manager is the role played by the bank actor to manage and control all strategic data of account management tasks;
5. Profile Manager is the role played by the bank actor to keep track of all information about customers profile;
6. Accountant is the role played by the bank actor to keep track of all information about bank accounting and other financial data.

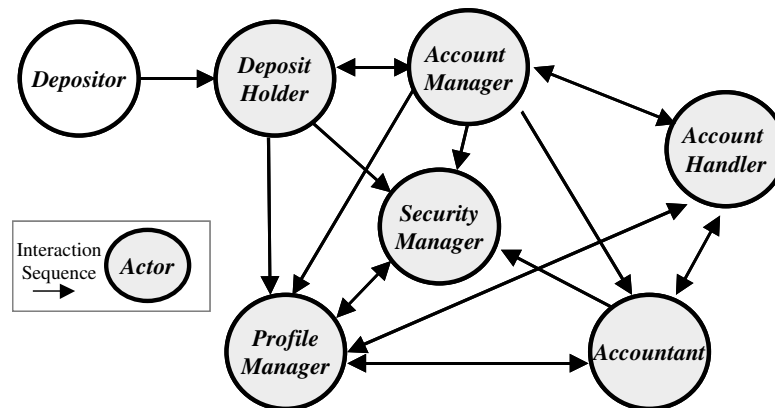


Figure 23 - Role interaction graph of the goal Deposit Money

However, when there are many actors and/or many kinds of relations, the graph representation can become so visually complicated that it is very difficult to see patterns. It is possible to represent information about relations in the form of matrices. Matrix representation allows the application of mathematical and computer tools to summarise and find patterns [Golub and Loan, 1996].

Matrices are collections of elements organised in rows and columns. The Adjacency Matrix is the starting point for almost all network analysis because it represents what actors is next to, or adjacent to other actor in the "social space" mapped by the relations that are measured. An adjacency matrix is a square actor-by-actor ($i=j$) matrix where the presence of pair-wise ties is recorded as an element. Binary choice data are usually represented with zeros and ones, indicating the presence (1) or absence (0) of each logically possible relationship between pairs of actors.

By convention, in a directed graph, the sender of a tie is the row and the target of the tie is the column. Let's look at the bank example. The directed graph of the goal Get Loan choices

among roles into bank organisation, as seen in the 6 by 6 matrix (see Table 6). The rows represent the source of directed ties, and the columns the targets; *Lender* chooses *Loan Handler* here, but *Loan Handler* does not choose *Lender*. This is an example of an "asymmetric" matrix that represents directed ties (ties that go from a source to a receiver). That is, the element i,j does not necessarily equal the element j,i .

Table 6 - Interaction matrix of the goal Get Loan

	Lender	Loan Manager	Profile Manager	Security Manager	Loan Handler	Accountant
1. Lender	--	1	1	1	0	0
2. Loan Manager	1	--	1	1	1	1
3. Profile Manager	1	0	--	1	1	1
4. Security Manager	0	0	1	--	0	0
5. Loan Handler	0	1	1	0	1	--
6. Accountant	0	0	1	1	--	1

The interaction matrix of the goal Deposit Money is shown in Table 7.

Table 7 - Interaction matrix of the goal Deposit Money

	Deposit Holder	Account Manager	Profile Manager	Security Manager	Account Handler	Accountant
1. Deposit Holder	--	1	1	1	0	0
2. Account Manager	1	--	1	1	1	1
3. Profile Manager	1	0	--	1	1	1
4. Security Manager	0	0	1	--	0	0
5. Account Handler	0	1	1	0	1	--
6. Accountant	0	0	1	1	--	1

The social analysis of these interaction graphs (and matrix) for the bank example will be outlined in SIRA Assignment Model in the next chapter.

Beyond role interactions, patterns of behaviour are specified through goals that must be either satisfied or avoided by actors. These norms are sometimes represented as restrictions to be imposed on an actor's tasks.

4.3.4 Norms

Norms are mechanisms to govern the behaviour of the organisation especially in those cases when actor behaviour might affect other members. They can be characterised by their prescriptiveness, sociality and social pressure. That is, a norm tells an actor how to behave (prescriptiveness) in situations where more than one actor is involved (sociality). And, if it is expected that norms conflict with the individual interests of actors, socially acceptable mechanisms are needed to force actors to comply with norms (social pressure) [Lopez and Luck, 2003].

For example, bank institutions will tend to be careful about who they loan money to. Banks are far less likely to offer loans to applicants with bad credit ratings, as by their very nature, banks are more cautious in terms of investment. A bank's lending is ultimately limited by its own capital. The capital adequacy norm requires that the ratio of its capital to risk-weighted assets be at least 8%. The assurance norm requires that the home or business is the security for a loan to buy the home or business. If the borrower fails to repay a loan, the bank assumes the ownership of the home or business.

In brief, norms specify something that ought to be done and, consequently, a set of normative goals must be included in the requirement model. Sometimes, these normative goals must be directly intended, while at other times their role is to inhibit specific states (as in the case of prohibitions).

The organisation of multi-agent systems in context of architectural styles is one aspect that the SIRA Framework addresses. The Architectural model is important because it represents the particular abstraction for understanding the structure of a system.

4.4 ANALYSIS OF ARCHITECTURAL ALTERNATIVES

Software architecture modelling is recognised as an important and explicit activity in software development. However, there is a great variation in use of architectural concepts in the state of the practice. The most cited definitions for software architecture emphasises architecture as a description of a system as a sum of smaller parts, and how those parts relate to and cooperate with each other to perform the system [Perry and Wolf, 1992], [Garlan and Perry, 1995], [Shaw and Garlan, 1996], [Bass et al, 1998].

In account of the architecture as a high-level structure of a software system, the following properties have been highlighted [Bachmann et al, 2000]:

- Sufficiently high-level abstraction to enable the system to be seen as a whole;
- The structure should provide the required functionality for the system and the architectural project should take into account the system's dynamic behaviour;
- The architecture has to conform with system quality requirements (non-functional requirements). These requirements include performance, safety and trust, and they are associated with the functionality, flexibility and extensibility.

Due to their unique suitability for several classes of computational problems, Multi-Agent Systems (MAS) are being advocated as a next generation model for engineering complex and distributed systems. Hence, it is important to characterise MAS as a software architectural style.

4.4.1 Architectural Elements in Multi-Agent Systems

The architecture of a Multi-Agent System (MAS) is the way in which the agents, the processes they perform, and the resources they produce make up an organisation structure. MAS Architecture can also be described as a collection of components (or set of agents), the interaction among these components and the organisational styles according to which the agents are combined to form the whole system. The MAS architecture is defined in terms of:

- ◆ **Architectural Styles:** Currently, there are several classical software architecture styles, such as pipes/filters style and layered structure style [Perry and Wolf, 1992] [Shaw and Garlan, 1996]. However, MAS can benefit from new organisational architectural styles, which are based on concepts and design alternatives coming from research on organisation management [Mintzberg, 1992]. These architectural styles (pyramid, joint venture, structure in 5, takeover, arm's length, vertical integration, co-optation, bidding) support the design of cooperative, dynamic and distributed MAS applications [Do et al, 2003], [Kolp et al, 2003].
- ◆ **Components:** Multi-agent models of organisation represent a way of looking at software architecture styles. Hence, in the SIRA Framework, MAS are systems comprised from components called sub-groups. Components are usually designed to be autonomous, where autonomy refers to a component not depending on the properties or the states of other components for its functionality.
- ◆ **Constraints** - The set of architectural constraints of software architecture includes all properties that are derived from the selection and arrangement of components and interactions. Examples include both the functional properties achieved by the system and non-functional, such as reusability of components, efficiency, and dynamic extensibility, often referred to as quality attributes.

In order to be able to say something about the relation between requirement and organisational architecture, we must be able to distinguish different pattern of architecture. This requires that we have an idea about which are the relevant properties of multi-agent architectures, and which architectural alternatives exist for each of those properties.

4.4.2 Non-Functional Properties of MAS

Non-Functional requirements have been recognised to be important since the early days of software engineering, as seen in [Perry and Wolf, 1992], [Ambriola and Gervasi, 1996] and [Shaw and Garlan 1996]. However, there are few comparative studies of MAS properties and which architectural alternatives exist for each of these properties.

Unlike functional requirements that define what the software is expected to do, non-functional requirements specify global constraints on how the software operates or how the functionality is exhibited. Non-functional requirements are as important as the functional ones. They are not simply desirable quality properties, but critical aspects of dynamic systems without which

the applications cannot work and evolve properly. The need to treat non-functional properties explicitly is a critical issue when software architecture is built.

Because quality requirements are often subjective by nature, they are often achieved not in an absolute sense, but to a sufficient or satisfactory extent (the notion of *satisficing*). A more precise and systematic analysis of these quality attributes can be obtained with goal-oriented frameworks such as KAOS [Dardenne and Lamsweerde, 1993], [Lamsweerde, 2001] or the NFR (non-functional requirements) Framework [Chung et al, 1995 & 2000], [Chung 1998].

Some quality properties are identified for architectures involving coordinated autonomous components from the perspective of the Multi-agent Systems [Weiss, 2000], [Wooldridge and Ciancarini, 2001], [Yu, 2001], [Sturm and Shehory, 2003]:

- **Predictability.** Agents have a high degree of autonomy in the way that they act and communicate in their domains. Hence, it can be difficult to predict individual characteristics as part of determining the behaviour of a distributed and open system at large.
- **Security.** Autonomous components are often able to identify their own data sources and they may undertake additional actions based on these sources. Protocols and strategies for verifying data authenticity by individual components are an important concern in the evaluation of overall system quality.
- **Adaptability.** Components may be required to adapt to modifications in their environment. They may include changes to the component's communication protocol or the dynamic introduction of a new kind of component previously unknown.
- **Coordinability.** Autonomous components are not particularly useful unless they are able to coordinate with other components. Coordinability takes two forms:
 - *Cooperativity.* They must be able to coordinate with other entities to achieve a common purpose.
 - *Competitivity.* The success of one component implies the failure of others.
- **Availability.** Components that offer services to other components must implicitly or explicitly guard against the interruption of offered services. Availability must actually be considered a sub-attribute of security. Nevertheless, we deal with it as a top-level software quality attribute due to its increasing importance in multi-agent system design.
- **Integrity.** The failure of one component does not necessarily imply the failure of the whole system. To prevent system failure, different components can have similar or replicated capabilities and refer to more than one component for a specific behaviour. The system needs to check the completeness and the accuracy of data, information and knowledge transactions and flows.

- **Modularity** increases the efficiency of task execution, reduces communication overheads and usually enables high flexibility. On the other hand, it implies constraints on inter-module communication.
- **Aggregability.** Some components are parts of other components. They surrender to the control of the composite entity. This control results in efficient task execution and low communication overheads.

Throughout the development process, the selection of architectural alternatives, the analysis of design tradeoffs, and the rationalisation of design decisions are all carried out in relation to the stated quality requirements and their refinements.

4.4.3 Analysis of Architectural Alternatives for the Bank Example

In the bank example, one of the most important non-functional requirements when building information systems to be used in the bank domain is security, i.e., protecting bank transactions against unauthorised access. Access to account transactions must be restricted to the holders of valid banking cards and personal identification numbers. The same reasoning is applied to *Integrity* and *Availability*. *Availability* measures the likelihood that the bank system will be available for use.

The SIRA Architectural model uses the organisational architectures of Tropos (Section 2.6.2 on page 32). The softgoal analysis should select or reject each one of the architectural alternatives based on the correlation catalogue (Table 1 on page 46). In our case study, the selected architectural style is the Structure-in-5, because it contributes more positively for satisficing the softgoals security, availability and integrity in the bank domain, as seen in Table 8.

Table 8 - The Correlation Catalogue for the Bank Example

CORRELATION	SEC	AVAIL.	INTEG.
Flat	--	+	+
Struct-5 (v)	+	+	++
Pyramid	++	+	--
Joint-Vent	+	++	
Bid	--	-	--
Takeover	++	+	
Arm's-Length	--	--	++
Hierarchical Ctr		+	
Vertical Integration	+	+	--
Cooptation	-	-	+

4.5 SUMMARY

This chapter has presented our approach modelling multi-agent systems, the SIRA Framework. This framework emphasises the organisational environment and helps to reduce

the gap among requirement specification and architectural models. Multi-Agent Systems (MAS) represent an exciting new means of analysing and designing complex software systems. The SIRA Framework advocates that MAS correspond to organisational structure, in which actors are members of a group in order to perform specific tasks.

In the current chapter we have given a detailed overview of the Organisational Model and have made use of a bank example to illustrate the capabilities and model choices of SIRA.

In the SIRA Organisational Model, an actor represents a group responsible for performing a task. An organisation comprises groups, members, roles and interactions. A member assigned to a role does not work in isolation but interacts and cooperates with other roles. As the use of MAS increase, it is necessary to analyse and study MAS requirements from the viewpoint of Software Architecture. In the SIRA, organisational styles integrate non-functional requirements with the MAS architecture, since non-functional requirements are part-compose these styles.

Additionally, performing a role demands a specific organisational structure and levels of autonomy. The Assignment model addresses the relation between the functional requirements of a sub-group and autonomy of the components. The model will be presented in the next chapter.

Chapter 5

The Assignment Model

This chapter overviews the SIRA Assignment Model. The purpose is to map organisational elements into architectural components. The result is the first architectural configuration of the multi-agent systems.

5.1 INTRODUCTION

Traditional multi-agent models often assume an individualistic perspective on the environment. Agents are taken as autonomous individuals pursuing their own goals based on their own beliefs and capabilities, while organisational characteristics are external to each individual and independent from the latter goals and behaviour. In this perspective, global behaviour emerges from individual interactions. This implies that, to a certain degree, multi-agent organisations can be viewed as consisting of individuals interconnected as members of social networks. Consequently, research on multi-agent systems can benefit from methods that are explicitly directed toward describing and measuring organisational forms and structures.

The SIRA Assignment Model proposes social network analysis as a method for framing and describing the effects of organisational characteristics and for relating multi-agent systems and organisational architectures. The method is useful because it clusters roles into sub-groups by measuring the strength of the relations among members in a group. In addition, the method focus is on group interactions.

This chapter is organised as follows. Section 5.2 outlines the techniques of social network analysis. Section 5.3 describes the basic structure of the Assignment Model. Section 5.4 explains the mapping of sub-groups into components. Finally, section 5.5 summarises the chapter.

5.2 SOCIAL NETWORK ANALYSIS

The organisational groups are analysed using concepts of social network analysis [Hanemman, 2005], [Wasserman and Faust, 1994], [Zack, 2000]. One purpose of network analysis especially of social groups is to identify important actors, relations, subgroups, roles and group characteristics, and so on, to answer substantive questions about organisational structure.

In this thesis, there are three main levels of interest to apply network analysis techniques: the member, groups and structure. On the member level, one is interested in properties of single actors and interactions. On the group level, one is interested in classifying the members of groups and properties of sub-groups. Examples are actor's equivalence classes and cluster identification. Finally, on the structure level, one is interested in properties of the overall organisational structure such as an architectural pattern.

The SIRA Assignment Model relies on two views of analysis to derive the assignment model. These are *The Centrality of Actors* and *The Structural Equivalence between Actors*.

5.2.1 The Centrality of Actors

Centralisation refers to the extent to which an organisation revolves around an actor [Freeman, 1979]. Organisational information systems may affect centrality. An organisation is more centralised to the extent the distribution of actor centrality is skewed towards a small number of individual members. It is a measure of the contribution of structural position to the importance, influence, and prominence of an actor in an organisational architecture. Figure 24 shows a star network, the central point (A) has complete centrality, and all other points have minimum centrality: the star is a maximally centralised graph.

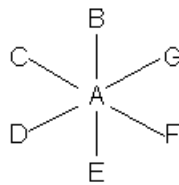


Figure 24 - Star Network

Centrality deals with two aspects to describe the location of individuals in terms of how close they are to the "center" of the action in a organisational structure [Zack, 2000], [Hanneman, 2004]:

1. **Degree centrality** [Freeman, 1979]- Actors who have more relations to other actors may be advantaged positions. Because they have many relations, they may have alternative ways to satisfy needs, and hence are more autonomous. They may also have access to, and be able to call on, more of the resources of the organisation as a whole. So, an effective measure of an actor's centrality and power potential is their degree. In the star example (Figure 24), actor A has more opportunities and alternatives than others. If actor D elects to not provide A with a resource, A has a number of other places to get it. However, if D elects to not exchange with A, then D will not be able to exchange at all. The more relations an actor has then, the more power they (may) have. In the star network, Actor A has degree six and all other actors have degree one.
2. **Closeness** - The second reason why actor A is more powerful than the other actors in the star network is that actor A is *closer* to more actors than any other actor. Actors who are able to reach other actors at shorter path lengths, or who are more reachable through shorter path lengths have favoured positions. This structural advantage can be translated into power. In the star network, actor A is at a geodesic distance of one from all other actors. By contrast, other actors are at a geodesic distance of two from all other actors (but A). Approaches that emphasise the distribution of closeness and distance as a source of power are based on this logic of structural advantage.

5.2.1.1 Degree Centrality Analysis

Degree Centrality measures the extent to which an actor is connected to other actors. In other words, greater the number of relations, higher is the degree of that actor. In a directed graph both in-degree centrality and out-degree centrality may be measured:

1. In-degree: If an actor receives many relations, the actor is often said to be *prominent*, or to have high *prestige*. In-degree centrality is the degree centrality of an actor calculated for all the fan-in (or receiving) connections. More in-degree centrality means more prominence for an actor in a group.
2. Out-degree: Actors who display high out-degree centrality are often said to be *influential* actors. Out-degree centrality is the degree centrality of an actor calculated for all the fan-out (or sending) connections. The higher the out-degree centrality, the greater the actor's influence in a group.

Degree centrality of an actor (**n**) is simply each actor's degree value (**d**) in a directed graph. Mathematically, degree centrality (**C_D**) is calculated using the following formula:

$$C_D(n_i) = d_i(n_i)$$

To standardise or normalise the degree centrality index so that networks of different sizes (**g**) may be compared, divide it by the maximum possible degrees (**g-1** actors), and express the result as either a proportion or percentage:

$$C'_D(n_i) = d_i(n_i)/(g-1)$$

Degree centrality measures might be criticised because they only take into account the immediate relations that an actor has, rather than indirect relations to all others.

In the individual analysis, an actor with high degree centrality maintains contacts with numerous other organisational actors. Actors have higher centrality to the extent they have access to and/or influence over others. A central actor occupies a structural position (or group location) that acts as a source or conduit for information exchange and other resource transactions. In contrast, peripheral actors maintain few or no relations and thus are located at the margins of the group.

In the organisational group analysis, degree centrality quantifies the dispersion or variation among individual centrality. The group centrality index contrasts the gap between the largest actor centrality and the others. It ranges from 0 to 1, reaching the maximum when all others choose only one central actor (star) and the minimum when all actors have identical centrality (circle).

5.2.1.2 Closeness Centrality Analysis

Closeness centrality approaches emphasise the distance of an actor for all others in the network by focusing on the geodesic distance from each actor to all others [Wasserman and Faust, 1994], [Hanemman, 2004]. The shortest path, technically, has the advantage of being

near to the source. In other terms, it is said to be in the excellent position to monitor all information flows within the network. The sum of these geodesic distances for each actor is the "farness" of the actor from all others. It can be converted into a measure of closeness centrality by taking the reciprocal (which is one divided by the farness) and normalising it relative to the most central actor (g-1). When an actor has a low closeness score, it tends to receive anything flowing through the network very quickly because nodes with low scores are close to all other actors.

The closeness centrality of an actor n is the inverse of the sum of geodesic distances from actor i to the g-1 other actors (i.e., the reciprocal of its "farness"). Mathematically, the closeness centrality (C_c) is calculated using the following formula [Freeman, 1979]:

$$C_c(n_i) = \left[\sum_{j=1}^g d(n_i, n_j) \right]^{-1}$$

Centrality refers to the importance of a position within a network. The specific type of importance varies from measure to measure. A position can be important because it can easily communicate with other members of the network (*closeness* centrality), or because it is itself connected to other central positions. In the context of organisational flow, the essence of closeness is time-until-arrival of something flowing through the organisation. The Freeman formula provides expected values of arrival times for resource deliveries and other flow processes in which things move along shortest paths or take all paths simultaneously.

5.2.1.3 Centrality of the Bank Example

Using the bank example from Figure 22 (on page 62) let's examine the degree centrality of a set of actors as the measure of who is central or influential in this social organisation. When evaluating the interactions to achieve the goal *Get Loan*, The Profile Manager and Loan Manager roles have the greatest out-degrees, and might be regarded as the most influential. The Profile Manager and Security Manager roles are merged when in-degree is examined. The results are shown in Table 9.

Table 9 - Degree Centrality Measures of the goal Get Loan (g=6)

	Out degree	In degree	Nrm-Out	Nrm-In
1. Lender	3	2	60	40
2. Loan Manager	5	2	100	40
3. Profile Manager	4	5	88	100
4. Security Manager	1	4	20	88
5. Loan Handler	3	3	60	60
6. Accountant	3	3	60	60

When evaluating the interactions to achieve the goal *Deposit Money*, Profile Manager has the greatest in-degree, and Deposit Holder, Account Handler and Accountant are merged when we examine out-degrees. The results are showed in Table 10.

Table 10 - Degree Centrality Measures of the goal Deposit Money (g=6)

	Out degree	In degree	Nrm-Out	Nrm-In
Deposit Holder	3	2	60	40
Account Manager	5	2	100	40
Profile Manager	4	5	88	100
Security Manager	1	4	20	88
Account Handler	3	3	60	60
Accountant	3	3	60	60

In the bank example, the group centrality has the degree of the number of actors, less one (g=6). Table 9 shows the partial amount of concentration or centralisation in this whole social group. In sum, the prominence or influence of individual actors varies partially, and this means that positional advantages are distributed between two actors in this organisational group.

In the bank example, the closeness centrality of the goals Get Loan and Deposit Money are shown in Table 11 and Table 12. Table 11 shows that Loan Manager, Profile Manager and Security Manager are the closest actors using closeness centrality analysis. This is because the sum of geodesic distances to other actors (a total of 7, across 6 other actors) is the lowest (in fact, the minimum possible sum of geodesic distances is 5). The Lender and Loan Handler have the greatest farness.

Table 11 - Closeness Centrality of the goal Get Loan (g=6)

	Farness	Closeness
1. Lender	7	71,42
2. Loan Manager	5	100
3. Profile Manager	5	100
4. Security Manager	6	83,34
5. Loan Handler	7	71,42
6. Accountant	6	83,34

The results in Table 12 shows that the Profile Manager is the most central actor for the goal Deposit Money using closeness centrality analysis. This is because the sum of geodesic distances to other actors (a total of 5, across 5 other actors) is the lowest (Here, the minimum possible sum of geodesic distances is 6). The Deposit Holder and Account Handler have the greatest farness.

Table 12 - Closeness Centrality of the goal Deposit Money (g=6)

	Farness	Closeness
Deposit Holder	7	71,42
Account Manager	5	100
Profile Manager	5	100
Security Manager	6	83,34
Account Handler	7	71,42
Accountant	6	83,34

Because the results of the interaction graph and social analysis of the two goals *Get Loan* and *Deposit Money* in the bank example are similar, the next analyses will use only one goal.

5.2.1.4 Centrality of the architectural style

The Structure-in-5 style was introduced in sub-section 2.6.2.2 (Figure 7 on page 33). The style specifies that an organisation is an aggregate of five sub-structures. The *Operational Core* component carries out the basic tasks and procedures directly linked to the production of products and services. The *Strategic Apex* component makes executive decisions and ensures that the organisation fulfils its mission in an effective way and defines the overall strategy of the organisation in its environment. The *Middle Line* component establishes a hierarchy of authority between Strategic Apex and Operational Core. It consists of managers responsible for supervising and coordinating the activities of the Operational Core. The *Technostructure* and *Support* components are separated from the main line of authority and influence the operating core only indirectly. Technostructure serves the organisation by making the work of others more effective, typically by standardising work processes, outputs, and skills. Support provides specialised services, at various levels of the hierarchy, outside the basic workflow.

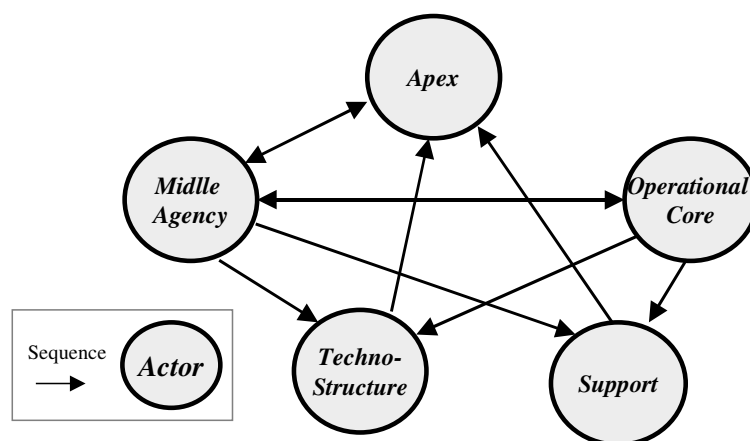


Figure 25 - Interaction Graph of the Structure-in-5 Architecture

The graph interaction for Structure-in-5 Style is shown in Figure 25. To perform its social analysis, it will be necessary to include two relations (or interactions) concerned to the

information exchanges between the actors Apex, Middle Agency, and Operational Core. The Apex actor needs to receive feedback to define the organisational strategy. Middle Agency needs to receive operational feedback to supervise the organisational services and performance.

The interaction matrix of the Structure-in-5 architecture is showed in Table 13.

Table 13 - Interaction matrix of the Structure-in-five architecture

	Apex	Middle Agency	Operational Core	Techno-structure	Support
1. Apex	-	1	0	0	0
2. Middle Agency	1	--	1	1	1
3. Operational Core	0	1	--	1	1
4. Techno-structure	1	0	0	--	0
5. Support	1	0	0	0	--

In the degree centrality measures, Middle Agency and Operational Core have the highest out-degree (sending) and the Apex has the highest in-degree (receiving) among the components of the Structure-in-5 Style (see Table 14).

Table 14 - Degree Centrality of the Structure-in-five Style (g=5)

	Out degree	In degree	Nrm-Out	Nrm-In
1. Apex	1	3	25	75
2. Middle Agency	4	2	100	50
3. Operational Core	3	1	75	25
4. Techno-structure	1	2	25	50
5. Support	1	2	25	50

In the closeness centrality analysis, Middle Agency is the most central actor. The other actors have same centrality measure (see Table 15).

Table 15 - Closeness Measures of the Structure-in-five Style

	Farness	Closeness
1. Apex	5	80
2. Middle Agency	4	100
3. Operational Core	5	80
4. Techno-structure	5	80
5. Support	5	80

5.2.2 The Structural Equivalence between Actors

The structural equivalence of two actors is the degree to which the two actors have the same profile of relations to all other actors in the group [Burt 1980], [Burt 1991], [Zack, 1999 & 2000], [Moody 2003]. Because exact structural equivalence is likely to be rare, we are interested in examining the degree of structural equivalence, rather than the simple presence or absence of exact equivalence.

The degree of structural equivalence can be measured by using a variety of indices. One of the most common ways of calculating such indices is the Pearson Correlation [Stockburger, 1998], [Hanneman, 2005]. Correlation is a technique for investigating the relationship between two variables. Linear relationships are associations between two variables that can be expressed as straight-line graphs. Pearson's correlation coefficient (r) is a **measure of the strength of this association**.

Pearson's correlation (r) ranges from -1.00 to +1.00. If $r = -1.00$, then there is a perfect negative correlation, meaning that the two actors have the opposite pattern of relations to each other actor (or structural dissimilarity). If $r = 0$, then the correlation is zero and there is no relationship between the variables. And, if $r = +1.00$, then there is a perfect positive correlation, meaning that the two actors always have exactly the same pattern of relations to other actors, in this case the two actors have perfect structural equivalence.

Table 16 shows the correlation coefficient and how it will to be interpreted in this thesis.

Table 16 - Interpretation of the Correlation Coefficients

-1,0 to -0,7	(--)	Strong negative correlation
-0,7 to -0,3	(-)	Partial negative correlation
-0,3 to +0,3	()	Little or no correlation
+0,3 to +0,7	(+)	Partial positive correlation
+0,7 to +1	(++)	Strong positive correlation

The formula for Pearson's correlation takes on many forms. A commonly used formula is [Pearson, 2005]:

$$r = \frac{n \sum XY - \sum X \sum Y}{\sqrt{[n \sum X^2 - (\sum X)^2][n \sum Y^2 - (\sum Y)^2]}}$$

The formula looks a bit complicated, but taken step by step as shown in the numerical example, it is really quite simple. Suppose we have two variables X and Y. If we have a series

of n measurements of X and Y written as x_i and y_i where $i = 1, 2, \dots, n$, then the Pearson correlation coefficient can be used to estimate the degree of structural equivalence of X and Y . For example, in the Bank group we take the actors Loan Manager and Profiler Manager as shown in Table 6 on page 64. The variables X and Y represent these two actors respectively. Given that $X = \{1, 0, 1, 1, 1, 1\}$, $Y = \{1, 0, 0, 1, 1, 1\}$ and $n=5$, then the Pearson correlation is calculated as follow:

X	Y	X ²	Y ²	XY
1	1	1	1	1
0	0	0	0	0
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
Σ	5	4	5	4

$$\begin{aligned}
 r &= \frac{6 \times 4 - 5 \times 4}{\sqrt{(6 \times 5 - 5^2)(6 \times 4 - 4^2)}} \\
 r &= \frac{4}{\sqrt{(5)(8)}} \\
 r &= \frac{4}{6,3245} \\
 r &= \mathbf{0,6324}
 \end{aligned}$$

In this example, the correlation coefficient is significantly different from zero, and, hence that there is evidence of a partial association between the two actors Loan Manager and Profiler Manager. The result indicates that the strength of association between the variables is partial positive ($r = 0.6324$). In conclusion, Loan Manager and Profiler Manager have a partial correlation in term of their pattern of relations with others actors in the group.

In the next sub-section the centrality and structural similarity measures will be applied to cluster roles into sub-groups using the bank example.

5.3 THE ASSIGNMENT MODEL

The Assignment Model assigns sub-groups into a particular architectural style. In other words, it aims to distribute the responsibility on a set of architectural components into particular categories or pattern of organisational architectural styles. The Assignment Model addresses two issues:

- 1) *How to Cluster Roles into Sub-groups?* Our analysis is based on an assumption of the importance of relationships among a set of roles to fulfil a goal. Our model clusters like and unlike roles into their respective sub-groups.
- 2) *How to Map Sub-groups into Architectural Components?* Our unit of analysis is the organisational group, which consists of a collection of sub-groups and the interaction among them. Our model examines the correlation between sub-group and components.

5.3.1 Clustering Roles into Sub-Groups

Clustering is a common technique to analyse large data sets. Its main purpose is the identification of natural groups within the data. In the SIRA framework, the cluster analysis is one way of to look for sub-groups. This analysis is thinking about how the roles in an organisation may be categorised on the basis of their patterns of relations with one another i.e., groupings of roles that are closer to one another than they are to the other groupings.

In the cluster analysis, one relevant question is “What level of autonomy fit best to the role responsibility to be performed”. The autonomy of an actor is determined by its organisation. Multi-Agent Systems are redefined as an organisation that performs all the tasks (operational and controlling) needed to fulfil its role responsibility. Thus, a sub-group is able to perform a responsibility (a role or a set of roles) autonomously if all the tasks regarding this responsibility can be performed by this sub-group. This autonomy is characterised by two properties [Zoethout et al, 2002]:

1. The first property states that these interactions are recursively defined by the system, i.e. the system maintains itself.
2. The second property states that the system distinguishes itself as a consequence of these interactions.

An actor when performing a role is an autonomous system that can be described as a set of interactions. These interactions are analysed using two techniques:

- In the centrality analysis, the prominence of actors can influence their responsibilities in the organisational hierarchy. The interactions among roles are useful for measuring the strength of the relationships and flows between roles and sub-groups. When a role has a low closeness score, it tends to receive anything flowing through the network very quickly because role with low scores are close to all other actors.
- In the structural similarity analysis, clusters roles that have similar patterns of relations with others, even if these roles do not have direct relations with each other. All actors playing a similar role are said to occupy similar structural or status positions. Hence, these roles are grouped into a sub-group.

Besides the actor social relations of the organisational group, an architectural style may also constrain the nature and distribution of actor relations. So, before comparing a group with an architectural style, it is necessary to cluster roles into sub-groups. For example, the Structure-in-5 organisational style is defined with five fixed positions, each one occupied by a component. So, the Roles of the Bank example should be clustered into five sub-groups.

5.3.1.1 Cluster Analysis for the Bank Example

The first step in cluster analysis is to measure the overall integration of a group. The centrality analysis indicates the extent to which a group is organised around its most central actor. This

is measured by computing the difference between the centrality scores of the most central point and those of all other points. The centrality analysis (see Table 9 and Table 11 on pages 74 and 75) provide the conclusion that the actors Loan Manager and Profile Manager are the most central roles, and they have partial similarity in centrality results:

1. Similar relations with other actors (Lender, Security Manager, Loan Handler and Accountant).
2. Related degree centrality. Loan Manager has an out-degree value=100 and Profile Manager has an out-degree value=88.
3. Identical closeness centrality. Loan Manager and Profile Manager have a closeness value=100

The second step in the cluster analysis is to produce a “similarity” or “distance” matrix for all pairs of actors. This matrix summarises the overall similarity (or dissimilarity) of each pair of actors in terms of their relations. What this does is to group together the roles that are most similar first. Taking the score in each cluster that is closest to some other cluster, similarity is recalculated, and the next most similar pair is joined.

The Pearson correlation formula is used to calculate each similarity coefficient [Pearson 2005]. Here, the interaction matrix of the goal Get Loan is used as an example (see Table 6 on page 64). The results cluster analysis is shown in Table 17.

Table 17 – The similarity coefficients of the bank example

	1	2	3	4	5	6
1. Lender	1					
2. Loan Manager	-0,4472	1				
3. Profile Manager	-0,7071	0,6325	1			
4. Security Manager	0,4472	0,2	-0,6325	1		
5. Loan Handler	0,3333	-0,4472	-0,7071	0,4472	1	
6. Accountant	0,3333	0,4472	0	0,4472	-0,3333	1

In the cluster analysis, two actors are structurally equivalent to the extent that the profile of scores in their rows and columns is similar. From the results on Table 17, the pair most similar is Loan Manager and Profile Manager. The calculated similarity is partial positive (0,6325). Additionally, the actors must have the same pattern of relations with others actors.

Based on these results, the actors Loan Manager and Profile Manager are clustered in a subgroup named Service Manager. The current interaction graph for the bank example has five subgroups defined as follows (see Figure 26):

1. The Service Holder sub-group performs roles (such as Lender and Deposit Holder) to provide interface transactions between customers and bank services. In the case of the

task Lending Money, this sub-group has to perform roles to provide basic services for allowing customers to borrow money.

2. The General Manager sub-group plays role (such as Security Manager) to provide strategic information about bank security strategic. In the case of the task Lending Money, the related sub-group has to perform roles to provide security management.
3. The Service Manager sub-group performs roles (such as Loan Manager and Profile Manager) to supervise and coordinate bank services. In the case of the task Lending Money, the related sub-group has to perform roles for providing management of the loan account and customer profiles.
4. The Product Handler sub-group performs roles (such as Loan Handler and Account Handler) to provide transactions of bank products and services. In the case of the task Lending Money, this sub-group has to perform roles to provide loan transactions.
5. The Financial Manager sub-group performs roles (such as Accountant) to handle financial and accountancy services. In the case of the task Lending Money, the related sub-group has to perform roles to provide accountancy services.

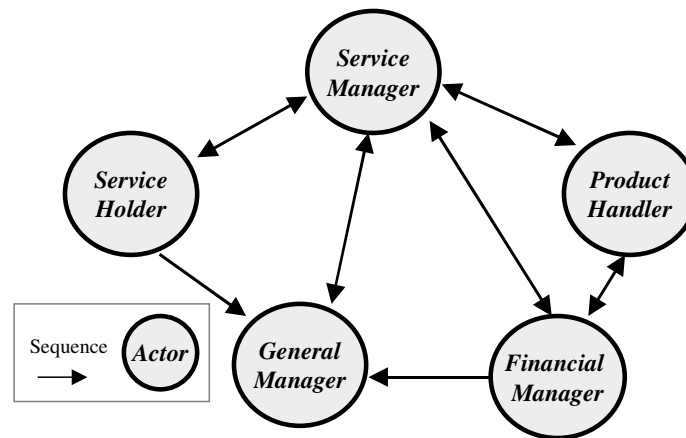


Figure 26 - Interaction Graph after the cluster analysis

The revised interaction matrix is shown in Table 18.

Table 18 - Interaction matrix after the cluster analysis

	1	2	3	4	5
1. Service Holder	--	1	0	1	0
2. Service Manager	1	--	1	1	1
3. Product Handler	0	1	--	0	1
4. General Manager	0	1	0	--	0
5. Financial Manager	0	1	1	1	--

The adjusted centrality analysis is shown in Table 19.

Table 19 - Centrality Measures (Degree and Closeness) (g=5)

	Out degree	In degree	Nrm-Out	Nrm-In	Farness	Closeness
1. Service Holder	2	1	50	25	5	80
2. Service Manager	4	4	100	100	4	100
3. Product Handler	2	2	50	50	6	66,67
4. General Manager	1	3	25	75	5	80
5. Financial Manager	3	2	75	50	4	100

This cluster analysis of the Bank example will be used to relate the bank actors and the Structure-in-5 components in the correlation analysis.

5.3.2 Correlation Analysis between Groups and Architecture

In the centrality correlation, the first measure of architectural structure is centralisation. Centralisation refers to overall integration or cohesion of a group interaction graph. Centralisation indicates the extent to which an architecture is organised around its most central component. We use the measure degree centralisation. In the centrality correlation analysis, the earliest correlation is between the actors from group and architecture based on centrality measures from the Cluster Analysis (sub-section 6.4.1 on page 98).

The centrality correlation analysis uses a correlation table (as see in Table 20) to compare the centrality results of the SIRA Groups and Architectural Style. The degree of an actor is shown by the number of arrows coming in (in-degree) or going out (out-degree) of the point in a graph [Freeman, 1979]. Conceptually, the degree of a point in the graph is the size of its neighbourhood. This is measured by the aggregate difference between the centrality scores of the most central point and those of all other points. In the centrality correlation analysis, the relationship between the centrality measures of all actors can reveal much about the overall organisational structure:

- The centrality correlation notation indicates the relations between sub-groups and components in terms of strong correlation (++), partial correlation (+), and no correlation ().
- One or a few very central actors dominate a very centralised organisational structure. If these actors are removed or damaged, the organisation quickly fragments into unconnected sub-groups. A highly central actor can become a single point of failure.
- A group centralised around a well-connected actor can fail abruptly if that actor is disabled or removed. Well-connected actors are members with high degree centrality.
- A less centralised organisational structure has no single points of failure. It is resilient in the face of many intentional random.

- Degree scores indicate the centrality of the organisational structure. It can range from 0 to 1. 0 being the score for a completely decentralised group. 0,5 being the score for a partial centralised group. And, 1 being the score for a completely centralised group.

Table 20 - Centrality Correlation Analysis

	Nrm-Out	Nrm-In	Closeness	Degree Score	Central Actor
Architecture				0 to 1	
Group					
		Components			
Sub-Groups		Centrality correlation (++) , (+) , ()			

After the centrality correlation analysis, the pattern of relations of the selected organisational style is evaluated in term of its similarity correlation with SIRA sub-groups. The similarity correlation is based on pattern of sending and receiving relations in each group. This analysis looks across the columns and rows to calculate the structural similarity coefficient for each pair of actors using the Pearson Correlation formula (as seen in sub-section 5.2.2 on page 78).

In the next sub-section, the correlation analysis will be detailed using the bank example.

5.3.2.1 The Correlation Analysis between Bank Sub-Groups and Structure-in-Five Components

The first step correlation analysis between sub-group and architectural components is based on centrality measures. Hence, in the measures of the bank example (as seen in Table 19 on page 83), and structure-in-five architecture (as seen in Table 14 on page 77), we have the following results:

1. The analysis of the most central actors:
 - The Service Manager is the most central actor in the Bank group. It has in-degree and out-degree value = 100 and closeness value=100.
 - The Middle Agency is the most central actor in the Structure-in-5 style. It has the highest out-degree value =100 and a closeness value = 100;
2. The analysis of the centrality of the organisational group structure. How the group are connected around the most central actor and others members.
3. The correlation between the centrality measures of the bank group and architectural style:
 - The most prominent (or high in-degree results) sub-groups in the Bank Group are Service Manager, General Manager and Financial Manager;

4. The General Manager has an out-degree value=25, in-degree value=75, and a closeness value=80. It can be related to the Apex, which has similar measures. These two actors have strong centrality correlation (++);
5. The Financial Manager has an out-degree value=75 and closeness value=100. It can be related to the Operational Core, which has an out-degree value=75 and a closeness value=80. These two actors have partial correlation (+);
6. The others actors have no centrality correlation.

Table 21 - Centrality Correlation in the Bank example

	Nrm-Out	Nrm-In	Closeness	Degree Score	Central Actor
Structure-in-5	100	50	100	1	Middle Agency
Bank Group	100	100	100	1	Service Manager
Structure-in-5 Components					
Bank Sub-Groups	Apex	Middle Agency	Operational Core	Techno-structure	Support
Service Holder					
Service Manager		++			
Product Handler			+		
General Manager	++				
Financial Manager			+		

Table 21 shows the results of the centrality correlation coefficients between sub-groups and components. The set of identified centrality correlation is General Manager and Apex, Service Manager and Middle Agency with a strong correlation (++), and Financial Manager and Operational Core with a partial correlation (+). In the centrality correlation, some relationships could not be identified.

The next step is to evaluate the similarity correlation between the Bank group and Structure-in-5 Style. Table 22 shows the similarity matrix in order to compute measures of the Pearson Correlation using the in-degree relations.

Table 22 - The structural similarity based on In-degree

	Apex	Middle Agency	Operational Core	Techno-structure	Support
Service Holder	0,4082	-0,4082	1	0,6123	0,6123
Service Manager	-0,4082	0,4082	-1	-0,6124	-0,6124
Product Handler	0,6666	-0,6666	0,6123	0,1666	0,1666
General Manager	0,1666	-0,1666	0,4082	-0,1666	-0,1667
Financial Manager	-0,1666	0,1666	-0,6123	1	1

In this analysis we look across the columns to count in-degree. The analyses of the correlation measures show that:

1. The Service Holder sub-group is similar to the Operational Core ($r = 1$), meaning that these two actors have exactly the same ties to other actors.
2. The Service Manager sub-group is the opposite of Operational Core ($r = -1$). It has a partial positive association ($r = 0,4825$) with the Middle Agency component.
3. Product Handler sub-group has partial positive correlation with a two components: The Apex ($r = 0,66667$) and Operational Core ($r = 0,61237$).
4. The General Manager sub-group has a partial positive correlation ($r = 0,40825$) with the Operational Core component.
5. The Financial Manager sub-group is similar to Techno-Structure and Support components with $r = 1$ (strong positive association).

Table 23 - The structural similarity based on Out-degree.

	Apex	Middle Agency	Operational Core	Techno-structure	Support
Service Holder	0,6123	-0,6123	0,6666	-0,4082	-0,4082
Service Manager	-1	1	-0,4082	0,25	0,25
Product Handler	0,6123	-0,6123	0,6666	-0,4082	-0,4082
General Manager	1	-1	0,4082	-0,25	-0,25
Financial Manager	0,4082	-0,4082	0,1666	-0,6124	-0,6124

Table 23 shows the similarity matrix computing measures of the out-degree (sending). In this analysis we look across the rows to count out-degree. The Pearson correlation results show that:

1. The Service Holder sub-group has partial positive correlation with the Operational Core component ($r = 0,6666$).
2. The Service Manager sub-group is similar to the Middle Agency component with a strong positive association ($r = 1$).
3. The Product Handler sub-group has a partial positive correlation with two components: The Apex ($r = 0,6123$) and Operational Core ($r = 0,6666$).
4. The General Manager sub-group is similar to the Apex component with strong positive correlation ($r = 1$).
5. The Financial Manager sub-group has a partial positive correlation with Apex component ($r = 0,4082$).

Table 24 – Similarity Correlation between Bank sub-groups and Structure-in-Five components

	Apex	Middle Agency	Operational Core	Techno-structure	Support
Service Holder	+	-	++	+	+
Service Manager	--	++	--	-	-
Product Handler	+	-	+	-	-
General Manager	++	--	+		
Financial Manager	+	-		+	+

The coefficient correlation between two actors was presented in Table 16 (page 78). The correlation measures respectively, strong (++) or partial (+) positive correlation and strong (--) or partial (-) negative correlation. Table 24 summarises the similarity correlation results of the correlation analysis between Sub-groups and Components to achieve the goal Get Loan. If some relationships are not identified, the similarity must be re-calculated including other sub-groups of roles that interact to accomplish other bank tasks and sub-tasks, such as Deposit Money (see Figure 21).

5.4 MAPPING SUB-GROUPS TO COMPONENTS

In order to relate requirements and architecture it is necessary to compare the behaviour of the SIRA sub-group and the behaviour of a component in a specific architectural style. Table 25 summarises the results of the correlation analysis between sub-groups and components in the Bank example. The notation indicates the relations in terms of strong correlation (++), partial correlation (+), and no correlation ().

Table 25 – Correlation Analysis of the Bank Example

	Apex	Middle Agency	Operational Core	Techno-structure	Support
Service Holder			++	+	+
Service Manager		++			
Product Handler			+		
General Manager	++				
Financial Manager				+	+

The correlation analysis of the Bank example is not complete, but the aim of presenting the SIRA Framework models has been achieved. It is possible to derive the first architectural configuration based on these preliminary results. Hence, sub-groups and architectural components are related as follow:

1. The sub-group Service Manager is related to *Middle Agency*.
2. The sub-group Product Handler is related to *Operational Core*.
3. The sub-group Security Manager is related to *Strategic Apex*.

4. The sub-group Bank Controller is related to *Technostructure*.
5. The sub-group Financial Manager is related to *Support*.

5.4.1 The Bank Architecture in Structure-in-5 Style

In bank example, a first architectural configuration is obtained from the sub-groups assigned to components of the *Structure-in-5* Style. This style specifies that an organisation is an aggregate of five sub-structures (see Figure 7 on page 33).

Based on the results presented in Table 25, the bank architectural configuration is shown in Figure 27:

- The *Product Handler* is a component that carries out the basic tasks and procedures directly linked to the production of products and services (the acquisition of inputs, the transformation of inputs into outputs, and the distribution of outputs). This component performs roles such as Lender, Loan Handler, Deposit Holder and Account Handler, and other roles to provide the transactions of the Bank Products and Services.
- The *General Manager* is a component that makes executive decisions to ensure that the organisation fulfils its mission in an effective way and it defines the overall strategy of the organisation in its environment. This component performs roles such as Security Manager to provide key information about bank security strategic.
- The *Service Manager* establishes a hierarchy of authority between the Strategic Apex and the Operational Core. It consists of managers responsible for supervising and coordinating the activities of the Operational Core. This component performs roles to supervise and coordinate bank services.
- The *Bank Controller* serves the organisation by making the work of others more effective, typically by standardising work processes, outputs, and skills. It is also in charge of applying analytical procedures to adapt the organisation to its operational environment. This component performs roles to keep track on strategic information.
- The *Financial Manager* provides specialised services, at various levels of the hierarchy, outside the basic operating workflow. This component performs roles (such as Accountant) to handle accounting and financial services.

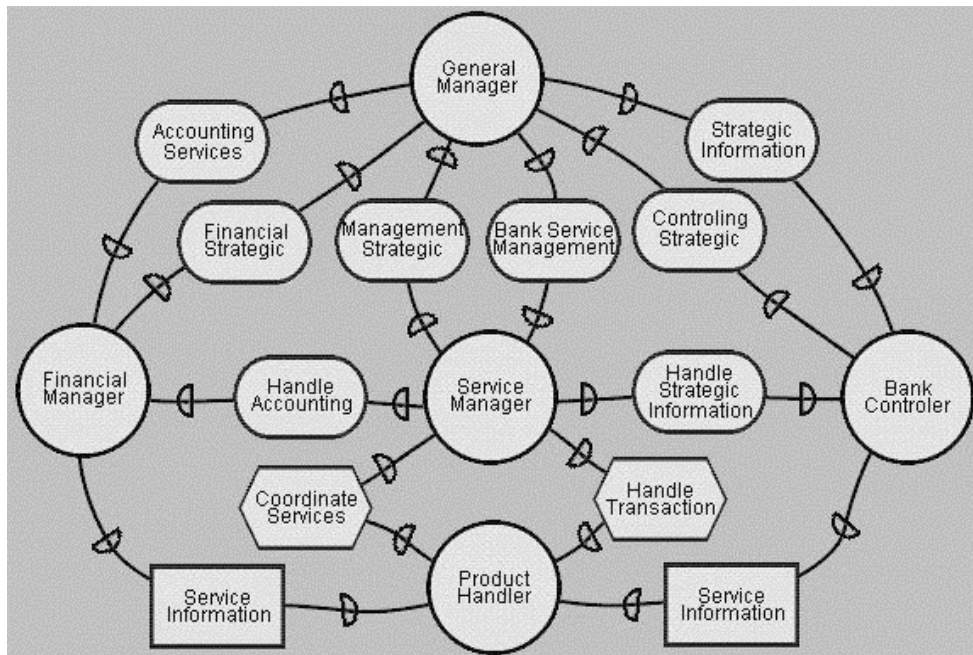


Figure 27 - Bank Architecture using the Structure-in-5 Style

5.5 SUMMARY

This chapter has given an overview of the SIRA Assignment Model. The Assignment Model explores the relationships between organisational model and architectural style and uses techniques of social analysis in order to derive these relationships. The assignment model uses structural equivalence (or similarity) to cluster roles into sub-groups and to match sub-group and components based on their patterns of interactions.

Cluster analysis pairs actors with the greatest similarity, recalculates similarities, and iterates until all actors are combined. The Structural equivalence of two actors is the degree to which the two actors have the same profile of relations across all other actors in the network. These techniques enable us to get a view of how the actors in one set are "approximately equivalent" and why different sets of actors are different. That is, these techniques enable the Assignment Model to describe the meaning of the sub-groups and the mapping of sub-groups to an organisational architecture in a general way. The bank example was used to show this mapping. The result is the first architectural configuration of the bank structure-in-5 architecture.

The next chapter will provide a case study for tracking the complete process of the SIRA Framework.

Chapter 6

The e-commerce case study

This chapter presents the modelling steps of the SIRA Process with an e-commerce example.

6.1 INTRODUCTION

In the SIRA Framework, Multi-Agent Systems are defined as organisational groups. An organisational group is a collectivist view that is based on cooperation through pre-established roles, responsibilities and interactions.

Having presented the SIRA models for modelling agent organisations, we will now present an e-commerce application following the SIRA Process. The process considers the influence of social organisational concepts on the requirements (in form of tasks and goals) and specifies the modelling steps to derive an architectural configuration for a Multi-Agent System in a particular domain.

This chapter is structured as follows: Section 6.2 introduces the e-commerce example that will be used to present each process activity. Section 6.3 outlines the *Organisational Model Specification*. Section 6.4 outlines the *Assignment Model of the e-commerce example*. Section 6.5 outlines the last activity: *Architectural Configuration*. Finally, section 6.6 summarises the chapter.

6.2 THE E-COMMERCE EXAMPLE

The Medi@ example will be used to describe the activities of the SIRA Process. The Media Shop is a store selling and shipping different kinds of media items such as books, newspapers, magazines, audio CDs, videotapes, and the like. To increase market share it decides to open up a B2C retail sales on the Internet, called Medi@ [Castro et al, 2002] (The i* strategic dependency model for this example is shown in chapter 2 - see Figure 4 on page 29).

As late requirements analysis proceeds, Medi@ is given additional responsibilities, and ends up as the *dependor* of several dependencies. This chapter presents the i* strategic rationale (SR) model for the Medi@ system. The i* Strategic Rationale model is a supplementary model and provides support to model the reasons associated with each actor and their dependencies. The Strategic Rationale model consists of nodes (goal, task, resource and softgoal) and link (mean-end and task decomposition), as discussed in section 2.6 on page 27, in Tropos project.

The analysis depicted in Figure 28 (on page 92) focuses on the software (Medi@), instead of an external stakeholder. The figure postulates a root task *Internet Shop Managed* to provide sufficient support to the softgoal Increase Market Share. That task is firstly refined into goals *Internet Order Handled* and *Item Searching Handled*, softgoals *Attract New Customer*, *Secure* and *Usable* and tasks *Produce Statistics* and *Maintenance*. *Internet Order Handled* is achieved through the task *Shopping Cart*, which includes the subtasks: *Select Item*, *Add Item*, *Check Out*, and *Get Identification Detail*. These are the main process activities required to design an operational on-line shopping cart [Conallen, 2000]. The latter task is achieved either through the sub-goal *Classic Communication Handled* to deal with phone and fax orders or *Internet Handled* to manage secure or standard form orderings. To allow for the ordering of new items not listed in the catalogue, *Select Item* is also further refined into two alternatives

sub-tasks, one dedicated to select catalogued items, the other to pre-order unavailable products. To provide sufficient support (++) to the *Adaptable* softgoal, Adaptation is divided into four subtasks that deal with catalogue updates, system evolution, interface updates and system monitoring. The goal *Item Searching Handled* might alternatively be fulfilled through tasks *Database Querying* or *Catalogue Consulting* with respect to customers' navigating desiderata, i.e., searching with particular items in mind by using search functions, or simply browsing the catalogued products.

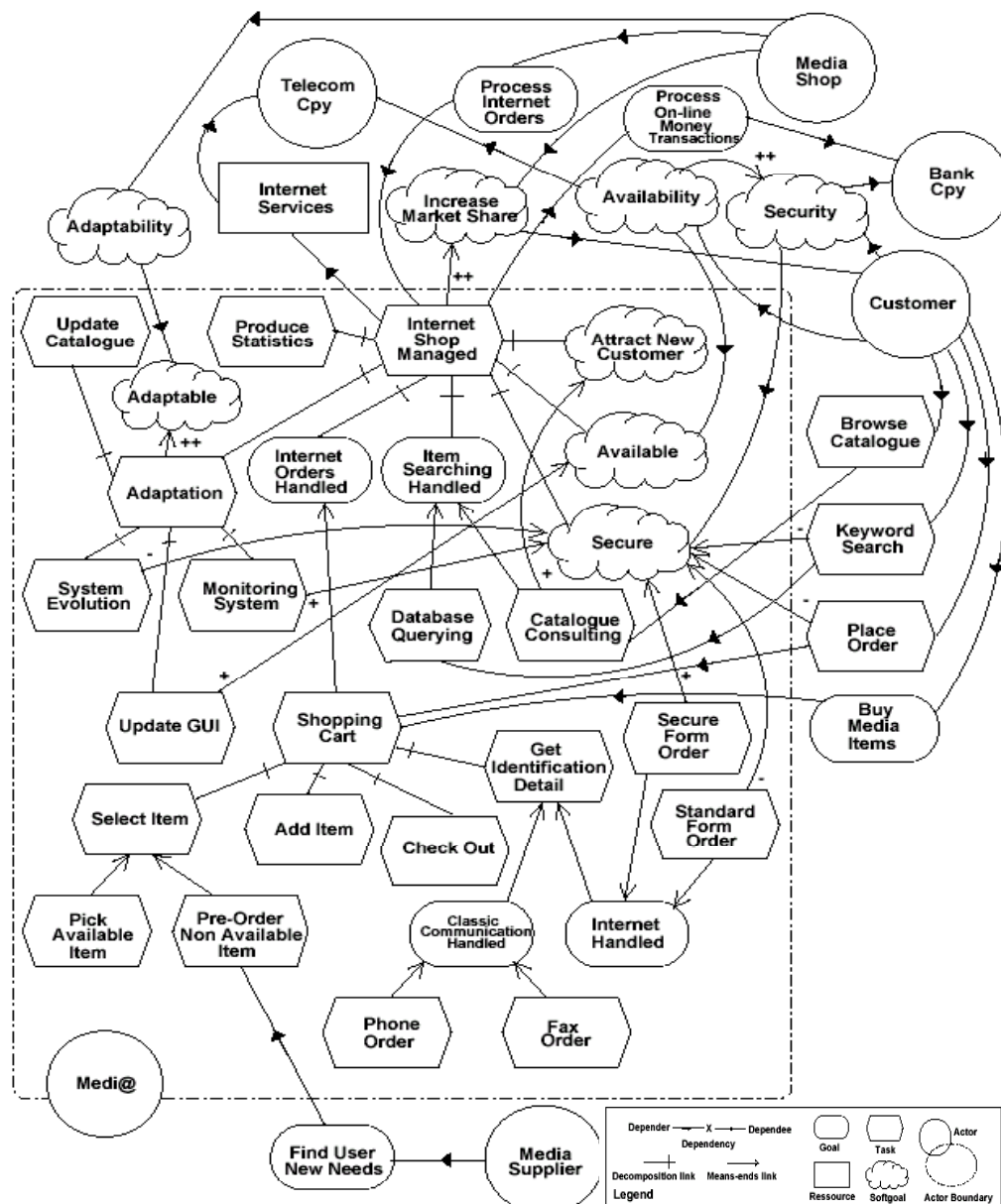


Figure 28 - Strategic Rationale model for Medi@ actor

As already pointed out, Figure 28 introduces softgoal contributions to model sufficient/partial positive (respectively, ++ and +) or negative (respectively, -- and -) support to the softgoals *Secure*, *Available*, *Adaptable*, *Attract New Customers* and *Increase Market Share*. The result of the means–ends analysis is a set of (systemic and human) actors who are *dependees* for some of the dependencies that have been described.

Each SIRA Process activity will be tracked in the remainder of this chapter, using this Medi@ actor as an example.

6.3 THE ORGANISATIONAL MODEL OF THE MEDI@ EXAMPLE

The first activity of the SIRA process is the specification of the Organisational Model. Goals, softgoals and tasks of the requirement models are the basis for an organisational context, which is used to identify roles and interactions that Multi-agent Systems have to provide. This activity supports the mapping from the i* requirement model onto SIRA Organisational Model.

As seen in Figure 12 on page 44, this activity includes three sub-activities: Goals and Tasks Refinement, Roles Identification, and Architectural Selection. Each one will be fulfilled in the next sub-sections.

6.3.1 Task Refinement

One way to express the behaviour of an actor with respect to its environment is in terms of the expected functionality of the organisational group, that is, what the group is expected to do or produce. In the *Tasks Refinement*, actors are *Organisational Groups* with responsibilities to be performed. The group responsibilities are identified from main dependencies among actors of the requirement models.

From Figure 28, Medi@ actor is an example of organisational group. *Media Shop* and *Customer* actors have a dependency of Medi@ actor to provide certain goals. The main dependencies of Medi@ actor is to fulfil the goals identified as: “Process Internet Orders”, “Buy Media Items” and “Find Users New Needs”.

Means-end analysis is applied for the discovery of the means (task and interactions) that contribute to achieve the objective. The main goals are refined into task and sub-tasks. Task refinement is related to the identification of sub-tasks and interaction sequence. This results in the refinement of tasks into sub-tasks and in the identification of group responsibilities. In particular, means-end analysis aims at identifying tasks that provide the means for achieving a goal [Bresciani et al, 2004].

Figure 29 shows a refinement of the goal Process Internet Orders. The goal is achieved through the task *Internet Shop Managed*. This principal task is performed through a set of tasks such as: *Monitoring System*, *Shopping Cart*, *Secure Form Order*, and *Consult Catalogue*. The refinement is further applied to the tasks *Shopping Cart* and *Consult Catalogue*, as follows:

1. The task *Shopping Cart* is achieved through the four sub-tasks: *Select Item*, *Add Item*, *Check Out* and *Get Identification Detail*;
2. The task *Consult Catalogue* is achieved through the tasks *Browse Catalogue* or *Keyword Search*.

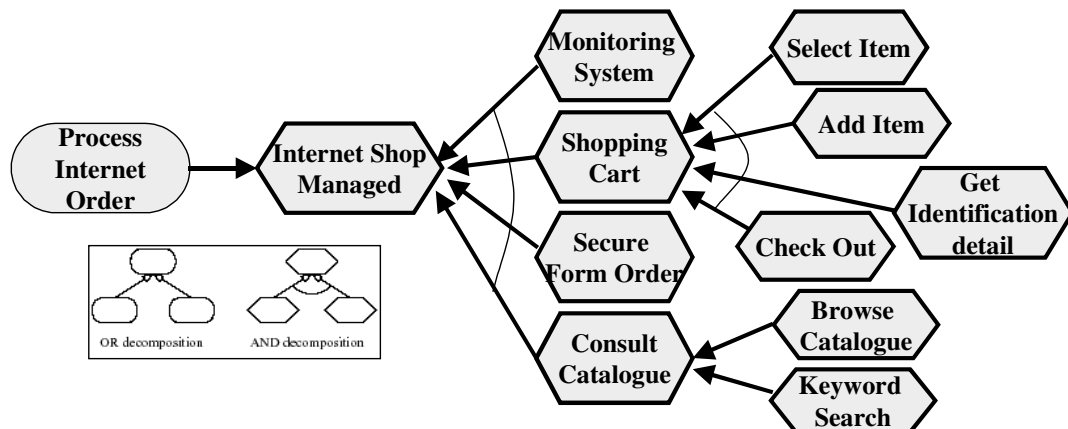


Figure 29 - Task refinement for the goal Process Internet Orders

6.3.2 Role Identification

In the *Role Identification* sub-activity, roles are identified to represent the division of labour among members. The organisational group is redefined into a set of roles and interactions. Every role communicates and interacts with other roles. So, role identification is concerning with the sequence and types of tasks that can be performed by an actor in collaboration with others.

In general, roles are captured in a specific and bounded domain. For example, the e-commerce domain can identify roles such as Buyer, Seller or Shop Manager. Medi@ roles are identified from Medi@ goals (*Process Internet Orders*, *Buy media items*, and *Find User new Needs*). In case of the goal *Process Internet Orders*, the set of roles for Medi@ Group can be defined as follows:

- *Buyer* is a role played by a customer that needs to buy items;
- *Shop Manager* is responsible for handling and controlling the internet shopping services, such as Internet shop Managed task.
- *Seller* is responsible for supplying customers with a web interface to keep track of items, such as Consult catalogue and Place order tasks;

- *Order Provider* is responsible for handling the process services that will be executed for a given order, such as Process Order task;
- *Security Manager* is responsible for monitoring and controlling security check services, such as Customer profile and Security form order tasks.

The role interaction is expressed within a particular organisational group or sub-groups. It is represented as a directed graph, with the nodes representing the roles and the lines representing the interaction sequences among them. This representation makes it possible to distinguish between roles that interact and those that do not, and to analyse the relations between sending and receiving information. Figure 30 shows an interaction graph with the roles necessary to fulfil the group goal *Process Internet Orders* in the e-commerce example.

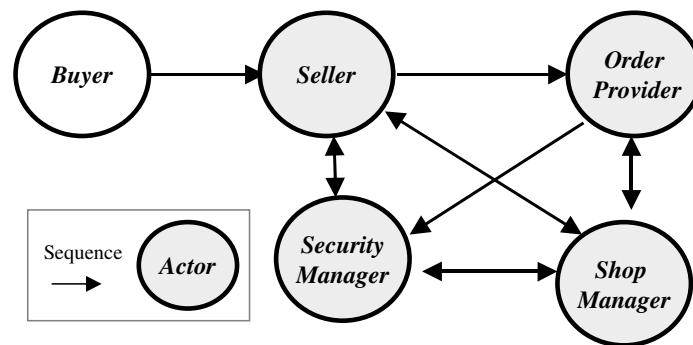


Figure 30 - Role interaction of the goal Process Internet Orders

Using a matrix representation, the interaction graph can be viewed as a binary matrix. Binary relations are represented with zeros and ones, indicating the presence or absence of each logically possible relation between pairs of actors. By convention, the sender of a relation is the row and the target is the column. In the e-commerce example, the directed graph choices among roles in Medi@, as seen in the 4 by 4 matrix (see Table 26). The rows represent the source of directed ties, and the columns the targets; *Seller* chooses *Order Provider* here, but *Order Provider* does not choose *Seller*.

Table 26 - Interaction matrix of the task “Internet Shop Managed”

	Seller	Shop Manager	Order Provider	Security Manager
1. Seller	--	1	1	1
2. Shop Manager	1	--	1	1
3. Order Provider	0	1	--	1
4. Security Manager	1	1	0	--

6.3.3 Architectural Selection

In requirement models, constraints come primarily from the business goals surrounding the system in the form of functional and non-functional requirements. The e-business application domain is one special case of these constraints where distributed and open architectures (e.g., Web, internet, agent, and peer-to-peer software systems) are becoming increasingly important.

The e-commerce example has three (soft)goals (Availability, Security, Adaptability) in the late requirements model (as seen in Figure 28 page 92). The first goal is Availability, which allows system agents to automatically decide which catalogue browser best fits customer needs or navigator/platform specifications. The second key softgoal is Security, which supports a number of security strategies while taking into account environment configurations. The third goal is Adaptability, meaning that catalogue content, database schema, and architectural model can be extended or modified to integrate new and future web-related technologies.

The choice of architectural style is based on the application domain. Once the architectural style has been chosen, it has to conform to the functional and non-functional requirements of the application domain. Additionally, a specific refinement, resolution or assignment is selected on the basis of the qualitative preferences dictated by positive contributions to high-priority softgoals. In the NFR framework, qualities are represented as *softgoals*. Analysing them involves refining these softgoals into sub-goals that are more specific and more precise, and then evaluating alternative architectural styles against them. For each type of alternative, decisions have to be made which, in the end, will produce different architectures.

Table 27 - The Correlation catalogue for Medi@

CORRELATION	SEC	ADAPT.	AVAIL.
Struct-5	+		+
Pyramid	++	+	+
Joint-Vent (v)	+	++	++
Cooptation	-	++	-

The set of organisational styles is compared against the three (soft)goals. The correlation catalogue was shown in Table 1 on page 46. The organisational patterns focus both on how to organise the components expected in an e-business system and on the intentional and social dependencies governing these components. Table 27 summarises strengths and weaknesses of the architecture styles with respect to the software quality attributes of the Medi@ application. Recall the notations: +, ++, -, --, that model partial/positive, sufficient/positive, partial/negative and sufficient/negative contributions, respectively.

An exhaustive evaluation is difficult at this point. But, given these preliminary results, we can argue that the *Joint-Venture* architecture is a better fit for systems and applications that need

open and cooperative components like the e-commerce example. The interaction graph for Joint-venture architecture is shown in Figure 31.

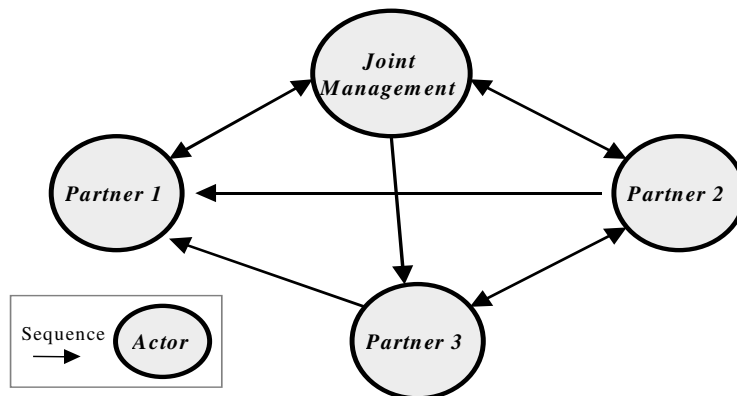


Figure 31 - Interaction Graph of the Join Venture Style

The interaction matrix of the Joint-venture architecture is shown in Table 28. Remember that the rows represent the source of directed relations and the columns the targets. Partner_2 send information to Partner_1, but Partner_1 does not sends information to Partner_2. To perform the social analysis it will be necessary to include two relations concerned to the information exchange among the actors Partner_1, Partner_2, and Joint Management. The Joint Management needs to receive information about partner responsibilities to define the organisational strategic management.

Table 28 - Interaction matrix of the Join Venture Style

	Joint Management	Partner_1	Partner_2	Partner_3
1. Joint Management	--	1	1	1
2. Partner 1	1	--	0	0
3. Partner 2	1	1	--	1
4. Partner 3	0	1	1	--

The result of this sub-activity is an architectural style with a collection of components. The following sub-activities analyse how the role and responsibilities from an organisational model will be assigned to each architectural component.

6.4 ASSIGNMENT MODEL OF THE MEDI@ EXAMPLE

As seen in section 3.4.2 on page 46, the Assignment Model definition includes two sub-activities (as depicted in Figure 14): Cluster Analysis and Similarity Analysis. Firstly, the cluster analysis groups the actors that have the same pattern of relations into sub-groups.

Finally, the similarity analysis measures the linear relationship between the sub-groups and components. This sub-activity will be fulfilled in following sub-sections.

6.4.1 Cluster Analysis

The *Cluster Analysis* sub-activity analyses the role interaction graph to measures the extent to which a role is connected to other roles in terms of centrality and structural equivalence.

Degree centrality measures the in-degree and out-degree relations among roles. The in-degree result represents the role prominence in a group (receiving relations) and the out-degree result represents the role influence in a group (sending relations). In other words, greater the number of relations, higher is the degree centrality of the role.

From a directed graph both in-degree centrality and out-degree centrality may be measured. In the e-commerce interaction graph (see Figure 30 on page 95), lets examine the in-degrees and out-degrees to measure what role is more central. When evaluating the interactions to achieve the task *Internet Shop Managed*, the roles *Seller* and *Shop Manager* have the greatest out-degrees, and might be regarded as the most influential roles. *Shop Manager* has the highest in-degree result. Degree centrality results for e-commerce example are shown in Table 29. From these results, there is a partial amount of centralisation in the whole group. That is, the *Shop manager* is the most central role in terms of in-degree and out-degree. The centrality of the other actors is partially distributed in this group (in most cases, it is 66,66).

Table 29 - Degree Centrality of the task *Internet Shop Managed* (g=4)

	Out degree	In degree	Nrm-Out	Nrm-In
1. Seller	3	2	100	66,66
2. Shop Manager	3	3	100	100
3. Order Provider	2	2	66,66	66,66
4. Security Manager	2	2	66,66	66,66

Closeness centrality measures how close a role is to other roles. Roles which are able to reach other roles at shorter path lengths, or which are more reachable by other roles have favoured positions in an organisation.

Table 30 - Closeness Centrality for the task *Internet Shop Managed* (g=4)

	Farness	Closeness
1. Seller	3	100
2. Shop Manager	3	100
3. Order Provider	3	100
4. Security Manager	4	75

In the e-commerce example, the results for the task *Internet Shop Managed* is shown in Table 30. The *Seller* and *Shop Manager* are the closest roles using this method. This is because the sum of geodesic distances to other roles (a total of 3, across 3 other actors) is the least (in fact, the minimum possible sum of geodesic distances is 3).

Recall from sub-section 5.2.2 on page 78, ‘structural similarity’ means that role have the same pattern of relations with all other roles. Consequently, the cluster analysis computes the measures of the degree to which roles are similar, and uses this as the basis for seeking to identify sets of roles that are very similar to or distinct from roles in other set. The result of this analysis is a set of sub-groups, each of which clusters a set of similar roles.

The matrix representation can be a lot more precise in assessing structural similarity. The e-commerce matrix was shown in Table 26 on page 95. Looking across the rows counts out-degree and looking down the columns counts in-degree. Even more generally, two roles are structurally equivalent to the extent that the profile of scores in their rows and columns are similar. In the e-commerce example, only the roles *Seller* and *Shop Manager* can be analysed in terms of their similarity degree.

Lets compute the correlation coefficient to measure the strength of the linear relationship between roles in the e-commerce example. The result is shown in Table 31. The correlation coefficient is always between -1 and +1. The person correlation formula and its interpretation were introduced in chapter 4 (page 78). The coefficient correlation between *Seller* and *Shop Manager* is a partial negative association ($r = -0,333$). These two roles belong to different sub-groups because they do not have the same pattern of relations.

Table 31 - Role similarity of the e-commerce example

	1	2	3	4
1. Seller	1			
2. Shop Manager	-0,33333333	1		
3. Order Provide	0,577350269	-0,57735	1	
4. Security Manager	-0,57735027	-0,57735	0	1

This cluster analysis concludes that the Medi@ system has four sub-groups of roles. In this e-commerce example, each sub-group is defined as follows:

- The *Store Front sub-group* clusters roles (such as *Seller* with input responsibilities) to provide a usable front-end web application.
- The *Store Manager sub-group* clusters roles (such as *Shop Manager* with managing responsibilities) to keep track of all web information about customers, products, sales, bills and others data of strategic importance to Media Shop.
- The *Order Processor sub-group* clusters roles (such as *Order Provider* with order-processing responsibilities) to support the processing for a given order.

- The *Security Manager sub-group* clusters roles (such as Security Manager with managing responsibilities) to supervise and control security, availability and adaptability of the Medi@ system.

In the Joint Venture Architecture (see Figure 31 on page 97), let's examine the in-degrees and out-degrees to measure what role is more central. When evaluating the interactions, the actors *Joint Management* and *Partner_2* have the greatest out-degrees, and might be regarded as the most influential actors. *Partner_1* has the highest in-degree result. Degree centrality results for the Joint Venture architecture are shown in Table 32 on page 46. From these results, there is a partial amount of centralisation in the whole group. But the centrality of the other actors is partially distributed in this group (in most cases is 66,66).

Table 32 - Degree Centrality Measures of the Joint Venture Style (g=4)

	Out degree	In degree	Nrm-Out	Nrm-In
1. Partner_1	2	3	66,66	100
2. Joint Management	3	2	100	66,66
3. Partner_2	3	2	100	66,66
4. Partner_3	2	2	66,66	66,66

Closeness centrality measures in the Joint Venture architecture are shown in Table 33. All actors have identical distributions of closeness and appear to be equal in terms of their structural position.

Table 33 - Closeness Centrality for the Joint Venture Architecture (g=4)

	Farness	Closeness
1. Partner_1	3	100
2. Joint Management	3	100
3. Partner_2	3	100
4. Partner_3	3	100

The results of this cluster analysis sub-activity are the basis of the similarity analysis between sub-groups and components.

6.4.2 Correlation Analysis between Group and Components

In order to relate requirements and architecture it is necessary to compare the behaviour of the SIRA sub-group and the behaviour of a component in a specific architectural style. The first correlation between Medi@ sub-groups and Joint Venture components is based on centrality measures of degree and closeness, as follows:

1. The *Store Front* has out-degree and closeness value of 100. It can be related to Joint Management or Partner_2, which have similar degree and closeness measures.
2. The *Store Manager* is the most central sub-group from e-commerce example. It has degree (both in-degree and out-degree) and closeness value of 100 in centrality methods. It can be related to Joint Management or Partner_2, which also have out-degree and closeness value of 100.
3. The *Order Processor* has degree value of 66,66, and closeness value of 100. It can be related to Partner_3, which has degree value of 66,66 and closeness value of 100.
4. The *Security Manager* has degree value of 66,66 and closeness value of 75. It can be related to Partner_3, which has degree value of 66,66 and closeness value of 100.

Table 34 shows the centrality correlation results. The notation indicates the relations between sub-groups and components in terms of strong positive correlation (++) and partial positive correlation (+).

Table 34 - Centrality Correlation between Medi@ Sub-groups and Joint Venture components

	Nrm-Out	Nrm-In	Closeness	Degree Score	Central Actor
Joint Venture	100	100	100	1	Partner_2/Joint Management
Medi@ group	100	100	100	1	Store Manager
Structure-in-5 Components					
Bank Sub-Groups		Partner_1	Joint Management	Partner_2	Partner_3
Store Front			++	++	
Store Manager			+	+	
Order Processor					++
Security Manager					+

The next step is to check if there is at least one similarity correlation between sub-group and the components of the architectural style. These sub-groups can become the architectural components. The similarity correlation analysis is based on a Pearson's correlation coefficient (as seen in sub-section 5.3.2 on page 83).

In the Medi@ e-commerce example, Table 35 shows the similarity matrix computing measures of the out-degree (sending). The Pearson correlation results show that:

1. The Store Front sub-group has a partial positive correlation with Partner_1 ($r = 0,33$). This means that Store Front sub-group is more similar to the Partner_1 component.
2. The Store Manager sub-group has a strong positive correlation with Join Management component ($r=1$). The pattern of relations of these actors is similar.

3. The Order Processor sub-group has a partial positive correlation with Partner_1 and Partner_2 ($r = 0,57735$). This means that these actors have partial similarity of relations.

Table 35 - Similarity based on Out-degree for the e-commerce example

	Partner_1	Joint Man.	Partner_2	Partner_3
Store Front	0,333333	-0,333333	-0,333333	-0,57735
Store Manager	-1	1	-0,333333	0,57735
Order Processor	0,577350	-0,57735	0,57735	-1
Security Manager	0,577350	-0,57735	0,57735	0

Table 36 shows the similarity correlation with measures of the in-degree.

Table 36 - Matrix based on In-degree for the e-commerce example

	Partner_1	Joint Man.	Partner_2	Partner_3
Store Front	0,577350	-1	1	0
Store Manager	-0,333333	0,57735	-0,57735	-0,57735
Order Processor	-0,57735	0	0	0
Security Manager	-0,333333	0,57735	-0,57735	0,57735

The Pearson correlation results for in-degree measures show that:

1. The Store Front sub-group has a strong positive correlation with Partner_2 ($r = 1$). This means that Store Front sub-group is similar to the Partner_2 component.
2. The Store Manager sub-group has a partial positive correlation with Joint Management component ($r=0,577$).
3. The Security Manager sub-group has partial similarity with Joint Management and Partner_3, with a partial positive correlation ($r = 0,57735$) with regard to the in-degree result.

The similarity coefficient correlation models strong (++) or partial (+) positive association, and strong (--) or partial (-) negative associations. Table 37 summarises the in-degree and out-degree results of the similarity correlation of the e-commerce example. If some relationships are not identified, the similarity should be re-calculated to include other sub-groups of roles that interact to accomplish a goal.

Table 37 - Similarity Correlation Table of the Medi@ e-commerce example

	Partner_1	Joint Management	Partner_2	Partner_3
Store Front	+	--	+	-
Store Manager	--	++	-	+
Order processor	+	-	+	-
Security Manager	+			+

Based on these centrality and similarity correlation, it is possible to derive the first configuration of the e-commerce architecture.

6.5 ARCHITECTURAL CONFIGURATION

This activity in the *SIRA Process* defines the map between requirement model and architectural style. Each joint venture component will be related to each sub-group identified in the *Cluster Analysis*.

Table 38 - Correlation Analysis of the Medi@ e-commerce example

	Partner_1	Joint Management	Partner_2	Partner_3
Store Front			+	
Store Manager		+		
Order processor	+			
Security Manager				+

In this case study, the selected style was the *Joint Venture* (Figure 6 page 33). Accordingly, the Medi@ architecture can be defined as containing four autonomous components: a manager component named *Joint Management* and other components identified as *Partner_1*, *Partner_2*, *Partner_3*, and *Partner_4*. Medi@ sub-groups are related to the architectural components of Joint Venture Style, based on the results of the correlation analysis as shown in Table 38:

- *The Store Front* sub-group is related to the *Partner_2* component to provide customers with a usable front-end web application;
- *The Store Manager* sub-group is related to *Joint Management*. It keeps track of all web information about customers, products, sales, bills and others data of strategic importance to Media Shop.
- *The Order Processor* sub-group is related to *Partner_1* to support the processing for a given order in *Store Front*.
- *The Security Manager* sub-group is related to *Partner_3* to manage and control the security, availability and adaptability of the Medi@ System.

The Medi@ architecture is shown in Figure 32:

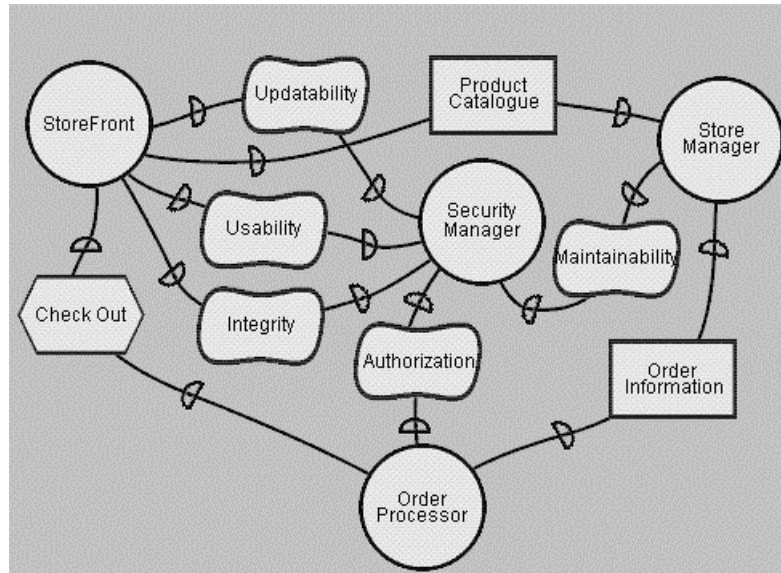


Figure 32 - Medi@ system architecture model

6.6 SUMMARY

This chapter has presented a process to track the transition between requirements and architecture. The SIRA Process focuses on a systematic way to support the transition from requirement specification to architectural model. The process takes the organisational perspective as starting point and generates the architectural configuration of the Multi-Agent System. The architecture describes the Multi-Agent System at a macroscopic level in terms of a manageable number of sub-groups.

E-commerce example was used to describe each activities of the SIRA Process. Based on the quality attributes of the Medi@ application, the Joint Venture architecture was selected from a set of organisational styles.

The next chapter will concludes this thesis with final considerations, contributions and points to future works.

Chapter 7

Conclusion

This chapter outlines the final considerations and results of this thesis.

7.1 FINAL CONSIDERATIONS

With the advances in software development in recent years, it is appropriated to establish the relations between requirements and architectural elements. Although the conceptual differences between them, system-modelling techniques must encompass a systematic way to deal with the relationships between requirement and architectural models.

In this research we have made a first step in the direction of analysing requirements of multi-agent systems (MAS) as a software architecture style on the basis of organisational concepts. In the SIRA Framework, multi-agent system corresponds to the organisational structure, in which actors are members of a group in order to perform specific tasks. An organisation comprises groups, members, roles and interactions. Roles are one of the basic components of our organisational model. A member assigned to roles does not work in isolation but interact and cooperate with other roles. Role descriptions in the organisational model identify functions or services necessary to achieve group goals and enable to abstract from the actors that will perform the role. Furthermore, roles must describe the necessary capabilities that must be performed by any member performing that role. Roles are organised into a role-relationship network. Roles are identified by goals and interactions. The goals of a role describe the responsibilities that a member has to perform. Interactions describe the relations between roles and specify how member interacts with each other when performing a role.

In order to say something about the relation between organisational requirements and architectures, we define which are the relevant properties of multi-agent architectures, and which alternatives exist for each of those properties. The organisational styles are used as a type of multi-agent topology. The functional decomposition of multi-agent systems is a kind of division of labour among agents. It is especially applicable in multi-agent models where agents have predetermined roles (in form of tasks and interactions). This thesis was structured of four main parts:

- A review and discussion of organisational concepts in requirements models and multi-agent systems;
- An organisational model on the basis of which our framework analyses functional and non-functional requirements for identifying roles and interactions of a social environment;
- An assignment model on the basis of which our framework analyses the relation among *Roles*, *Sub-groups*, and *Components* for deriving an architectural configuration;
- A process on the basis of which our framework carries out the transition from requirements to architecture.

Our Framework is located in Tropos project context. In this thesis research, we observed that the existent information in the requirement models of Tropos is not enough to derive

architectural components. In order to address this issue, our framework complements the requirement with information about groups, roles and interactions. And, to derive the architectural model, we apply some heuristics to relate sub-groups and architectural components.

7.2 CONTRIBUTIONS

In spite of the significant progress accomplished in the areas of requirement and architecture, we still need frameworks, techniques and tools to support the systematic achievements of the architectural objectives in the complex context of the stakeholders' needs. The main contribution of our thesis is to reduce the semantic gap between requirements and architecture. The key concepts addressed in this thesis are the following:

- SIRA Framework provides an intermediate model reducing the semantic gap between software requirements and high-level architectural model
- Requirement model extended with organisational theory in which actors are members of a group in order to perform specific roles. Role does not work in isolation but interacts and cooperates with others roles;
- Requirements description supported by a development methodology, the Tropos Methodology;
- Software architectures and requirements modelling described using the same concepts. This is useful to provide the same understanding about the multi-agent organisation and its relation to the requirements and architectural model;
- Multi-Agent requirements mapped into architecture maintaining consistency. This integration makes it possible provide the traceability from requirements specification to architectural model. It is a means of achieving integrity and completeness of that integration;
- The architectural components represent or are associated to the main organisational requirements, which later or will be fulfilled by an agent of a Multi-Agent system.

In summary, our research contribution was focused on given an organisational requirements specification, how to model an organisational architecture and how to assure that the architectural components will meet the system requirements.

7.3 DISCUSSION OF RESULTS

Organisational concepts lend to modelling and reasoning about the distribution of capabilities and assignment of responsibilities within the Multi-Agent Systems, and to show how architectural components are intended to contribute to the overall goals. The rationale for using such concepts to modelling social groups is based on the human organisations, where

different members may have different business goals and rationales. Agents enable the various interests within an organisation to be expressed by linking goals to the roles of this organisational model.

Agents offer a way to deal with complex systems that have multiple and distinct components, and are often used as a metaphor for autonomous entities. Even though agents are adequate to represent rational entities, because of their autonomy, flexibility in content and participation, reactive and proactive behaviour, communication and reasoning capabilities, when considering an organisational setting more is needed. In an organisational environment, the behaviour of the global system and the collective aspects of the domain - such as stability over time, predictability and commitment to overall aims and strategies - must be considered. That is, the concept of desirable social behaviour is of utmost importance when multi-agent systems are considered from an organisational point of view. This leads to a rising awareness that multi-agent systems can best be understood and modelling inspired by social concepts [Castelfranchi, 2000], [Artikis et al, 2001], [Zambonelli et al, 2001]. This is, in many ways, a novel concept within agent research, even if sociability has always been considered an important characteristic of agents.

7.3.1 Agents and Requirements

The requirement engineering research has recognized the importance of goal modelling [Dardenne and Lamsweerde, 1993], [Anton, 1997], [Yu, 2001]. However, goals are typically used to guide the establishing of requirements of business process, and serve as criteria for requirements completeness. In this thesis, goals are used during the requirements analysis and architectural modelling. Goals denoting quality requirements provide an effective means for denoting constraints over design efforts. These goals served as a guide in the search for architectural alternatives, and serve as criteria for choosing among them. The goals elaborated in requirements models are used to derive software architectures. High-level goals are refined, originating other lower-level goals, which is assigned to components of multi-agent architectures.

An organisation is a goal-oriented coalition of software agents in which the agents are engaged in one or more tasks [Far et al, 2000]. The structure of an organisation defines the roles of the various internal actors, their responsibilities, defined in terms of tasks and goals. Secondly, an organisation defines how to coordinate the activities of various components and how they depend on each other.

An organisational architectural style defines the class of organisational structure, and offers a set of requirements that can influence the division of labour and the interaction mechanisms, thereby affecting how the organisations functions.

7.3.2 Agents and Architectures

While several architectural styles have been recognised and described in the literature, multi-agent systems have not yet gone through this process. The agent autonomy refers to a component not depending on the properties or the states of other components for its

functionality. Nevertheless, to construct meaningful multi-agent architectures, these components are able to interact, usually by a predefined protocol. Due to their unique suitability to several classes of computational problems, it is important to characterise MAS as a software architecture style and to provide the architectural model of such systems. Such model will equip system designers with a family of appropriate solutions for highly distributed problems in open, heterogeneous, dynamic and information rich environments. In this thesis we make a preliminary effort towards this goal.

The use of intermediary models among requirements and architectures could be useful to describe the mapping between requirement and architectural models. The Multi-Agent approaches can provide some degree of relationships between requirements and organisational structure. In this thesis, we have some preliminary step to capture systematically the overall relationships of an agent organisation and to provide it in the same notation for architectural modelling.

With respect to the organisation or architecture of a multi-agent system we have distinguished the aspects of structure and functional decomposition. The organisational style is the most promising type of multi-agent structure. It extends the decentralised computational market structure by introducing notions of resources, dependency and cooperation. The functional decomposition aspect of a multi-agent system is a kind of description that stems from general systems theory. It is especially applicable in multi-agent models where agents have predetermined responsibilities (in terms of tasks, goals and resources).

7.4 FUTURE WORKS

The integration between Requirements Engineering and Software Architecture encompasses a wide set of activities, and this thesis has not been able to tackle all of these in depth. There remain a number of problems with particular aspects of this integration, and a number of activities that the SIRA Process does not really address.

Several aspects of the SIRA models require further works:

- to test the usefulness of SIRA Framework in a broad range of applications and to gather more information about the associated strengths and weaknesses;
- to consider the study of norms, more than the specification of the normative structure of an organisational model. Norms that guide organisations are usually described in regulations, with a high level of abstraction. Mechanisms are needed to explicitly specify how norms expressed in abstract regulations are translated into concrete norms for the specific organisation being modelled;
- to resolve conflicting decisions. Mechanisms are needed to provide resolution and explanation when multiple conflicting decisions are made for the same part of the SIRA models, such as goal refinement, cluster analysis and correlation analysis.
- to enhance the particular features of the SIRA models in terms of improving the guidelines of the SIRA Models by incorporating practical experiences;

Additionally, in our future work we intend to extend SIRA Framework in the following directions:

- The main direction for future research is obviously the development of practical tools to build SIRA models. Tools to build agent organisations will ease the design, implementation and verification of multi-agent systems. Such tools should be able to guide the SIRA process, and enable the specification, and the automatic configuration of agent societies according to the organisational model.
- The work in this thesis opens up many new researches areas. Although demonstrated in the context of the Tropos Project, we believe that SIRA Framework has potential for wide applicability and provides a generic framework of bridging requirements into architecture and design (e.g., UML). Further validation is needed by exploring SIRA Organisational and Assignment Models using additional requirements and architecture methods.

7.5 SUMMARY

We have presented and demonstrated the relation between requirement and architectures based on organisational concepts. Our framework was developed in response to the observation that the requirements modelling plays an important role in deriving the architectural model.

Our approach was used to develop a process for these activities. The application of these models to the requirements process is novel, and opens up a new area of research to integrate the activities of the software engineering.

Bibliography

- Alencar, F.M.R., Castro, J.F.B., Cysneiro, G.A.A., Mylopoulos, J. (2001): "Integrating Organizational Requirements and Object Oriented Modeling". 5th IEEE International Symposium on Requirements Engineering, Toronto.
- Ambriola, V. and Gervasi, V. (1996): "Representing Structural Requirements in Software Architecture". Published by Chapman & Hall. IFIP.
- Anton, A. I. (1997): "Goal Identification and Refinement in the Specification of Software-Based Information Systems," Ph.D. Thesis, Georgia Institute of Technology, Atlanta GA.
- Anton, A.I and Potts, C. (1998): "The use of goals to surface requirements for evolving systems," in Proc. of the 1998 Int. Conference on Software Engineering (ICSE'98), Kyoto, Japan, 19–25 April, pp 157–166.
- Artikis, A., Kamara, L., Pitt, J. (2001): "Towards an Open Agent Society Model and Animation". Proc. of the 2nd. Agent-Based Simulation Workshop, Passau, pp. 48 – 55.
- Bachmann F, et all (2000): "Software Architecture Documentation in Practice: Documenting Architectural Layers". Special Report. CMU/SEI-2000-SR-004. March.
- Banco Central do Brasil - BACEN (2005) - <http://www.bacen.gov.br/?SFNCOMP>
- Bass, L., Clements, P. and Kazman, R. (1998): "Software Architecture in Practice". Addison-Wesley,
- Bastos, L.R.D. and Castro, J.F.B. (2003): "Integrating Organizational Requirements and Socio-Intentional Architectural Styles". Proceedings of the Second International Workshop From Software Requirements to Architectures (STRAW03), 2003. p.114 – 121. In 25th International Conference on Software Engineering 2003 (ICSE'03). Portland, May.
- Bastos, L.R.D. and Castro, J.F.B. (2003): "Integration between Organizational Requirements and Architecture". WER03 - VI Workshop em Engenharia de Requisitos. Piracicaba, Brasil, November.
- Bastos, L.R.D. and Castro, J.F.B. (2004): "Enhancing Requirements to derive Multi-Agent Architectures". WER04 - VII Workshop em Engenharia de Requisitos. Tandil, Argentina, December.
- Bastos, L.R.D. and Castro, J.F.B. (2004): "Systematic Integration between Requirements and Architecture". Third International Workshop on Software Engineering for Large-Scale Multi-Agent Systems SELMAS'04. Edinburgh, Scotland, May. LNCS 3390: "Software Engineering for Multi-Agent Systems III", pp. 85-103.R.Choren et all. (Eds.), Springer-Verlag, 2005.
- Bastos, L.R.D. and Castro, J.F.B. (2005): "From Requirements to Multi-Agent Architecture based on Organisational Concepts". 4th International Workshop on Software Engineering for Large-Scale Multi-Agent Systems SELMAS'05. 15 - 16 May 2005. St. Louis, Missouri - USA In Conjunction with ICSE 2005.

- Bastos, L. R.D. e Castro, J. F.B. (2005): "Organizational Model to Derive Multi-Agent Architecture from Requirements". Proceedings of the 17th Conference on Advanced Information Systems Engineering - CAISE'05 Forum - Porto, Portugal, June 13-17, 2005.
- Biddle B. J. and Thomas, E. J. (1979): "Role Theory: Concepts and Research". New York: Robert E. Krieger Publishing Company.
- Boehm B., Egyed A., Kwan J., Port D., Shah A., Madachy R. (1998): "Using the WinWin Spiral Model: A Case Study". IEEE Computer, 7:33-44.
- Boehm B., Grünbacher P., Briggs B. (2001): "Developing GroupWare for Requirements Negotiation: Lessons Learned". IEEE Software, May/June, vol 18(3) - pp. 46-55.
- Boehm, B. W. (1989): "Software Risk Management". IEEE Computer Society Press. Washington, pp.1-469.
- Bresciani, P., Giorgini, P., Giunchiglia, F., Mylopoulos, J. and Perini, A. (2004): "TROPOS: An Agent-Oriented Software Development Methodology". In Journal of Autonomous Agents and Multi-Agent Systems, Kluwer Academic Publishers Volume 8, Issue 3, Pages 203 - 236, May.
- Bresciani, P., Perini, A., Giorgini, P., Giunchiglia, F. and Mylopoulos, J. (2001): "Modeling early requirements in Tropos: a transformation based approach". Second International Workshop on Agent-Oriented Software Engineering (AOSE-2001), Montreal, Canada - May 29th.
- Bubenko, J. (1983): "On Concepts and Strategies for Requirements and Information Analysis". In Information Modelling. Chartwell-Bratt.
- Castelfranchi, C. (2000): "Engineering Social Order". In: Omicini, A., Tolksdorf, R., Zambonelli, F. (Eds.): Engineering Societies in the Agents World, LNAI 1972, Springer-Verlag, pp. 1 - 19.
- Castro, J., Kolp, M. and Mylopoulos, J. (2002): "Towards Requirements Driven Information Systems Engineering: The Tropos Project". In Information Systems, Vol. 27. Elsevier, Amsterdam, The Netherlands 365-389.
- Chung, L. (1998): "Architecting Quality Using Quality Requirements". Proc. KUST, Oct. 22-24, Vienna, Virginia.
- Chung, L. and Yu, E (1998): "Achieving System-Wide Architecture Qualities". OMG-DARPA-MCC Workshop on Compositional Software Architectures, January 6-8, Monterey, California.
- Chung, L., B. Nixon and Yu, E. (1995): "Using Non-Functional Requirements to Systematically Select Among Alternatives in Architectural Design"; Proc., 1st International Workshop on Architectures for Software Systems, Seattle, April 24-28, pp. 31-43.
- Chung, L., Nixon, B. A., Yu, E. and Mylopoulos, J. (2000): "Non-Functional Requirements in Software Engineering". Kluwer Publishing.
- Cockburn, A. (1995): "Structuring use cases with goals". 7691 Dell Rd, Salt Lake City, UT 84121, Human and Technology.

- Conallen, J. (2000): "Building Web Applications with UML". Addison-Wesley, Reading, MA.
- Crook, R., Ince, D. and Nuseibeh, B. (2002): "Towards an Analytical Role Modeling Framework for Security Requirements". In proceedings of Eighth International Workshop on Requirements Engineering: Foundation for Software Quality REFSQ'02. September, Essen, Germany.
- Cysneiro, L. M. and Leite, J.C.P.: "Integrating Non Functional Requirements into Data Modeling". In Proc. of the fourth International Symposium on Requirements Engineering – ER'99, pp.162-171, June 1999, Limerick, Ireland. IEEE Computer Society Press.
- Dardenne, A, van Lamsweerde, Fickas, S. (1993): "Goal Directed Requirements Acquisition". Science of Computer Programming, April - vol. 20, pp. 3-50.
- Davidsson, P. (2001): "Categories of Artificial Societies ". In *Engineering Societies in the Agents World II*, Springer Verlag LNCS series, Vol. 2203, 2001.
- DeLoach, S. A. (2002): "Modeling Organizational Rules in the Multi-agent Systems Engineering Methodology". R. Cohen and B. Spencer (Eds.): AI 2002, LNAI 2338, pp. 1-15, 2002. Springer-Verlag Berlin, Heidelberg.
- Dignum, V., Dignum, F. (2001): Modeling Agent Societies: Coordination Frameworks and Institutions. In: Brazdil, P, Jorge, A.: Progress in Artificial Intelligence. LNAI 2258, Springer-Verlag, 191 - 204.
- Dignum, V., Meyer, J-J., Weigand, H. (2002): Towards an Organizational Model for Agent Societies Using Contracts. In: Proc. of AAMAS, the 1st International Joint Conference in Autonomous Agents and Multi-Agent Systems, Bologna, Italy, July.
- Do, T. T., Faulkner, S. and Kolp, M. (2003): "Organizational Multi-Agent Architectures for Information Systems". In Proceedings of the 5th International Conference on Enterprise Information Systems (ICEIS 2003), Angers, France, April.
- Dubois, E., P. Heymans. (1998): "Scenario Based Techniques for supporting the elaboration and the validation of Formal Requirements". RE Journal- Vol 3, Issue 3-4, 202-218.
- Durfee, E.H., Lesser, V.R. and Corkill, D.D. (1989): "Trends in Cooperative Distributed Problem Solving". In: IEEE Transactions on Knowledge and Data Engineering, March, KDE-1 (1), pages 63-83.
- Esteva, M., Rodriguez, J., Sierra, C., Garcia, P., Arcos J. (2001): "On the formal specifications of electronic institutions", In: Dignum F., Sierra C. (Eds.): Agent-mediated Electronic commerce (The European AgentLink Perspective), LNAI 1991, Springer, pp. 126-147.
- Far B. H., Hajji H., Saniepour S., Soueina S.O. (2000): "Formalization of Organizational Intelligence for Multiagent System Design". IEICE Transactions, Vol. E83-D, no. 4, 2000.
- Fayol, H. (1916): "Administration, industrielle et générale". Extrait du Bulletin de la Société de l'Industrie Minérale, 3e livraison. Quarantième Mille. Paris: Dunod,
- Feather M. and Menzies T. (2002): "Converging on the Optimal Attainment of Requirements". RE02 IEEE

Joint Conference on Requirements Engineering, Essen, Germany.

Ferber, J. and Gutknecht O. (1998): "A meta-model for the analysis and design of organizations in multiagents systems". In Demazeau, Y., editor, ICMAS' 98, pages 128–135, Paris.

Ferber, J., Gutknecht, O. and Michel, F. (2003): "From Agents to Organizations: An Organizational View of Multi-agent Systems". In Proceedings of The Fourth International Workshop on Agent-Oriented Software Engineering (AOSE-2003). Melbourne, Australia.

Fincham, R. and Rhodes, P. (1999): "Principles of Organizational Behaviour". Third Edition, Oxford University Press, August.

Fox, M., Barbuceanu, M., Gruninger, M. and Lin, J. (1996): "An Organization Ontology for Enterprise Modelling". Simulating Organizations: Computational Models of Institutions and Groups, M. Pritula, K. Carley & L. Gasser (Eds), Menlo Park CA: AAAI/MIT Press, pp. 131-152.

Freeman, L.C. (1979): "Centrality in Social Networks I: Conceptual Clarification". Social Networks, 1, 215-239.

Fuxman, A., Giorgini, P., Kolp, M. and Mylopoulos, J. (2001): "Information Systems as Social Structures". In Proceedings of the Second International Conference on Formal Ontologies for Information Systems (FOIS-2001), Ogunquit, USA, October 17-19.

Fuxman, A., Liu, L., Pistore, M., Roveri, M. and Mylopoulos, J. (2003): "Specifying and Analyzing Early Requirements: Some Experimental Results". In Proceedings of the 11th IEEE International Symposium on Requirements Engineering (RE'03), Monterey Bay, California, USA. September 8-12.

Fuxman, A., Pistore, M., Mylopoulos, J., and Traverso P. (2001): "Model Checking Early Requirements Specifications in Tropos". In Proceedings of the 5th IEEE International Symposium on Requirements Engineering (RE'01), Toronto, Canada, August 27-31.

Garlan, D. (2000): "Software Architecture: a Roadmap". In proceedings of the 22nd International Conference on Software Engineering, ICSE'00. IEEE Computer Society Press. Limerick, Ireland. Jun.

Garlan, D. and Shaw, M. (1993): "An introduction to software architecture". Ambriola & Tortola (eds.), Advances in Software Engineering & Knowledge Engineering, vol. II, World Scientific Pub Co., Singapore, pp. 1-39.

Garlan, D., and Perry, D. (1995): "Introduction to the Special Issue on Software Architecture". IEEE Transactions on Software Engineering 21, 4 (April).

Gazendam, H.W.M. and Jorna, R.J. (1993): "Theories about architecture and performance of multi-agent systems". III European Congress of Psychology, Tampere, Finland, July 4-9.

Giorgini, P., Kolp, M., and Mylopoulos, J. (2002): "Multi-Agent and Software Architecture: A Comparative Case Study". Proceedings of the 1st International Conference on Autonomous Agent and Multi Agent Systems (AAMAS'02), Bologna, Italy, July.

- Golub G. and Loan C. van (1996): "Matrix computations". Third edition, The Johns Hopkins University Press, London. http://www.fon.hum.uva.nl/praat/manual/Golub___van_Loan__1996_.html.
- Gross, D. and Yu, E. (2001): "From Non-Functional Requirements to Design through Patterns". Springer-Verlag London Limited, Requirements Eng (2001) 6:18–36.
- Grünbacher, P., Egyed, A., Medvidovic, N. (2004): "Reconciling Software Requirements and Architectures with Intermediate Models". *Journal for Software and Systems Modeling (SoSyM)*.
- Hanneman, R. (2005): "Introduction to Social Network Methods". Department of Sociology. University of California, Riverside. On-line textbook. Last accessed in January. <http://faculty.ucr.edu/~hanneman/SOC157/TEXT/TextIndex.html>.
- IEEE Std 1471-2000. (2000): "Recommended Practice for Architectural Description of Software Intensive Systems".
- IEEE Std 830-1998: (1998): "IEEE Guide to Software Requirements Specification". Software Engineering Standards Committee of the IEEE Computer Society Approved 25, June. (Revision of IEEE Std 830-1993). The Institute of Electrical and Electronics Engineers, New York, EUA.
- IEEE. (1999): "IEEE Software Engineering Standards Collection". Computer Society Press, Edition. Vol 3.
- ISO/IEC 9126 (1991) Standard for Information Technology, Software Product Evaluation - Quality Characteristics and Guidelines for their use, Geneva.
- ISO/IEC TR 15504 (1999) - SPICE Project: <http://www-sqi.cit.gu.edu.au/spice/>.
- Jennings, N.R., Sycara, K. and Wooldridge, M. (1998): "A Roadmap of Agent Research and Development". In: Autonomous Agents and Multi-Agent Systems Journal, N.R. Jennings, K. Sycara and M. Georgeff (Eds.), Kluwer Academic Publishers, Boston, Volume 1, Issue 1, pages 7-38.
- Kavakli, E. (1999): "Goal-Driven Requirements Engineering: Modelling and Guidance". PHD Thesis. Institute of Science and Technology, University of Manchester.
- Kavakli, E. (2004): "Modeling organizational goals: Analysis of current methods". Proceedings of the 2004 ACM Symposium on Applied Computing, Nicosia, CY, March 2004, ISBN:1-58113-812-1, pp. 1339 - 1343.
- Kazman, R.: "Software Architecture". In Handbook of Software Engineering and Knowledge Engineering, S-K Chang (ed.). World Scientific Publishing, 2001
- Kolp, M., Giorgin, P., and Mylopoulos, J.: "A goal-based organizational perspective on multi-agents architectures". In Proceedings of the Eighth International Workshop on Intelligent Agents: Agent Theories, Architectures, and Languages (ATAL '01), Seattle, USA, August 2001.
- Kolp, M., Giorgini, P. and Mylopoulos, J.: "Organizational Patterns for Early Requirements Analysis". In Proceedings of the 15th International Conference on Advanced Information Systems Engineering (CAiSE'03). Velden, Austria. June 2003.

- Lamsweerde, A. van.: "Formal Specification: a Roadmap". In the Future of Software Engineering, A. Finkelstein (ed.), ACM Press, 2000.
- Lamsweerde, A. van.: "Goal-Oriented Requirements Engineering: A Guided Tour". Proceedings RE'01, 5th International Symposium on Requirements Engineering. Toronto, Canadá. August 2001, 249-263.
- Lamsweerde, A. van.: "Requirements Engineering in the Year 00: A Research Perspective". 22nd Proceedings of International Conference on Software Engineering, Limerick, Ireland. ACM Press, 2000, pp. 5-19.
- Leite, J.C.S.P, Rossi, G., Balaguer, F., Maiorana, V.: "Enhancing a requirements baseline with scenarios". In Proceedings of the Third IEEE International Symposium on RE97-Requirements Engineering -, pages 44-53. IEEE Computer Society Press, January, 1997
- Letier, E. and Lamsweerde, A. van. (2002): "Agent-based Tactics for Goal-Oriented Requirements Elaboration", Proceedings of 24th International Conference on Software Engineering (ICSE'02), ACM Press, May 2002.
- Lopez, F. L, Luck, M.: "Modelling Norms for Autonomous Agents". Fourth Mexican International Conference on Computer Science. P. 238. IEEE. September 08 - 12, 2003. Tlaxcala, Mexico
- Loucopoulos, P. and Karakostas, V.: "System Requirements Engineering". McGraw-Hill Book Company.1995.
- Lupu, E.C. and Sloman, M. (1997): "Towards a role based framework for distributed systems management". Journal of Network and Systems Management, vol. 5, no. 1, Plenum Press, 1997.
- Lupu, E.C.(1998): "Role based framework for distributed systems management". PhD Thesis, Faculty of Engineering of the University of London, 1998.
- Mao, X. and Yu, E. (2004): "Organisational and Social Concepts in Agent Oriented Software Engineering". Agent-Oriented Software Engineering (AOSE) IV, James Odell, P. Giorgini, Jörg Müller, eds, Lecture Notes on Computer Science volume, Springer, Berlin.
- MESSAGE (2004): Methodology for Engineering Systems of Software Agents. www.eurescom.de/~pub-deliverables/P900-series/P907/TI1/. Last accessed in June 2004.
- Mintzberg, H. (1992): "Structure in Fives: Designing effective organisations". Prentice Hall.
- Morgan, G. (1996): "Images of organization". Beverly Hills: SAGE.
- Mylopoulos J., Borgida A., Jarke M., and Koubarakis M. (1990): "Telos: Representing Knowledge About Information Systems". ACM Transactions on Information Systems, 8(4): 325-362.
- Mylopoulos, J. and Castro. J. (2000): "Tropos: A Framework for Requirements-Driven Software Development". In J. Brinkkemper and A. Solvberg (eds.), Information Systems Engineering: State of the Art and Research Themes, Lecture Notes in Computer Science, Springer-Verlag, p. 261-273, June.
- Mylopoulos, J., Kolp, M. and Castro, J. (2001): "UML for Agent-Oriented Software Development: the Tropos Proposal". In Proceedings of the Fourth International Conference on the Unified Modeling Language

- (<<UML>> 2001), Toronto, Canada, October.
- Newell, A. and Simon, H. (1963): "GPS: A program that simulates human thought". In *Computers and Thought*, ed. Feigenbaum and Feldman. McGraw-Hill, New York..
- Object Management Group (OMG). www.omg.org . Página visitada em 2003.
- Odell, J., Parunak, H.V.D., and Fleischer, M. (2003): "The Role of Roles in Designing Effective Agent Organizations". *Software Engineering for Large-Scale Multi- Agent Systems*, Alessandro Garcia et al, LNCS, Springer.
- OMG Agent Working Group <http://www.objs.com/isig/agents.html>. 2004.
- Partsakoulakis, I., and Vouros, G (2004): "Roles in MAS: Managing the Complexity of Tasks and Environments". *Multi-Agent Systems: An application Science*, T. Wagner (eds.), Kluwer Academic
- Parunak, H. V. D., Odell, J. (2002): Representing Social Structures in UML. In: Wooldridge, M., Weiss, G., Ciancarini P. (Eds.): *Agent-Oriented Software Engineering II*, LNCS 2222, Springer.
- Pearson Correlation Coefficient (2005): <http://www.cmh.edu/stats/definitions/correlation.htm>. Last accessed in February.
- Pearson Correlation Coefficient in Microsoft Excel (2005). http://www.bized.ac.uk/timeweb/crunching/crunch_relate_illus.htm#excelexp. Last accessed in June.
- Perry, D. E. and Wolf, A. L. (1992): "Foundations for the Study of Software Architecture", *ACM SIGSOFT Software Engineering Notes*, 17(4), pp. 40-52.
- Pohl, K. (1996) *Process-Centered Requirements Engineering*, Research Studies Press Ltd., Taunton, Somerset, England, 1996.
- Rolland, C., Souveyet, C., Achour, C. B. (1998): "Guiding Goal Modeling Using Scenarios". *IEEE Transactions on Software Engineering*, Vol 24 (12), Special Issue on Scenario Management. December.
- Santander, V. F. A. and Castro, J. F. B. (2002a): "Deriving Use Cases from Organizational Modeling". RE02 - *IEEE Joint Conference on Requirements Engineering*. Essen, Germany.
- Santander, V. F. A. and Castro, J. F. B. (2002b): "Integrating Use Cases and Organizational Modeling". SBES02 - *XVI Brazilian Symposium on Software Engineering*, Gramado. October.
- Shaw, M. (2001): "The Coming-of-Age of Software Architecture Research". *Proceedings of the 23rd International Conference on Software Engineering (ICSE 2001)*. Toronto, Canada. May.
- Shaw, M. and Clements, P. (1997): "A field guide to boxology: Preliminary classification of architectural styles for software systems". In *Proceedings of the Twenty-First Annual International Computer Software and Applications Conference (COMPSAC'97)*, Washington, D.C., Aug. 1997, pp. 6-13
- Shaw, M., and Garlan, D. (1996): "Software Architecture: Perspectives on an Emerging Discipline", New Jersey.

- Silva, C. T. L. L., Tedesco, P. C., Castro, J. B. F., Pinto, R. C. C. (2004): "Comparing Agent-Oriented Methodologies Using NFR Approach". Third International Workshop on Software Engineering for Large-Scale Multi-Agent Systems SELMAS'04. Edinburgh, Scotland, May.
- Stockburger, D. W.: "Introductory Statistics: Concepts, Models, and Applications". WWW Version 1.0 First Published 1996. Revised 1998. <http://www.psychstat.smsu.edu/introbook/sbk00.htm>.
- Sturm, A., Shehory, O. (2003): A Framework for Evaluating Agent-Oriented Methodologies. Fifth International Bi-Conference Workshop on AGENT-ORIENTED INFORMATION SYSTEMS (AOIS-2003) 17 July, Melbourne, Australia, at AAMAS'03.
- Sudeikat, J., Braubach, L., Pokahr, A., and Lamersdorf, W.: "Evaluation of Agent-Oriented Software Methodologies: Examination of the Gap Between Modeling and Platform". Agent-Oriented Software Engineering (AOSE) IV, James Odell, P. Giorgini, Jörg Müller, eds., Lecture Notes on Computer Science volume (forthcoming), Springer, Berlin, 2004.
- The Foundation for Intelligent Physical Agents (2004). FIPA Agent Management Specification. <http://www.fipa.org>.
- Trzebiatowski, G. L. and Münch, I.: "The Role Concept for Agents in Multi-Agent Systems". Modelling Artificial Societies and Hybrid Organizations. Workshop at KI2001, the Joint German/Austrian Conference on Artificial Intelligence. Vienna, September 19-21, 2001.
- Wagner, G. (2000): Agent-Oriented Analysis and Design of Organizational Information Systems. Proceedings of the 4th IEEE International Baltic Workshop on Databases and Information Systems, Vilnius, Lithuania, May.
- Wasserman, S. and Faust, K. (1994): "Social Network Analysis". Cambridge: Cambridge University Press. <http://lrs.ed.uiuc.edu/tse-portal/analysis/social-network-analysis/>
- Weber, M. (1976): "Basic Concepts in Sociology". Translated by H. P. Secher. Kensington Publishing Corp, 123 pages. Published: May 1976.
- Weiss, G. (2000): "Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence". 2nd Edition. MIT Press.
- Wieringa, R., Dubois, E., Huyts, S. (1997): Integrating Semi-Formal and Formal Requirements, In Proceedings of the 7 th Int. Conference on Advanced Info. Systems Engineering – CaiSE'97, Barcelona, Spain.
- Wood, M.F. and DeLoach, S.A. (2001): "An overview of the multiagent systems engineering methodology". First international workshop, AOSE 2000 on Agent-oriented software engineering, p.207-221, January, Limerick, Ireland.
- Wooldridge, M. (2002): "An Introduction to Multiagent Systems". John Wiley & Sons (Chichester, England). Published in February.
- Wooldridge, M. and Ciancarini, P. (2001): "Agent-Oriented Software Engineering: The State of the Art". In

Agent-Oriented Software Engineering. Springer-Verlag Lecture Notes in AI Volume 1957.

- Wooldridge, M., and Jennings, N. R. (1995): "Intelligent agents: Theory and practice". Knowledge Engineering Review 10(2):115–152.
- Wooldridge, M., Jennings, N.R. and Kinny D. (2000): "The Gaia Methodology for Agent-Oriented Analysis and Design". Autonomous Agents and Multi-Agent System, 3, 285-312. Kluwer Academic Publishers, Netherlands.
- Wooldridge, M.J. and Jennings, N.R. (1998): "Pitfalls of Agent-Oriented Development". In: Proceedings of the Second International Conference on Autonomous Agents (Agents '98), K. P. Sycara and M. Wooldridge (Eds.), ACM Press, May 1998.
- Yoshino M.Y. and Rangan U. S. (1995): "Strategic alliances: an entrepreneurial approach to globalisation". Boston, Mass., Harvard Business School Press, 1995.
- Yu, E. (1993): "An organization modeling framework for information systems requirements engineering". In Proc. Of the 3rd Workshop on Information Technologies and Systems WITS'93, Orlando FL, December 4-5.
- Yu, E. (1995): "Modeling Strategic Relationships for Process Reengineering". Ph.D. thesis, Department of Computer Science, University of Toronto, Canada.
- Yu, E. (2001): "Agent Orientation as a Modeling Paradigm". Wirtschaftsinformatik. 43(2), April- pp.123-132.
- Yu, E. and Mylopoulos, J. (1998): "Why Goal-Oriented Requirements Engineering". In Proceedings of the 4th International Workshop on Requirements Engineering: Foundations of Software Quality- REFSQ'98, pp. 15-22. Pisa, Italy. June.
- Yu, E., and Mylopoulos, J. (1997): "Modeling Organizational Issues for Enterprise Integration". ICEIMT'97 - International Conference on Enterprise Integration and Modeling Technology. Turin, Italy. October.
- Yu, Eric and Mylopoulos, John. (1994): "Understanding ``Why" in Software Process Modeling, Analysis, and Design". Proceedings of 16th International Conference on Software Engineering. May 16-21, Sorrento, Italy, pp. 159-168.
- Zack, M. H. (1999): "Managing Codified Knowledge". MIT Sloan Management Review. Summer 1999, Vol. 40, No. 4, pp. 45–58.
- Zack, M. H. (2000): "Researching Organizational system using social network analysis". Proceedings of the 33rd Hawaii International Conference on System Sciences. Maui, Hawaii, January (IEEE 2000).
- Zambonelli, F. (2002): Abstractions and Infrastructures for the Design and Development of Mobile Agent Organizations. In: Wooldridge, M., Weiss, G., Ciancarini P. (Eds.): Agent-Oriented Software Engineering II, LNCS 2222, Springer- Verlag, pp. 245 – 262.
- Zambonelli, F., Jennings, N., Wooldridge, M. (2001): "Organisational Abstractions for the Analysis and Design of Multi-Agent Systems". In: Ciancarini P., Wooldridge, M. (eds.): Agent-Oriented Software

Engineering, LNCS 1957, Springer- Verlag, pp. 235 – 251.

Zambonelli, F., Jennings, N.R., and Wooldridge, M. (2003): “Developing Multi-agent Systems: The Gaia Methodology”. *ACM Transactions on Software Engineering and Methodology*, vol. 12, no. 3, pp.417 -- 470, 2003

Zave, P. (1997) *Classification of Research Efforts in Requirements Engineering*, *ACM Computing Surveys*, Vol. 29, No. 4, 1997, pp. 315-321.

Zoethout, K., Jager, W. and Molleman, E. (2002): “ A Theoretical Framework for Describing Self-Organizing Social Processes of Task Allocation”. Special Workshop "Multidisciplinary Aspects of Learning", European Society for the Study of Cognitive Systems, pages 17-19. January, Clichy (Paris), France.