Universidade Federal de Pernambuco
Graduação em Engenharia da Computação
Centro de Informática

Graduation Work

A STUDY OF THE ***SIRA*** PROCESS

(***S***ystematic ***I***ntegration between ***R***equirements and ***A***rchitecture)

Turah Xavier de Almeida

(txa@cin.ufpe.br)

Supervisor

Jaelson Freire Brelaz De Castro

(jbc@cin.ufpe.br)

Recife, March 2007.

ABSTRACT

Requirements Engineering and Software Architecture are recognized within the software engineering community as important areas of research and practice. Even though requirements and architecture are clearly related, the transition between them remains as a challenging problem since there is still a lack of systematic guidelines for this transition. The SIRA Process focuses on a systematic way to assist the transition from requirements to architecture. The aim of this work is to evaluate the SIRA process in order to determine if it is possible to develop a CASE tool able to support the SIRA process automatically.

ABSTRACT

Engenharia de Requisitos e Arquitetura de Software são reconhecidas pela comunidade de Engenharia de Software como importantes áreas de pesquisa e prática. Apesar de estarem claramente relacionadas, a transição entre requisitos e arquitetura continua sendo um problema desafiador devido à falta de guias que ajudem essa transição. O processo SIRA consiste numa forma sistemática de ajudar na transição de requisitos para arquitetura. O objetivo desse trabalho é avaliar o processo SIRA e determinar se é possível o desenvolvimento de uma ferramenta CASE para suportar o processo de forma sistemática.

ACKNOWLEDGEMENTS

Firstly, I would like to thank my supervisor, Professor Jaelson Castro, for taking me into his group and for all the support and guidance I have been given.

Also, I would like to thank Márcia Lucena for her guidance, enriching discussions over this work and for being available to help me whenever I needed.

I want to thank all of my computer engineering classmates, especially my friends Junior, Adson, Diego, Adriano, Boppito, Paty, Duds and my best friend Hélice for the all the support and friendship.

And my loving family, specially my sister Thalita, vovós, tia Sandra, tio Ricardo, gordo, bibi and kessy.

Finally, I want to thank my parents for the unconditional love and support and for being even better parents than I could have ever asked for.

My sincere thanks to everyone that somehow contributed to this work and inspired me during my Computer Engineering course.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. Introduction

Both Requirements Engineering and Software Architecture are recognized within the software engineering community as important areas of research and practice. During software development, there is a large conceptual distance between what to do (requirements) and how to do it (architecture, design and code). Even though requirements and architecture are clearly related, it is surprising how little research has been done so far to systematically derive architecture based on requirements. Currently, this process is a difficult task mainly based on intuition and experience of architects and designers. The transition between requirements and architecture remains as a challenging problem since there is still a lack of systematic guidelines to support this transition.

A lot of effort has been made to try and solve this problem. The following processes can be highlighted:

- Architecture Description Languages (ADL): offer means of representing Software Architectures but do not adequately support the transition from requirements to architecture [6].
- Goal-Based Approach: performs a transition from requirements to architecture to meet functional and non-functional requirements but the qualitative reasoning in the refinement process should be more formal to allow extended tool support. Also, the relation between global architecture decisions at early stages of the process and the final refinements to meet all non-functional requirements should be more explicit [6].
- Problem Frames: allow the classification of software problems and the decomposition of a large problem into sub-problems that can then be solved and combined into a solution of the original problem. It is not clear though if the notation covers all aspects for creating proper architectures [6].

- Use Case Maps (UCMs): present scenarios as visual behavior structures manipulated and reused as architectural entities but the lack of well-defined semantics and the large human input required may be a disadvantage [6].
- Rule-Based Decision Making: supports automated reasoning for eliciting architectural decisions based on requirements but significant human interaction is required to perform the transition from requirements to architecture [6].
- Architecting Requirements: by structuring and organizing requirements we may be able to identify components for a prospective architecture already during the requirements engineering phase before design and implementation. It is not clear what kind of architectural representation is generated as output of the process and what aspects of an architecture are addressed [6].
- Object-Oriented Transition: has the idea of transforming the object-oriented output of the requirements engineering phase into an object-oriented architecture/design. Still, it does not provide a complete solution for mapping requirements into architecture [6].

It is in this context that the SIRA Process comes into place: the SIRA Process focuses on a systematic way to assist the transition from requirements to architecture. It is provided by the SIRA Framework that was presented in [3] as a framework able to help reduce the gap among Multiagent Systems (MAS) requirement models and architectural models. This framework describes a software system from the perspective of an organization and is located in the context of the Tropos Project [4].

The aim of this work is to evaluate the SIRA process in order to determine if it is possible to develop a CASE tool able to support the SIRA process automatically. A case study will be applied in each activity of the process to try to resolve its conflicting decisions. Mechanisms will be created to help provide resolutions and explanations when multiple conflicting decisions are made for the same part of the SIRA model, such as goal refinement, cluster analysis and correlation analysis [3]. Possible failures in the process will be identified and corrected (when possible) and suggestions for improvement will be made.

This work is organized in five chapters:

- Chapter 2 gives an overview of the basic theories related to this work;
- Chapter 3 briefly presents the SIRA Process;
- Chapter 4 evaluates the SIRA Process by applying the Conference Management example to each activity of the process;
- Chapter 5 concludes this work and suggests future research direction considering its limitations.

2. Background

During software development, a requirements engineering process is a crucial factor for the success and quality of the final products. This process involves activities such as requirements elicitation, requirements analysis and negotiation, requirements specification, requirements validation and requirements management.

Good software architecture is also a major factor for successful products. Software architectures are important because they represent an abstraction for understanding the structure of a system [1]. There are important reasons why we should care about architectures: They work as a mutual understanding and negotiation among stakeholders, they represent early design decisions and allow further analysis of the system, and they provide developers with an implementation-independent, reusable and transferable abstraction of future system [2].

The relationship between requirements and architecture of a system to be is neither clear nor obvious: requirements are often elicited informally while entities in software architecture are often specified in a kind of formal way, stakeholders may have conflicting goals and expectations, non-functional requirements are hard to be mapped to an architectural entity, etc.

The SIRA Framework has made a first step in the direction of analyzing requirements of multi-agent systems (MAS) as a software architecture style based on organizational concepts. A multi-agent system corresponds to an organization where roles are members of a group with a goal to be fulfilled. Groups, goals, members, roles and interactions compose an organization. Such an organizational model provides help to the integration between requirements and architecture [3].

2.1. Requirements Engineering

The aim of system requirements is to define what services the system should provide and under what circumstances it should operate. It is impossible to define a standard way of describing requirements as it depends on who is writing or reading them as well as what is the application domain of the system and what are the general practices of the organization.

During computer-based systems development, some difficulties with the system requirements may arise. The following requirements problems can be highlighted:

- The requirements don’t reflect the costumer’s real needs;
- The requirements are inconsistent and / or incomplete;
- It is expensive to make changes to requirements after an agreement has been made with the costumer;
- Some misunderstandings can arise between costumers, system requirements developers and software engineers responsible for the development of the system.

The problems with writing requirements are universal and there will never be a complete solution to that. However, good requirements engineering practice can reduce the number of problems and minimize their impact on the final system [5].

There is no common definition of requirements engineering but there is an agreement that it is the process of determining what the stakeholders want from the product before it starts to be built. In other words, it is the process of discovering and documenting a set of requirements for a system. During this process, techniques should be applied to make sure that these requirements are complete, consistent, etc.

As it has been mentioned before, a requirements engineering process is a crucial factor for the success and quality of the final products and it involves activities such as requirements elicitation, requirements analysis and negotiation, requirements specification, requirements validation and requirements management.

Even though traditional approaches to requirements engineering focus on the functionalities of the system to be, in the last few years, researches in the field have evolved from software systems to a broader perspective that incorporate aspects of organizational environment [10].

In general, requirements can be classified into three different, but strongly related basic classes: functional, non-functional and organizational [10]. Informally, functional requirements describe *what* the system must do., non-functional requirements describe *how well* these functional requirements will be satisfied in the system and they must be understood within an organizational context [3].

2.2. Software Architecture

As it has been mentioned before, good software architecture is a major factor for successful products. Software architectures are important because they represent an abstraction for understanding the structure of a system [1].

There are important reasons why we should care about architectures: They work as a mutual understanding and negotiation among stakeholders, they represent early design decisions and allow further analysis of the system, and they provide developers with an implementation-independent, reusable and transferable abstraction of future system [2].

There is no agreed definition of software architecture either, but the most cited definitions emphasis software architecture as a description of a system as a sum of smaller parts, and how these parts relate to and cooperate with each other to perform the system [1][2][11].

The architecture of a system should be in conformity with its non-functional requirements, such as performance, safety, flexibility, etc [3]. In the context of the SIRA Process, it is considered that the architecture of a software system defines the system in terms of components and of interactions among these components [3].

Components are architectural elements that represent computational elements and data store of a system. Examples of components include clients, servers, filters and databases. Connectors are architectural elements that represent the interactions among components [12].

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. In software architectures, styles are required to implement higher-level components in terms of lower-level ones [12].

Some unconventional architectural styles have emerged in the literature recently. These architectural styles are based on organizational theory. The Tropos Project has a defined catalogue of organizational styles and will be overviewed further on.

In the context of the SIRA process, Multi Agent System is a system consisting of components identified as agents that communicate and cooperate on the basis of organizational architectures [3]. Some basic concepts of Multi Agent Systems (MAS) will be presented next.

2.3. The Multiagent Systems

There are many different definitions to the term Multiagent systems (MAS). As defined in [14], MAS is a loosely coupled network of problem solver entities that works together to find answers to problems that are beyond the individual capabilities or knowledge of each entity.

A Multiagent System is concerned with breaking down the global problem into sub-problems and assigning these sub-problems to agents with the best abilities to solve them. This assignment is done considering the best (most appropriate and efficient) way to solve the global problem.

The study of MAS focuses on systems in which different agents interact with each other. These agents are considered to be autonomous entities, such as software programs or robots. Their interactions can be either cooperative or selfish. Meaning the agents can share a common goal or they can pursue their own interests [13].

The following characteristics of MAS can be highlighted [13]:

- Each agent has incomplete information or capabilities for solving the problem and, thus, has a limited viewpoint;
- There is no system global control;
- Data are decentralized;
- Computation is asynchronous.

The most important reason to use MAS when designing a system is that some domains require it. In particular, if there are different people or organizations with different (possibly conflicting) goals, then a Multiagent system is needed to handle their interactions. Even if each organization wants to model its internal affairs with a single system, the organizations will not give authority to any single person to build a system that represents them all: the different organizations will need their own systems that reflect their capabilities and priorities [15].

In the context of the SIRA process, MAS consist of a group of agents that can take specific roles within an organizational structure. An agent is an active component that interacts with its environment and has the ability to play one or more roles. Agents are represented as roles and are persistent and have relatively long-lived goals representing the functional and non-functional aspects of what they are doing [3].

2.4. The TROPOS Project

Existing software development methodologies have traditionally been inspired by programming concepts, not organizational ones, leading to a semantic gap between the software system and its operational environment. Tropos is an information system development methodology that has been proposed to reduce this gap [4].

The Tropos methodology is requirements-driven in the sense that it is based on concepts used during early requirements analysis. Tropos spans four phases [4]:

- Early requirements: concerned with the understanding of a problem by studying an organizational setting. The output of this phase is an organizational model which includes relevant roles, their respective goals and their inter-dependencies.
- Late requirements: where the system-to-be is described within its operational environment, along with relevant functions and qualities.
- Architectural design: where the system's global architectural is defined in terms of sub-systems, interconnected through data, control and other dependencies.
- Detailed design: where the behavior of each architectural component is defined in further detail.

Tropos adopts the concepts offered by i* modeling framework [16] such as roles (agents, positions or roles) and the social dependencies among roles (goal, softgoal, task and resource). The i* framework consists of two models:

- The Strategic Dependency Model (SD): describes the network of dependency relationships among roles. It consists of a set of nodes and links. Each node represents a role and each link between two roles indicates that one role (depender) depends on the other role (dependee) in order to achieve some goal. The *dependum* is the type of the dependency and describes the nature of the agreement. There are four types of dependencies [4]:

  - Goal-dependency: is used to represent delegation of responsibility for fulfilling a goal;
  - Task-dependency: is used in situations where the dependee is required to perform a given activity;
  - Softgoal-dependency: are similar to goal dependencies but their fulfillment cannot be defined precisely;
  - Resource-dependency: requires the dependee to provide a resource to the depender.

- The Strategic Rationale Model (SR): determines through a means-ends analysis how the goals and softgoals determined on the SD model can actually be fulfilled through the contributions of other roles. It is a graph with four types of nodes (goal, task, resource, softgoal) and two types of links (means-ends links and task decomposition links) [4].

Tropos also defined organizational architectural styles for multiagent systems. They are organized in an architectural catalogue based on concepts and design alternative from research on organization management. These organizational styles help guide the architectural design of MAS. They are called: *pyramid*, *joint venture*, *structure-in-5*, *takeover*, *arm's length*, *vertical integration*, *co-optation* and *bidding*.

These organizational styles have been evaluated and compared using software quality attributes identified for architectures involving coordinated autonomous components such as *predictability*, *security*, *adaptability*, *coordinability*, *cooperativity*, *availability*, *integrity*, *modularity* or *aggregability*. The first task during the architectural design phase in Tropos is to select among alternative architectural styles using as criteria these desired qualities. Tropos uses the NFR framework [16] to conduct such quality analysis [4].

The structure-in-5 style will be detailed in the next sub-section as it will be used further on in this work.

2.4.1. Structure-in-5 Style

The structure-in-5 organizational style (Figure1) is based on five sub-units. This structure defines a hierarchy of roles inside the organization, the responsibilities associated with each sub-units, and inter-dependencies among them.

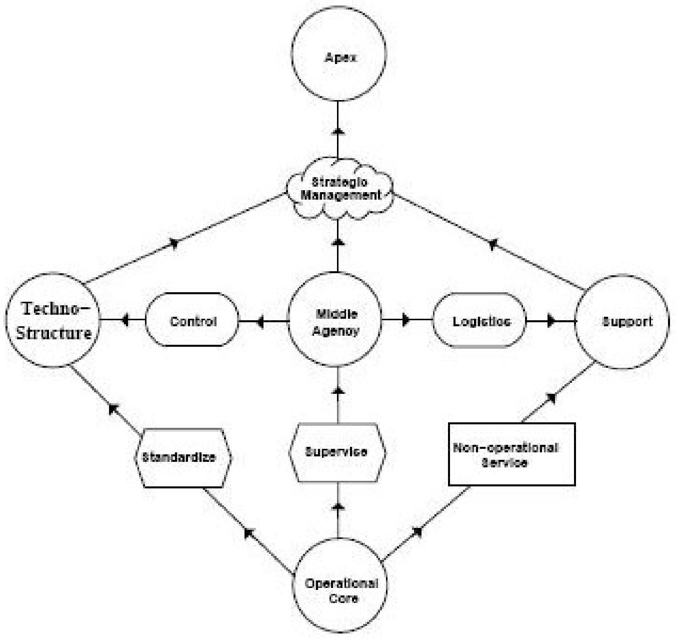


Figure 1. Structure-in-5.

At the base level, the Operational Core takes care of basic tasks – the input, processing, output and direct support procedures – associated with running the organization. At the top lies the Apex, composed of executive roles. Below it, sit the Technostructure, Middle Agency and Support roles, which are in charge of control/standardization, management and logistics, respectively. The Technostructure component carries out the tasks of standardizing the behavior of other components, in addition to applying analytical procedures to help the organization adapt to its environment. Roles 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 [4].

3. The SIRA Process

In order to narrow the structural gap between requirements and architecture, the SIRA Framework was presented in [3] to identify and map key components and interactions, based on system requirements and organizational concepts. It describes a software system from the perspective of an organization and is located in the context of the Tropos project [4].

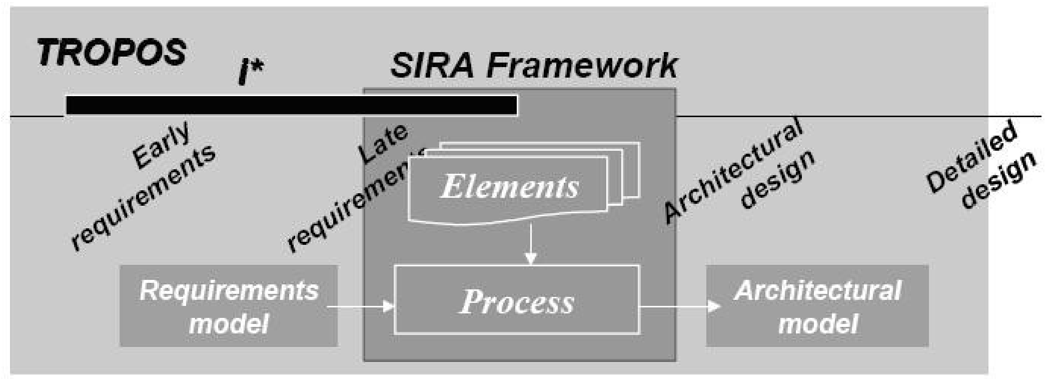


Figure 2. The SIRA Framework in Tropos context.

The organizational view extracted from SD and SR models is used to capture system-related goals in Early Requirements and Late Requirements phases. Although these models provide hints about a deeper understanding of the business process, Tropos does not explicitly cover the correlation between requirement models and architectural elements [3].

The SIRA Process is provided by the SIRA Framework to accomplish the transition from requirements to architectural configuration. It consists of three activities [3]:

1. Organizational Model Specification: identifies the roles and interactions of an organized group. Takes requirements models and an architectural catalogue as input and generates the SIRA Organizational Model as an output.

2. Assignment Model Definition: clusters roles into sub-groups and matches sub-groups to components. Takes the Organizational Model as input and generates the Assignment Model as an output.
3. Architectural Configuration: links the SIRA Organizational Model to an architectural style. Assigns sub-groups to architectural components and generates the architectural model of the MAS. Takes the Assignment Model as input and generates the Architectural Configuration of the MAS.

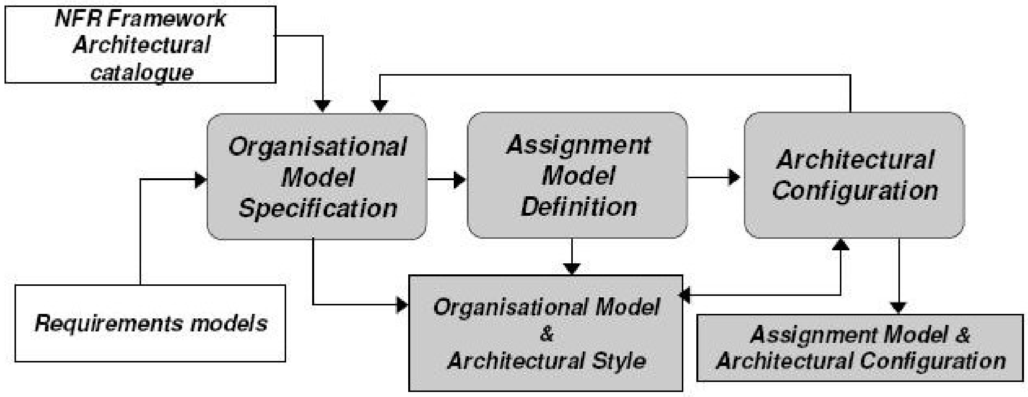

NFR Framework Architectural catalogue
Organisational Model Specification
Assignment Model Definition
Architectural Configuration
Requirements models
Organisational Model & Architectural Style
Assignment Model & Architectural Configuration

Figure 3. The SIRA Process Activities

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

3.1. The Organizational Model Specification

The first activity of the SIRA Process supports the mapping from the i* requirement model onto SIRA Organizational Model. This activity includes three sub-activities: The Goal and Task Refinement, The Role Identification and The Architectural Selection.

3.1.1. Goal and Task Refinement

In this sub-activity, roles are Organizational Groups with responsibilities to be performed. These responsibilities are identified from main dependencies among roles of the requirements model. In Tropos, goal analysis is applied to early and late requirements models to refine them and can be conducted by three techniques [17]:

- Means-ends analysis: identifies tasks, resources and softgoals that provide means for achieving a goal.
- Contribution analysis: identifies goals that can contribute positively or negatively in the fulfillment of the goal to be analyzed.
- AND/OR decomposition: combines AND and OR decompositions of a root goal into sub-goals, modeling a finer goal structure.

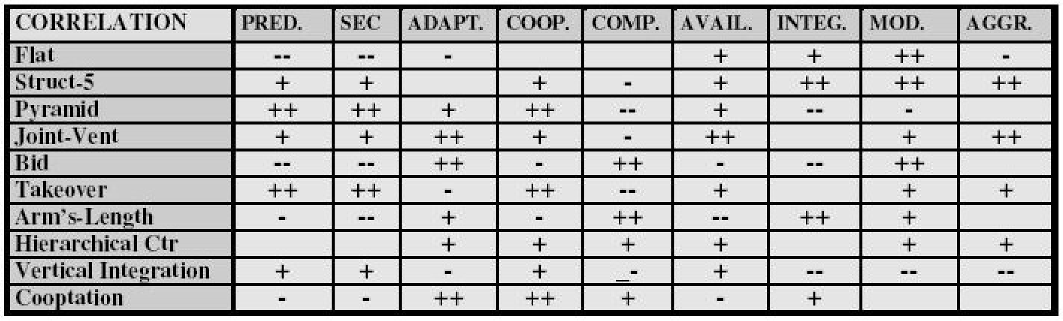
3.1.2. Role Identification

This sub-activity is concerned with the identification of the sequence and type of tasks that can be performed by an role in collaboration with other roles. Therefore, role identification involves the distribution of tasks and interactions to perform each group goal in a coordinated way. In the SIRA Framework, the organizational group is redefined into a set of roles and interactions so it represents the division of work among members [3].

3.1.3. Architectural Selection

As previously mentioned before, the organizational styles from Tropos are evaluated and compared with the non-functional properties of MAS architectures.

The following table summarizes the catalogue for the organizational patterns and top-level quality attributes considered in [21], [22].



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	-	-	++	++	+	-	+		

Table 1. The Correlation Catalogue

The following notation used by the NFR (non-functional requirements) framework +, ++, -, --, represents partial/positive, sufficient/positive, partial/negative and sufficient/negative contributions, respectively [3].

In this sub-activity, 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 categorization 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 [16].

3.2. The Assignment Model Definition

The Assignment Model Definition activity proposes the social network analysis as a method for framing and describing the effects of organizational characteristics to relate multiagent systems and organizational architectures. Networks can have a few or many roles, and one or more kind of relations between pair of roles. To build a useful understanding of a social network, it is necessary to know about all relationships between each pair of roles [3].

This activity includes two sub-activities: Cluster Analysis and Correlation Analysis.

3.2.1. Cluster Analysis

This sub-activity 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. These sub-groups will be compared to architectural components with respect to the set of roles that they perform [3].

To be more specific, this sub-activity measures the extent to which a role is connected to other roles in terms of centrality and structural equivalence:

- Centrality: *Degree centrality* measures the in-degree (receiving relations) and out-degree (sending relations) relations among roles. The greater the number of relations of a role, the higher is his degree centrality. *Closeness centrality* measures how close a role is to other roles. Roles that are able to reach other roles at shorter path lengths (or that are more reachable by other roles) have favored positions in an organization.
- Structural Equivalence: is concerned with identifying roles that have the same pattern of relations with other roles. 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 other sets of roles. As a result, a set of sub-groups are created that clusters a set of similar roles.

3.2.2. Correlation Analysis

This sub-activity is concerned with measuring the degree association between two roles. This is done by examining the correlation between sub-group and components in terms of centrality and similarity correlation analysis [3].

In the centrality correlation analysis, SIRA Framework compares the centrality degree and closeness measures of each pair of roles in the sub-group and architecture style. The first step is to analyze the measures of the most central role in each group. After that, the centrality measures of the each role in organizational group are compared with the centrality measures of each component in architecture. The result is the centrality correlation table [3].

In the similarity correlation analysis, two roles that are structurally equivalent have the same pattern of relation to all other roles -- 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 roles, and for forming structural equivalence groups. SIRA Framework measures the structural equivalence by calculating the Pearson correlation coefficient [19]. The result is the similarity correlation table [3].

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, SIRA Framework analyzes the correlation in terms of strong correlation (++), partial correlation (+) and no correlation () [3].

3.3. The Architectural Configuration

This activity defines the map between the requirement model and the architectural style. The Architectural Configuration Activity assigns sub-groups to each architectural component and generates the architectural configuration of the MAS. This configuration describes the MAS at a macroscopic level in terms of a manageable number of sub-groups using the organizational styles from Tropos [3].

4. The Conference Management Case Study

In this chapter, the Conference Management example will be applied in each activity of the SIRA Process in order to evaluate it.

4.1. The Conference Management Example

A conference involves several individuals and consists of three different phases: the submission phase, the review phase and the final phase.

- In the submission phase, authors submit papers and are informed that their papers have been received and have been assigned a submission number.
- In the review phase, the program committee (PC) has to handle the review of the papers by contacting potential referees and asking them to review a number of papers according to their expertise. Eventually, reviews come in and are used to decide about acceptance or rejection of the submissions.
- In the final phase, authors need to be notified of the decisions and, in case of acceptance, must be asked to produce and submit a revised version of their papers. The publisher has to collect these final versions and print proceedings.

The conference management problem naturally leads to a conception of the whole system as a number of different MAS organizations, one for each phase of the process. In each organization, the corresponding MAS can be viewed as being made up of agents being associated to the persons involved in the process to support their work, and representing the active part of the system. The roles played by each agent reflect the ones played by the associated person in the conference organization [18].

The i* strategic dependency (SD) model for a Conference Management is shown next.

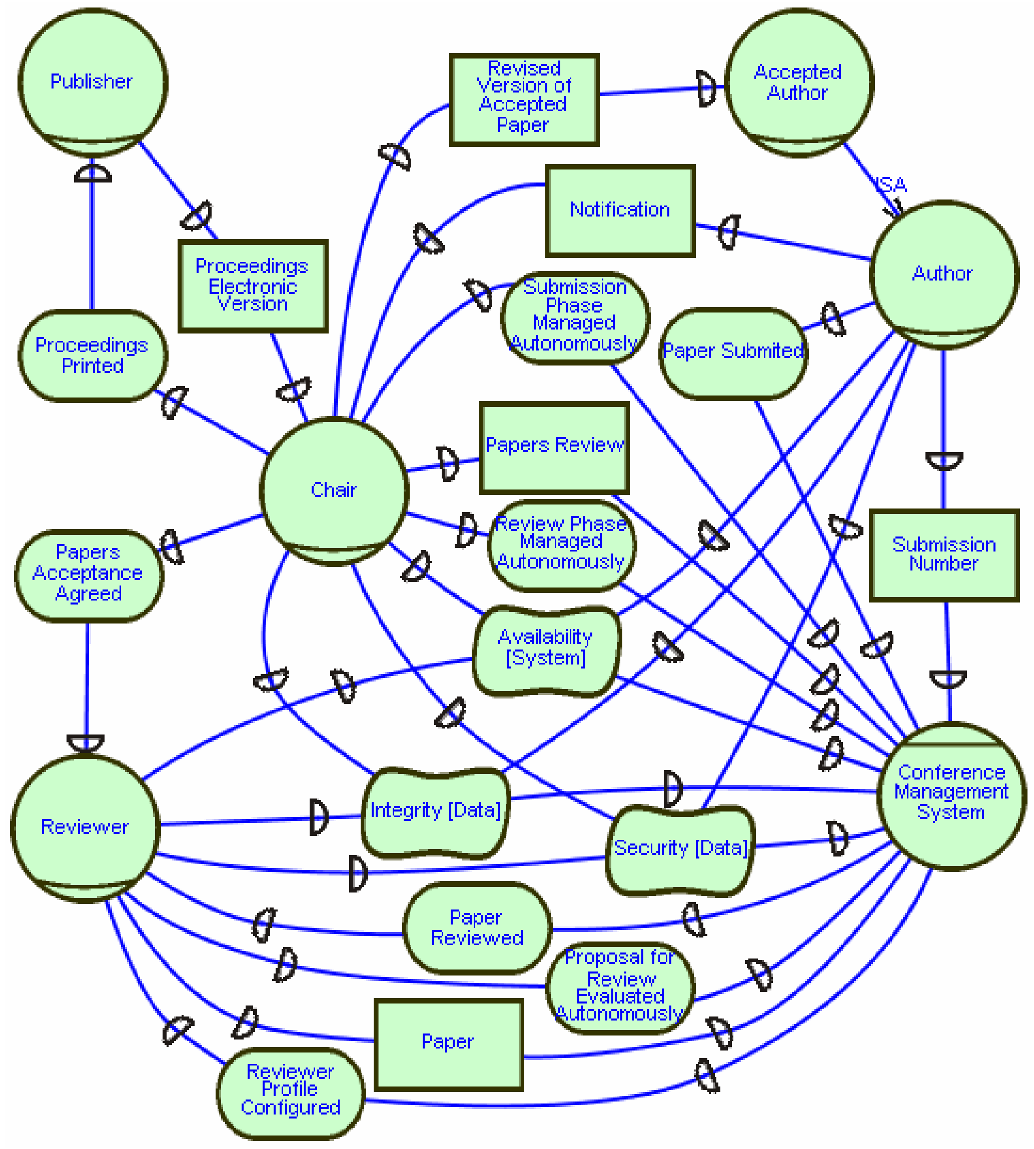

Publisher
Revised Version of Accepted Paper
Accepted Author
ISA
Notification
Author
Proceedings Electronic Version
Submission Phase Managed Autonomously
Paper Submited
Proceedings Printed
Papers Review
Chair
Submission Number
Review Phase Managed Autonomously
Papers Acceptance Agreed
Availability [System]
Conference Management System
Reviewer
Integrity [Data]
Security [Data]
Paper Reviewed
Proposal for Review Evaluated Autonomously
Paper
Reviewer Profile Configured

Figure 4. Strategic Dependency model for a Conference Management.

In the Conference Management Example, the role *Author* depends on *Conference Management System* to have his *Paper Submitted* while *Conference Management System* is expected to give *Author* a *Submission Number* for his submitted paper. Role *Chair* depends on *Conference Management System* to have the *Review* and the *Submission Phase Managed Autonomously* and also needs *Availability* from the system. *Conference Management System* is expected to provide *Papers Review* to role *Chair*. In order to do so, it depends on role *Reviewer* to have a *Paper Reviewed* and to have the *Reviewer Profile*

Configured. Role *Reviewer* depends on *Conference Management System* to have *Proposal for Review Evaluated Autonomously* and needs it to keep I*ntegrity* and S*ecurity* of data exchanged. *Conference Management System* is expected to provide a *Paper* to *Reviewer* role so he can review that paper.

As late requirements analysis proceeds, a strategic rationale (SR) model is necessary to provide support to model the reasons associated with each role and their dependencies. The SR model for the Conference Management System is presented next.

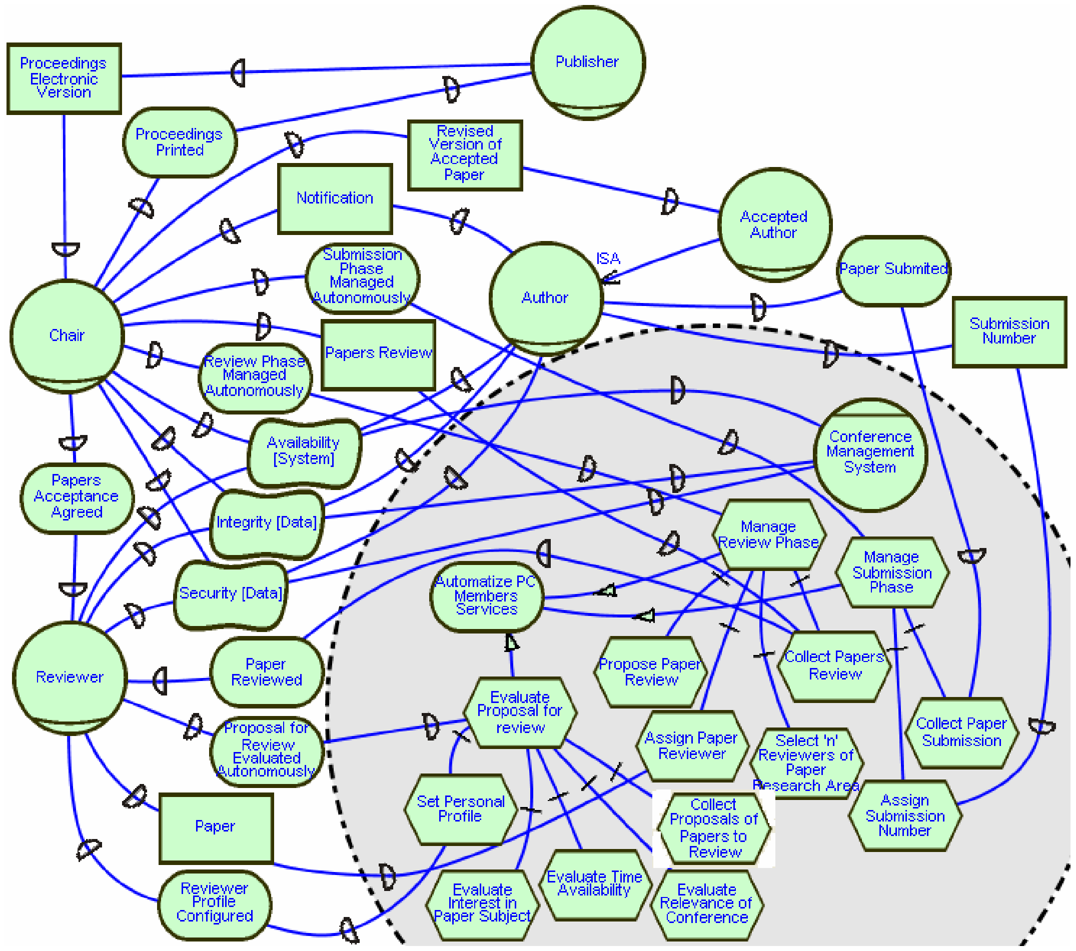

Proceedings Electronic Version
Publisher
Proceedings Printed
Revised Version of Accepted Paper
Notification
Accepted Author
Submission Phase Managed Autonomously
ISA
Author
Paper Submited
Chair
Papers Review
Review Phase Managed Autonomously
Submission Number
Conference Management System
Availability [System]
Papers Acceptance Agreed
Integrity [Data]
Manage Review Phase
Manage Submission Phase
Security [Data]
Automatize PC Members Services
Propose Paper Review
Collect Papers Review
Reviewer
Paper Reviewed
Evaluate Proposal for review
Collect Paper Submission
Proposal for Review Evaluated Autonomously
Assign Paper Reviewer
Select 'n' Reviewers of Paper Research Area
Assign Submission Number
Set Personal Profile
Collect Proposals of Papers to Review
Paper
Evaluate Time Availability
Reviewer Profile Configured
Evaluate Interest in Paper Subject
Evaluate Relevance of Conference

Figure 5. Strategic Rationale model for the Conference Management System.

In the SR for the Conference Management System, the goal *Submission Phase Managed Autonomously* is achieved through the fulfillment of task *Manage Submission Phase*, which is refined into tasks *Collect Paper Submission* and *Assign Submission Number*. The goal *Review Phase Managed Autonomously* is achieved through the fulfillment of task *Manage Review Phase*, which is refined into tasks *Collect Papers Review, Select 'n' Reviewers of Paper Research Area*, *Assign Paper Reviewer* and *Propose Paper Review*. The goal *Proposal for Review Evaluated Autonomously* is achieved through the fulfillment of task *Evaluate Proposal for Review,* which is refined into tasks *Set Personal Profile, Evaluate Interest in Paper Subject*, *Evaluate Time Availability, Evaluate Relevance of Conference* and *Collect Proposals of Papers to Review*. The goal *Paper Submitted* is achieved through the fulfillment of task *Collect Paper Submission*. In this example, we have left three soft-goals (Availability, Security and Integrity) in the late requirements model. They will be used further on in this work to select the architectural style using the NFR framework.

4.2. The Organizational Model of the Conference Management System

As mentioned before, the first activity of the SIRA process is the specification of the Organizational Model and it includes three sub-activities. Each sub-activity will be fulfilled next.

4.2.1. Goal and Task Refinement

From the SR model for the Conference Management example, *Conference Management System* role is an example of an organizational group. As mentioned before, roles *Reviewer, Chair* and *Author* have dependencies of *Conference Management System* role to provide certain goals:

- Role *Reviewer* depends on role *Conference Management System* to fulfill the goal identified as *Proposal for Review Evaluated Autonomously*.

- Role *Chair* depends on role *Conference Management System* to fulfill the goals identified as *Review Phase Managed Autonomously* and *Submission Phase Managed Autonomously*.
- Role *Author* depends on role *Conference Management System* to fulfill the goal identified as *Paper Submitted*.

Means-ends analysis is applied for the discovery of the means (tasks and interactions) that contribute to achieve the objective. The main goals are refined into tasks and sub-tasks. This refinement results in the refinement of tasks into sub-tasks and in the identification of group responsibilities.

In the Conference Management example, the main goals (*Proposal for Review Evaluated Autonomously*, *Review Phase Managed Autonomously*, *Submission Phase Managed Autonomously* and *Paper Submitted*) that were identified from its SR model were already refined into tasks and tasks into sub-tasks and further task refinement does not seem necessary here. This can be better visualized in the following figures.

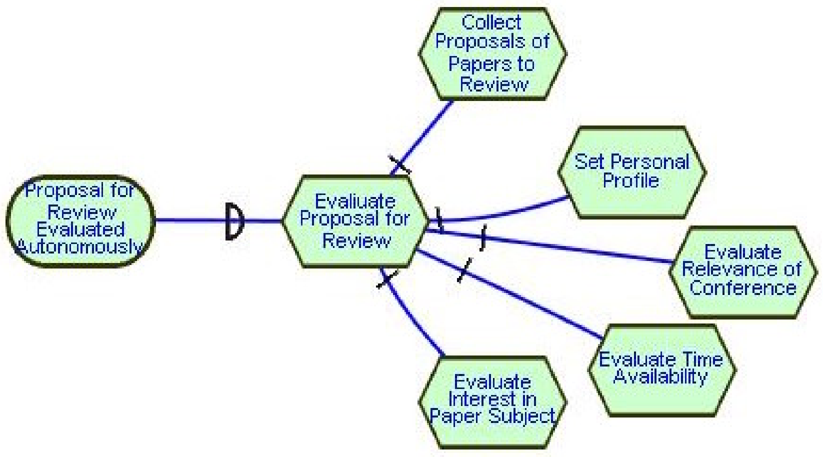

Collect Proposals of Papers to Review
Set Personal Profile
Proposal for Review Evaluated Autonomously
Evaluiate Proposal for Review
Evaluate Relevance of Conference
Evaluate Time Availability
Evaluate Interest in Paper Subject

Figure 6. Refinement of the goal Proposal for Review Evaluated Autonomously.

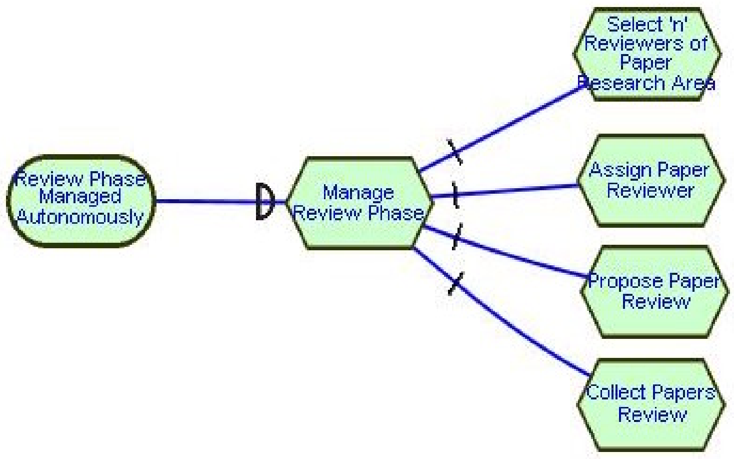


Figure 7. Refinement of the goal Review Phase Managed Autonomously.

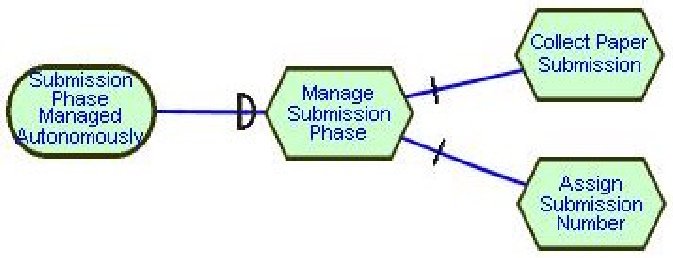


Figure 8. Refinement of the goal Submission Phase Managed Autonomously.

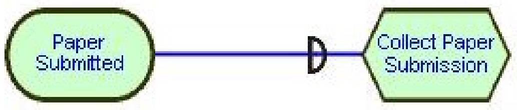


Figure 9. Refinement of the goal Paper Submitted.

4.2.1.1. Goal and Task Refinement Considerations

Applying the Conference Management example in this sub-activity raised a few doubts as to what extent and when goals and tasks should be refined. In the Conference Management example, it did not seem necessary to refine tasks and goals as they were already refined in the SR model.

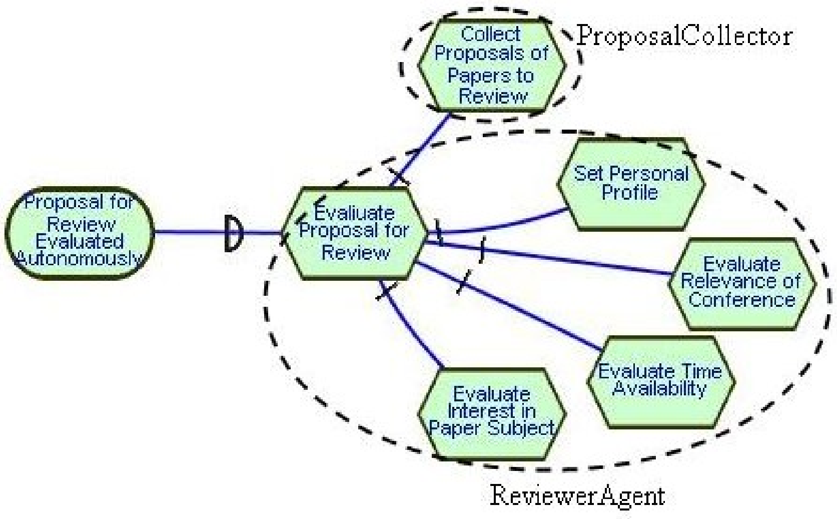
Goal and task refinement sub-activity is a step that can not be automatized due to the fact that it is based on human domain knowledge, experience and intuition.

4.2.2. Role Identification

In general, roles are captured in a specific and bounded domain. In the Conference Management domain, the following roles could be identified:

- Review Manager: role in charge of managing the review phase as well as proposing papers review to reviewers according to their research area.
- Review Catcher: role in charge of selecting reviewers and assigning papers to them.
- Review Collector: role in charge of collecting papers review.
- Reviewer Agent: role responsible for evaluating a paper proposal according to the reviewer preferences or skills.
- Proposal Collector: role in charge of collecting the proposals of papers to be reviewed.
- Submission Handler: role responsible for managing the submission phase as well as assigning submission numbers for papers that were submitted.
- Submission Collector: role in charge of collecting submitted papers.

These roles can be best visualized in the following figures:



Collect Proposals of Papers to Review
ProposalCollector
Proposal for Review Evaluated Autonomously
Evaluate Proposal for Review
Set Personal Profile
Evaluate Relevance of Conference
Evaluate Time Availability
Evaluate Interest in Paper Subject
ReviewerAgent

Figure 10. Role identification of goal Proposal for Review Evaluated Autonomously.

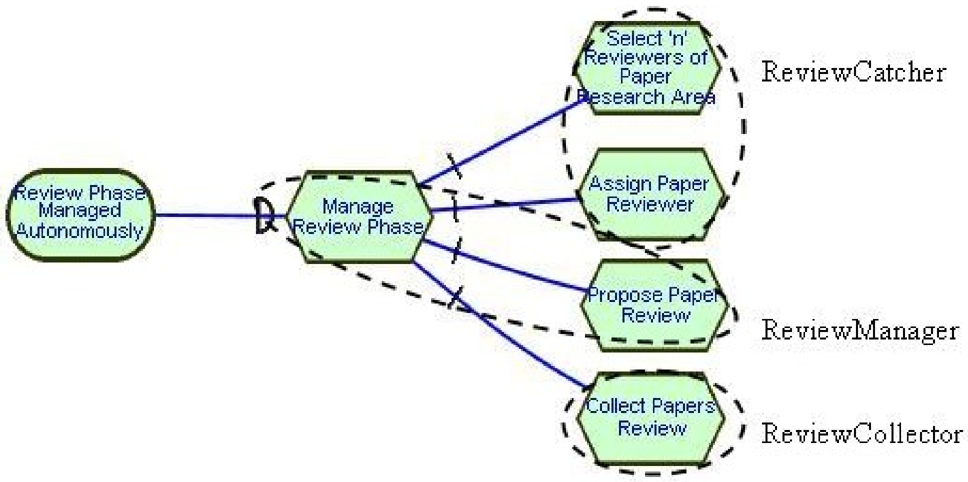


Figure 11. Role identification of goal Review Phase Managed Autonomously.

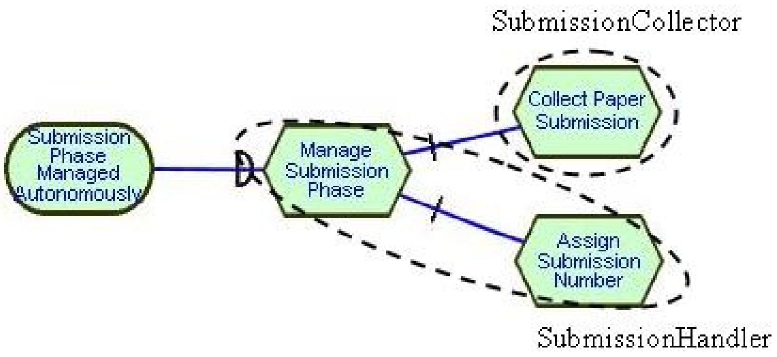


Figure 12. Role identification of goal Submission Phase Managed Autonomously.

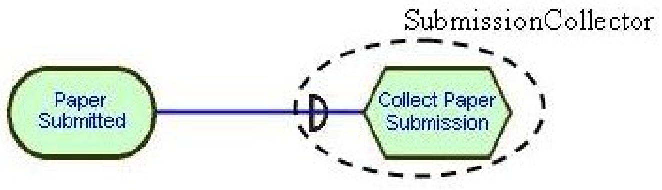


Figure 13. Role identification of goal Paper Submitted.

4.2.2.1. Role Identification Considerations : Identifying roles

Applying the Conference Management example to identify roles was a difficult step as it is not clear how this should be done. The choices were all made based on domain knowledge and intuition.

Once again, identifying the roles is a step that can not be automatized. In role identification, humans subjectively create the roles based on their domain knowledge and intuition.

A role interaction graph representation makes it possible to distinguish between roles that interact and those that do not, and to analyze the relations between sending and receiving information. It is represented as a direct graph, with nodes roles being represented as nodes and the lines representing the interaction sequences among them.

The following figure represents the role interaction graph of goals *Paper Submitted*, *Submission Phase Managed Autonomously*, *Review Phase Managed Autonomously* and *Proposal for Review Evaluated Autonomously*.

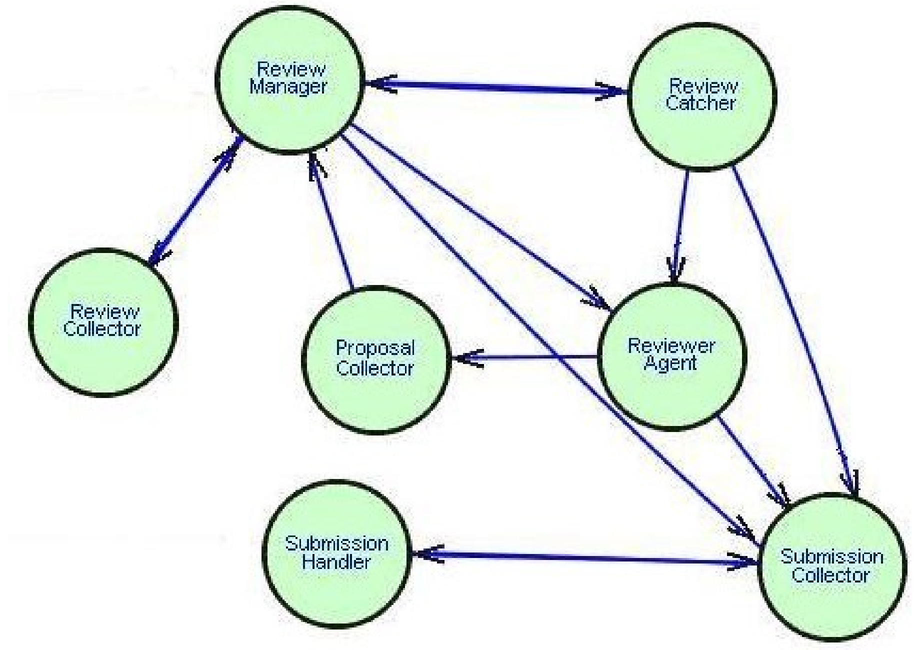


Figure 14. Role interaction graph of Conference Management.

4.2.2.2. Role Identification Considerations: Creating the role interaction graph

Applying the Conference Management example to create the role interaction graph was a very difficult task as, once again, it was entirely based on human domain knowledge and intuition. The choices made for each role will be explained next:

- Role *Review Catcher*:
  - To S*elect 'n' Reviewers of Paper Research Area*, *Review Catcher* needs to have a list of reviewers that must be provided by role *Review Manager* as it is his responsibility to *Manage Review Phase*. This means that *Review Catcher* must interact with *Review Manager*.
  - To *Assign Paper Reviewer*, *Review Catcher* needs information about the papers that were submitted (provided by *Submission Collector*) and about the reviews proposals

that were accepted (provided by *Reviewer Agent*). This means that *Review Catcher* must interact with *Submission Collector* and *Reviewer Agent*.

- Role *Reviewer Agent*:
    - To *Evaluate Interest in Paper Subject*, *Evaluate Time Availability* and *Evaluate Relevance to Conference*, *Reviewer Agent* needs information about the papers that were submitted (provided by *Submission Collector*). This means that *Reviewer Agent* must interact with *Submission Collector*.
    - To *Evaluate Proposal for Review*, *Reviewer Agent* needs information about paper review proposals (provided by *Proposal Collector*). This means that *Reviewer Agent* must interact with *Proposal Collector*.
- Role *Review Collector*:
    - To *Collect Paper Review*, *Review Collector* needs the paper review that is provided by role *Reviewer* to *Review Manager*. This means that *Review Collector* must interact with *Review Manager*.
- Role *Proposal Collector*:
    - To *Collect Proposals of Papers to Review*, *Proposal Collector* needs the proposal of paper review that is provided by *Review Manager*. This means that *Proposal Collector* must interact with *Review Manager*.
- Role *Review Manager*:
    - To *Propose Paper Review*, *Review Manager* needs information about submitted papers (provided by *Submission Collector*) and about reviewers (provided by *Reviewer*

    Agent). This means that *Review Manager* must interact with *Submission Collector* and *Reviewer Agent.*

    - To *Manage Review Phase*, *Review Manager* needs paper reviews provided by *Review Collector*. This means that *Review Manager* must interact with *Review Collector*.

- Role *Submission Collector*:
    - To *Collect Paper Submission*, *Submission Collector* needs the papers that were submitted that are provided by role *Author* to *Submission Handler*. This means that *Submission Collector* must interact with *Submission Handler*.
- Role *Submission Handler*:
    - To *Assign Submission Number* and *Manage Submission Phase*, *Submission Handler* needs information about submitted papers provided by *Submission Collector*. This means that *Submission Handler* must interact with *Submission Collector*.

In Role Identification sub-activity, creating a role interaction graph is a complicated and difficult step and can not be automatized. Here, once again, humans subjectively create the interactions among roles based on their domain knowledge and intuition.

This role interaction graph can also be viewed as a binary matrix. By convention, the sender of the relation is the row and the target is the column. In binary relations, zeros indicate absence and ones indicate presence of each logically possible relation between pairs of roles. The role interaction graph represented as an interaction matrix is shown next.

	Review Manager	Review Catcher	Reviewer Agent	Review Collector	Proposal Collector	Submission Handler	Submission Collector
Review Manager	--	1	1	1	0	0	1
Review Catcher	1	--	1	0	0	0	1
Reviewer Agent	0	0	--	0	1	0	1
Review Collector	1	0	0	--	0	0	0
Proposal Collector	1	0	0	0	--	0	0
Submission Handler	0	0	0	0	0	--	1
Submission Collector	0	0	0	0	0	1	--

Table 2. Interaction matrix table of the Conference Management.

4.2.3. Architectural Selection

The choice of an architectural style is based on the application domain. In the conference management domain, the Structure-in-5 architectural style has been chosen using the NFR framework.

The software quality attributes *Availability*, *Security* and *Integrity* that have been left in the late requirements model (SR model) guided the selection process of the appropriated architectural style. They have been compared against the set of organizational styles as shown in the next table.

CORRELATION	SECURITY	AVAILABILITY	INTEGRITY
Structure-in-5	+	+	++
Pyramid	++	+	--
Joint Venture	+	++	
Takeover	++	+	

Table 3. The Correlation Catalogue for the Conference Management.

The role interaction graph for the Structure-in-5 architectural style is shown in the following figure and was defined in [3].

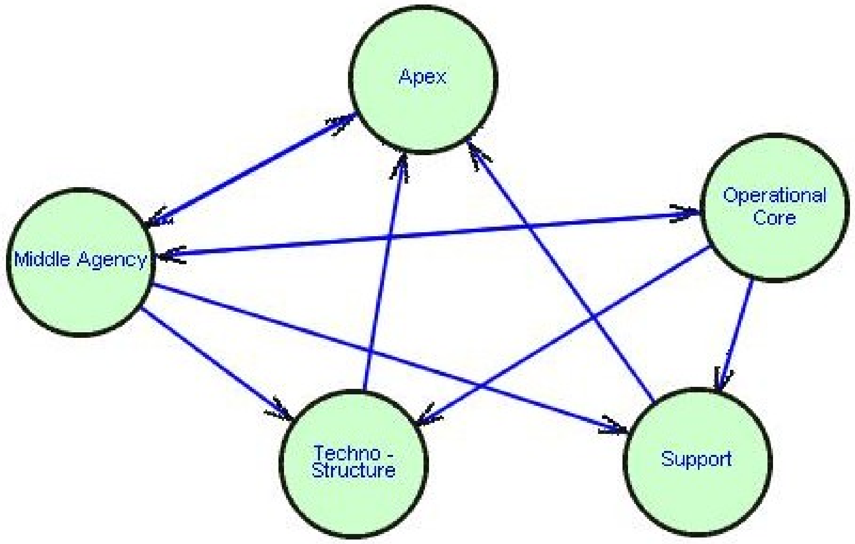


Figure 15. Role interaction graph of the Structure-in-5 architectural style.

The interaction matrix of the Structure-in-5 architectural style is shown next.

	Apex	Middle Agency	Operational Core	Techno-structure	Support
Apex	--	1	0	0	0
Middle Agency	1	--	1	1	1
Operational Core	0	1	--	1	1
Techno-structure	1	0	0	--	0
Support	1	0	0	0	--

Table 4. Interaction matrix table of the Structure-in-five architectural style.

4.2.3.1. Architectural Selection Considerations

Selecting the architectural style is a step that can be easily automatized as the NFR framework is used to select the architectural style and the role interaction graph and the interaction matrix of the architectural styles do not change once defined.

4.3. The Assignment Model of the Conference Management System

As mentioned before, the second activity of the SIRA process is the definition of the Assignment Model and it includes two sub-activities. Each sub-activity will be fulfilled next.

4.3.1. Cluster Analysis

This sub-activity analyzes the role interaction graph to measure the extent to which a role is connected to other roles in terms of centrality and structural equivalence.

4.3.1.1. Centrality Equivalence

The first step in cluster analysis is the centrality analysis. It indicates the extent to which a group is organized around its most central role and it is measured in terms of *degree centrality* and *closeness centrality*.

As mentioned before, the *degree centrality* measures the in-degree (receiving) and out-degree (sending) relations among roles. The greater the number of relations of a role, the higher is his degree centrality.

From the role interaction graph of the conference management example (Figure 14), the following table of degree centrality measures was created.

	Out Degree	In Degree	Nrm - out	Nrm - in
Review Manager	4	3	100	75
Review Catcher	3	1	75	25
Reviewer Agent	2	2	50	50
Review Collector	1	1	25	25
Proposal Collector	1	1	25	25
Submission Handler	1	1	25	25
Submission Collector	1	4	25	100

Table 5. Degree centrality measures of the Conference Management.

When evaluating the interactions to achieve the goals identified, *Review Manager* has the greatest out-degree and a high in-degree value and can be considered the most influential role. This means that *Review Manager* is the most central role in terms of degree centrality.

From the role interaction graph of the structure-in-5 architectural style (Figure 15), the following table of degree centrality measures was created.

	Out Degree	In Degree	Nrm - out	Nrm - in
Apex	1	3	25	75
Middle Agency	4	2	100	50
Operational Core	3	1	75	25
Techno-Structure	1	2	25	50
Support	1	2	25	50

Table 6. Degree Centrality measures of the Structure-in-5 architectural style.

When evaluating the interactions of the Structure-in-5 style, *Middle Agency* has the greatest out-degree and a high in-degree value and can be considered the most influential role. This means that *Middle Agency* is the most central role in terms of degree centrality.

As mentioned before, *closeness centrality* measures how close a role is to other roles. Roles that are able to reach other roles at shorter path lengths (or that are more reachable by other roles) have favored positions in an organization.

From the role interaction graph of the conference management example (Figure 14), we can also determine the following closeness centrality table.

	Farness	Closeness
Review Manager	8	100
Review Catcher	10	80
Reviewer Agent	12	66,66
Review Collector	15	53,33
Proposal Collector	12	66,66
Submission Handler	x	x
Submission Collector	x	x

Table 7. Closeness centrality measures of Conference Management.

The farness value of a role is the sum of the geodesic distances from that role to all other roles. Let’s take role *Review Manager* as an example and see how we calculate his farness. From the interaction graph (Figure 14), the distance from *Review Manager* to:

- *Review Collector* is 1;
- *Review Catcher* is 1;

- *Reviewer Agent* is 1;
- *Submission Collector* is 1;
- *Proposal Collector* is 2: because he does not communicate directly with this role. First he has to communicate with *Reviewer Agent* and then reach *Proposal Collector*;
- *Submission Handler* is 2: he does not communicate directly with this role. First he has to communicate with *Submission Collector*.

The sum of all these geodesic distances is 8 (1 + 1 + 1 + 1 + 2 + 2), so *Review Manager* has a farness value of 8.

Farness can be converted into a measure of closeness centrality by taking the reciprocal (one divided by farness) and normalizing it in relation to the most central role [3].

From the closeness centrality measures table, *Review Manager* is the most central role in terms of closeness centrality.

From the role interaction graph of the structure-in-5 architectural style (Figure 15), we can also determine the following closeness centrality table.

	Farness	Closeness
Apex	7	57
Middle Agency	4	100
Operational Core	5	80
Techno-Structure	9	44
Support	9	44

Table 8. Closeness centrality measures of the Structure-in-5 architectural style.

From this table, *Middle Agency* is the most central role in terms of closeness centrality.

This first step in cluster analysis measures the overall integration of a group. Based on the degree centrality (Table 5) and closeness centrality (Table 7) tables, it is possible to conclude that:

- *Review Collector, Proposal Collector* and *Submission Handler* have identical degree centrality and similar degree centrality with *Reviewer Agent*;
- *Reviewer Agent* and *Proposal Collector* have identical closeness centrality and similar closeness centrality with *Review Collector.*

4.3.1.1.1. Centrality Equivalence Considerations

Applying the Conference Management example to calculate closeness centrality raised a few doubts. Some roles, like Submission Collector and Submission Handler, do not have a sending relation and farness values could not be calculated. Despite that, the first step of cluster analysis sub-activity can be automatized as it consists mainly on calculations based on results from previous steps.

4.3.1.2. Similarity Equivalence

The second step in cluster analysis is to group together the roles that are most similar in a group. The Pearson’s correlation formula is used to calculate the similarity coefficient of each pair of role in a group. The following table shows the correlation coefficient (r) and how it is interpreted in the SIRA Process.

-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 +1,0	++	**Strong positive correlation**

Table 9. Interpretation of the correlation coefficients table.

To compute the correlation coefficient (r) of each pair of role in the Conference Management example, we use its interaction matrix (Table 2) and the Pearson’s Formula as shown next.

$$r = \frac{n\sum XY - \sum X \sum Y}{\sqrt{\left[n\sum X_2 - \left(\sum X\right)_2\right]\left[n\sum Y_2 - \left(\sum Y\right)_2\right]}}$$

Figure 16. The Pearson's correlation formula.

The variables X and Y represent the two roles that are being analyzed and ‘n’ is the number of roles in the group. In the Conference Management group, if we take roles *Review Manager* and *Review Catcher* respectively, then X = {0, 1, 1, 1, 0, 0, 1}, Y = {1, 0, 1, 0, 0, 0, 1} and n = 7. The similarity coefficients of the Conference Management example are presented next.

	Review Manager	Review Catcher	Reviewer Agent	Review Collector	Proposal Collector	Submission Handler	Submission Collector
Review Manager	1						
Review Catcher	0,1667	1					
Reviewer Agent	-0,0913	0,0913	1				
Review Collector	-0,4714	0,4714	-0,2582	1			
Proposal Collector	-0,4714	0,4714	-0,2582	1	1		
Submission Handler	0,3536	0,4714	0,6455	-0,1667	-0,1667	1	
Submission Collector	-0,4714	-0,3536	-0,2582	-0,1667	-0,1667	-0,1667	1

Table 10. Similarity Coefficients table of the Conference Management.

Before comparing a group with an architectural style, it might be necessary to cluster roles into sub-groups. As the Structure-in-5 architectural style is defined with five fixed positions, the roles of the Conference Management example must be clustered into 5 sub-groups, each of which clusters a set of similar roles.

From the similarity coefficients table and based on centrality equivalence analysis, it is possible to conclude that:

- *Review Collector* and *Proposal Collector* have the strongest positive correlation (r = 1 means that they have perfect structural equivalence), identical degree centrality and similar closeness centrality;

- *Submission Handler* and *Reviewer Agent* have a partial positive correlation (r = 0,6455) and similar degree centrality.

Based on these results, roles *Proposal Collector* and *Review Collector* are clustered in a sub-group named *General Collector*. Also, roles *Submission Handler* and *Reviewer Agent* are clustered in a sub-group named *Review Handler*.

The interaction graph, interaction matrix and centrality measures table of the Conference Management after the cluster analysis are presented next.

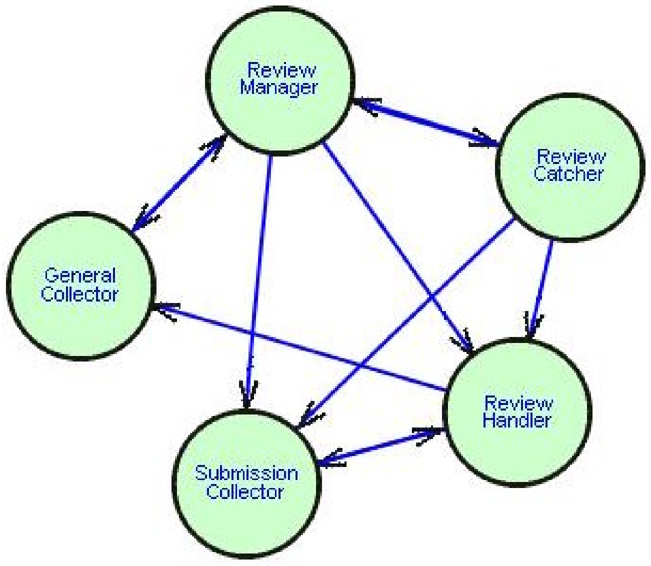


Figure 17. Role Interaction graph after the cluster analysis.

The role interaction graph after the cluster analysis is created based on the role interaction graph before the cluster analysis. If the roles that were clustered in a sub-group had in-degree relations and out-degree relation to other roles, the sub-group that clusters these roles must have the same relations. For example, role *Submission Handler* had an out-degree relation and an in-degree relation with role *Submission Collector* which means that *Review Handler* must have the same relations with *Submission Collector*.

	Review Manager	Review Catcher	Review Handler	Submission Collector	General Collector
Review Manager	--	1	1	1	1
Review Catcher	1	--	1	1	0
Review Handler	0	0	--	1	1
Submission Collector	0	0	1	--	0
General Collector	1	0	0	0	--

Table 11. Interaction matrix table after the cluster analysis.

	Out Degree	In Degree	Nrm - out	Nrm - in	Farness	Closeness
Review Manager	4	2	100	50	4	100
Review Catcher	3	1	75	25	5	80
Review Handler	2	3	50	75	7	57
Submission Collector	1	3	25	75	10	40
General Collector	1	2	25	50	7	57

Table 12. Centrality measures table after the cluster analysis.

This cluster analysis of the Conference Management example will be used to relate the Conference Management roles and the Structure-in-5 components in the correlation analysis.

4.3.1.2.1. Similarity Equivalence Considerations

The second step of cluster analysis sub-activity can be automatized as it consists mainly on calculations based on results from previous steps.

4.3.2. Correlation Analysis

As mentioned before, this sub-activity examines the correlation between sub-groups and components in terms of centrality and similarity correlation analysis.

4.3.2.1.The Centrality Correlation Analysis

The first correlation analysis between Conference Management sub-groups and Structure-in-5 components is based on degree and closeness centrality measures. To compare Conference Management to Structure-in-5 components, the following tables are used:

- The centrality measures table of Conference Management after the cluster analysis (Table 12);
- The degree centrality (Table 6) and closeness centrality (Table 8) measures table of Structure-in-5.

From these tables, the following results can be observed:

- *Review Manager* is the most central sub-group from the Conference Management (out-degree = 100, in-degree = 50, closeness = 100) and can be related to *Middle Agency*, which is also the most central component of Structure-in 5 (out-degree = 100, in-degree = 50, closeness = 100). They have identical centrality measures;
- *Review Catcher* sub-group (out-degree = 75, in-degree = 25, closeness = 80) and *Operational Core* component (out-degree = 75, in-degree = 25, closeness = 80) have a strong centrality correlation (++) as they also have identical centrality measures;
- *Review Handler* sub-group (out-degree = 50, in-degree = 75, closeness = 57) and *Apex* component (out-degree = 25, in-degree = 75, closeness = 57) have a partial centrality correlation (+) as they have similar centrality measures;
- *General Collector* sub-group (out-degree = 25, in-degree = 50, closeness = 57) and *Apex* component (out-degree = 25, in-degree = 75, closeness = 57) have a partial centrality correlation (+) as they have similar centrality measures;
- *General Collector* sub-group (out-degree = 25, in-degree = 50, closeness = 57) and *Support* component (out-degree = 25, in-degree = 50, closeness = 44) have a partial centrality correlation (+) as they have similar centrality measures;
- *General Collector* sub-group (out-degree = 25, in-degree = 50, closeness = 57) and *Techno-Structure* component (out-degree = 25,

in-degree = 50, closeness = 44) have a partial centrality correlation (+) as they have similar centrality measures;

- The other sub-groups and components have no centrality correlation.

	Nrm - out	Nrm - in	Closeness	Degree Score	Central Role
Structure-in-5	100	50	100	1	Middle Agency
Conference Management group	100	50	100	1	Review Manager
	STRUCTURE-IN-5 COMPONENTS				
CONFERENCE MANAGEMENT SUB-GROUPS	**Apex**	**Middle Agency**	**Operational Core**	**Techno-structure**	**Support**
Review Manager		++			
Review Catcher			++		
Review Handler	+				
Submission Collector					
General Collector	+			+	+

Table 13. Centrality Correlation between Conference Management sub-groups and Structure-in-5 components.

4.3.2.1.1. Centrality Correlation Analysis Considerations

The centrality correlation analysis step can be automatized as it consists mainly on comparing values from previous defined tables.

After the centrality correlation analysis, the pattern of relations of the selected organizational style is evaluated in terms of its similarity correlation with SIRA sub-groups.

4.3.2.2.The Similarity Correlation Analysis

The next step is to check if there is at least one similarity correlation between Conference Management sub-groups and the Structure-in-5 components. These sub-groups can become the architectural components.

This analysis is based on in-degree and out-degree relations from the interaction matrix of Conference Management after the cluster analysis (Table 11) and the interaction matrix of the Structure-in-5 (Table 4). It looks across the rows (out-degree) and columns (in-degree) for investigating the structural similarity coefficient for each pair of roles using the Pearson's Correlation formula as explained next:

- If we take sub-group *Review Manager* and component *Apex* respectively and our analysis is based on out-degree, then X = {0, 1, 1, 1, 1}, Y = {0, 1, 0, 0, 0}.
- If we take sub-group *Review Manager* and component *Apex* respectively and our analysis is based on in-degree, then X = {0, 1, 0, 0, 1}, Y = {0, 1, 0, 1, 1}.

The following table shows the similarity matrix based on In-degree relations for the Conference Management example.

	Apex	Middle Agency	Operational Core	Techno-structure	Support
Review Manager	0,6667	-0,6667	0,6124	0,1667	0,1667
Review Catcher	-0,6124	0,6124	-0,2500	-0,4082	-0,4082
Review Handler	0,1667	-0,1667	0,4082	-0,1667	-0,1667
Submission Collector	-0,6667	0,6667	0,4082	0,6667	0,6667
General Collector	-1,0000	1,0000	-0,4082	0,1667	0,1667

Table 14. Similarity matrix based on In-degree relations.

These Pearson correlation results show that:

- *Review Manager* sub-group has a partial positive correlation with components *Apex* and *Operational Core*;
- *Review Catcher* sub-group has a partial positive correlation with component *Middle Agency*;
- *Review Handler* sub-group has a partial positive correlation with component *Operational Core*;
- *Submission Collector* sub-group has a partial positive correlation with components *Middle Agency*, *Operational Core*, *Techno-Structure* and *Support*;
- *General Collector* sub-group has a strong positive correlation with component *Middle Agency*. They are identical in terms of in-degree relations.

The following table shows the similarity matrix based on Out-degree relations for the Conference Management example.

	Apex	Middle Agency	Operational Core	Techno-structure	Support
Review Manager	0,2500	-0,2500	0,6124	-1,0000	-1,0000
Review Catcher	-0,6124	0,6124	-0,6667	0,4082	0,4082
Review Handler	-0,4082	0,4082	0,6667	-0,4082	-0,4082
Submission Collector	-0,2500	0,2500	-0,6124	-0,2500	-0,2500
General Collector	-0,2500	0,2500	-0,6124	1,0000	1,0000

Table 15. Similarity matrix based on Out-degree relations.

These Pearson correlation results show that:

- *Revi*ew *Manager* sub-group has a partial positive correlation with component *Operational Core*;
- *Review Catcher* sub-group has a partial positive correlation with components *Middle Agency, Techno-Structure* and *Support*;
- *Review Handler* sub-group has a partial positive correlation with components *Middle Agency* and *Operational Core*;
- *Submission Collector* sub-group has no positive correlation;
- *General Collector* sub-group has a strong positive correlation with components *Techno-Structure* and *Support*. They are identical in terms of in-degree relations.

The following table summarizes the in-degree and out-degree results of the similarity correlation of the Conference Management example.

	Apex	Middle Agency	Operational Core	Techno-structure	Support
Review Manager	+	-	+	--	--
Review Catcher	-	+	-		
Review Handler		+	+	-	-
Submission Collector	-	+		+	+
General Collector	--	++	-	++	++

Table 16.Similarity Correlation table.

This similarity correlation table is created based on the in-degree (Table 14) and out-degree (Table 15) similarity tables. Although it is not clear in [3] how to combine these tables, this is done by considering how the NFR framework deals with conflicts (negative correlations) and harmony (positive correlations) of non-functional requirements in software engineering in [23].

To best express this solution, we explain the results of Table 16 as follows:

- *Review Manager* sub-group and *Apex* component:
    - In-degree: Partial positive correlation, r = 0,6667;
    - Out-degree: Little positive correlation, r = 0,2500;
    - We can conclude that *Review Manager* has a partial positive correlation with component *Apex*: (+);
- *Review Manager* sub-group and *Middle Agency* component:
    - In-degree: Partial negative correlation, r = -0,6667;
    - Out-degree: Little negative correlation, r = -0,2500;
    - We can conclude that *Review Manager* has a partial negative correlation with component *Middle Agency*: (-);
- *Review Manager* sub-group and *Operational Core* component:
    - In-degree: Partial positive correlation, r = 0,6124;
    - Out-degree: Partial positive correlation, r = 0,6124;
    - We can conclude that *Review Manager* has a partial positive correlation with component *Operational Core*: (+);
- *Review Manager* sub-group and *Techno-Structure* component:
    - In-degree: Little positive correlation, r = 0,1667;
    - Out-degree: Strong negative correlation, r =-1,0000;
    - We can conclude that *Review Manager* has a strong negative correlation with component *Techno-Structure*: (--);
- *Review Manager* sub-group and *Support* component:

    - In-degree: Little positive correlation, r = 0,1667;
    - Out-degree: Strong negative correlation, r =-1,0000;
    - We can conclude that *Review Manager* has a strong negative correlation with component *Support*: (--);
- *Review Catcher* sub-group and *Apex* component:
    - In-degree: Partial negative correlation, r = -0,6124;
    - Out-degree: Partial negative correlation, r = -0,6124;
    - We can conclude that *Review Catcher* has a partial negative correlation with component *Apex*: (-);
- *Review Catcher* sub-group and *Middle Agency* component:
    - In-degree: Partial positive correlation, r = 0,6124;
    - Out-degree: Partial positive correlation, r = 0,6124;
    - We can conclude that *Review Catcher* has a partial positive correlation with component *Middle Agency*: (+);
- *Review Catcher* sub-group and *Operational Core* component:
    - In-degree: Little negative correlation, r = -0,2500;
    - Out-degree: Partial negative correlation, r = -0,6667;
    - We can conclude that *Review Catcher* has a partial negative correlation with component *Operational Core*: (-);
- *Review Catcher* sub-group and *Techno-Structure* component:
    - In-degree: Partial negative correlation, r = -0,4082;
    - Out-degree: Partial positive correlation, r = 0,4082;
    - We can conclude that *Review Catcher* has no correlation with component *Techno-Structure*: ( );
- *Review Catcher* sub-group and *Support* component:
    - In-degree: Partial negative correlation, r = -0,4082;

    - Out-degree: Partial positive correlation, r = 0, 4082;
    - We can conclude that *Review Catcher* has no correlation with component *Support*: ( );
- *Review Handler* sub-group and *Apex* component:
    - In-degree: Little positive correlation, r = 0,1667;
    - Out-degree: Partial negative correlation, r = -0,4082;
    - We can conclude that *Review Handler* has a partial negative correlation with component *Apex*: (-);
- *Review Handler* sub-group and *Middle Agency* component:
    - In-degree: Little negative correlation, r = -0,1667;
    - Out-degree: Partial positive correlation, r = 0,4082;
    - We can conclude that *Review Handler* has a partial positive correlation with component *Middle Agency*: (+);
- *Review Handler* sub-group and *Operational Core* component:
    - In-degree: Partial positive correlation, r = 0,4082;
    - Out-degree: Partial positive correlation, r = 0,6667;
    - We can conclude that *Review Handler* has a partial positive correlation with component *Operational Core*: (+);
- *Review Handler* sub-group and *Techno-Structure* component:
    - In-degree: Little negative correlation, r = -0,1667;
    - Out-degree: Partial negative correlation, r = -0,4082;
    - We can conclude that *Review Handler* has a partial negative correlation with component *Techno-Structure*: (-);
- *Review Handler* sub-group and *Support* component:
    - In-degree: Little negative correlation, r = -0,1667;
    - Out-degree: Partial negative correlation, r = -0,4082;

    - We can conclude that *Review Catcher* has a partial negative correlation with component *Support*: (-);
- *Submission Collector* sub-group and *Apex* component:
    - In-degree: Partial negative correlation, r = -0,6667;
    - Out-degree: Little negative correlation, r = -0,2500;
    - We can conclude that *Submission Collector* has a partial negative correlation with component *Apex*: (-);
- *Submission Collector* sub-group and *Middle Agency* component:
    - In-degree: Partial positive correlation, r = 0,6667;
    - Out-degree: Little positive correlation, r = 0,2500;
    - We can conclude that *Submission Collector* has a partial positive correlation with component *Middle Agency*: (+);
- *Submission Collector* sub-group and *Operational Core* component:
    - In-degree: Partial positive correlation, r = 0,4082;
    - Out-degree: Partial negative correlation, r = -0,6124;
    - We can conclude that *Submission Collector* has no correlation with component *Operational Core*: ( );
- *Submission Collector* sub-group and *Techno-Structure* component:
    - In-degree: Partial positive correlation, r = 0,6667;
    - Out-degree: Little negative correlation, r = -0,2500;
    - We can conclude that *Submission Collector* has a partial positive correlation with component *Techno-Structure*: (+);
- *Submission Collector* sub-group and *Support* component:
    - In-degree: Partial positive correlation, r = 0,6667;
    - Out-degree: Little negative correlation, r = -0,2500;

    - We can conclude that *Submission Collector* has a partial positive correlation with component *Support*: (+);

- *General Collector* sub-group and *Apex* component:
    - In-degree: Strong negative correlation, r = -1,0000;
    - Out-degree: Little negative correlation, r = -0,2500;
    - We can conclude that *General Collector* has a strong negative correlation with component *Apex*: (--);
- *General Collector* sub-group and *Middle Agency* component:
    - In-degree: Strong positive correlation, r = 1,0000;
    - Out-degree: Little positive correlation, r = 0,2500;
    - We can conclude that *General Collector* has a strong positive correlation with component *Middle Agency*: (++);
- *General Collector* sub-group and *Operational Core* component:
    - In-degree: Partial negative correlation, r = -0,4082;
    - Out-degree: Partial negative correlation, r = -0,6124;
    - We can conclude that *General Collector* has a partial negative correlation with component *Operational Core*: (-);
- *General Collector* sub-group and *Techno-Structure* component:
    - In-degree: Little positive correlation, r = 0,1667;
    - Out-degree: Strong positive correlation, r = 1,0000;
    - We can conclude that *General Collector* has a strong positive correlation with component *Techno-Structure*: (++);
- *General Collector* sub-group and *Support* component:
    - In-degree: Little positive correlation, r = 0,1667;
    - Out-degree: Strong positive correlation, r = 1,0000;

- We can conclude that *Submission General* has a strong positive correlation with component *Support*: (++).

4.3.2.2.1. Similarity Correlation Analysis Considerations

The similarity correlation analysis step can also be automatized as it consists mainly on comparing values from previous defined tables and generating a new table based on these values.

Based on these centrality and similarity correlation analysis, it is possible to derive the first configuration of the Conference Management architecture.

4.4. The Architectural Configuration

As mentioned before, this activity defines the map between the requirement model and the architectural style.

Based on the similarity correlation table (Table 16) and on the centrality correlation table (Table 13), the following table was created and it represents the correlation analysis of the Conference Management example.

	Apex	Middle Agency	Operational Core	Techno-structure	Support
Review Manager		+			
Review Catcher			+		
Review Handler	+				
Submission Collector				+	
General Collector					++

Table 17. Correlation Analysis of the Conference Management example.

To best express how this table was created, we explain the results of Table 17 as follows:

- *Review Manager* sub-group and *Middle Agency* component:
  - From Table 16: Partial negative correlation (-);
  - From Table 13: Strong positive correlation (++);

    - We can conclude that *Review Manager* has a partial positive correlation with component *Middle Agency*: (+);
- *Review Catcher* sub-group and *Operational Core* component:
    - From Table 16: Partial negative correlation (-);
    - From Table 13: Strong positive correlation (++);
    - We can conclude that *Review Catcher* has a partial positive correlation with component *Operational Core*: (+);
- *Review Handler* sub-group and *Apex* component:
    - From Table 16: Little negative correlation (-);
    - From Table 13: Partial positive correlation (+);
    - We can conclude that *Review Handler* has a partial positive correlation with component *Apex*: (+);
- *Submission Collector* sub-group and *Techno-Structure* component:
    - From Table 16: Partial positive correlation (+);
    - From Table 13: no correlation ();
    - We can conclude that *Submission Collector* has a partial positive correlation with component *Techno-Structure*: (+);
- *General Collector* sub-group and *Support* component:
    - From Table 16: Strong positive correlation (++);
    - From Table 13: Partial positive correlation (+);
    - We can conclude that *General Collector* has a strong positive correlation with component *Support*: (++).

Based on the results of the correlation analysis (Table 17), it is possible to relate the Conference Management sub-groups to the Structure-in-5 components as following:

- The sub-group *Review Manager* is related to *Middle Agency* component;
- The sub-group *Review Catcher* is related to *Operational Core* component;
- The sub-group *Review Handler* is related to *Apex* component;
- The sub-group *Submission Collector* is related to *Techno-Structure* component;
- The sub-group *General Collector* is related to *Support* component.

4.4.1. Architectural Configuration Considerations

The process of generating the correlation analysis table (Table 17) based on the similarity correlation table (Table 16) and on the centrality correlation table (Table 13), is not clearly defined in [3]. Again, this is done by considering how the NFR framework deals with conflicts (negative correlations) and harmony (positive correlations) of non-functional requirements in software engineering in [23].

The architectural configuration step can also be automatized as it consists mainly on comparing values from previous defined tables and generating a new table based on these values. From this new table, the values are analyzed and sub-groups and component are related deriving the first architectural configuration.

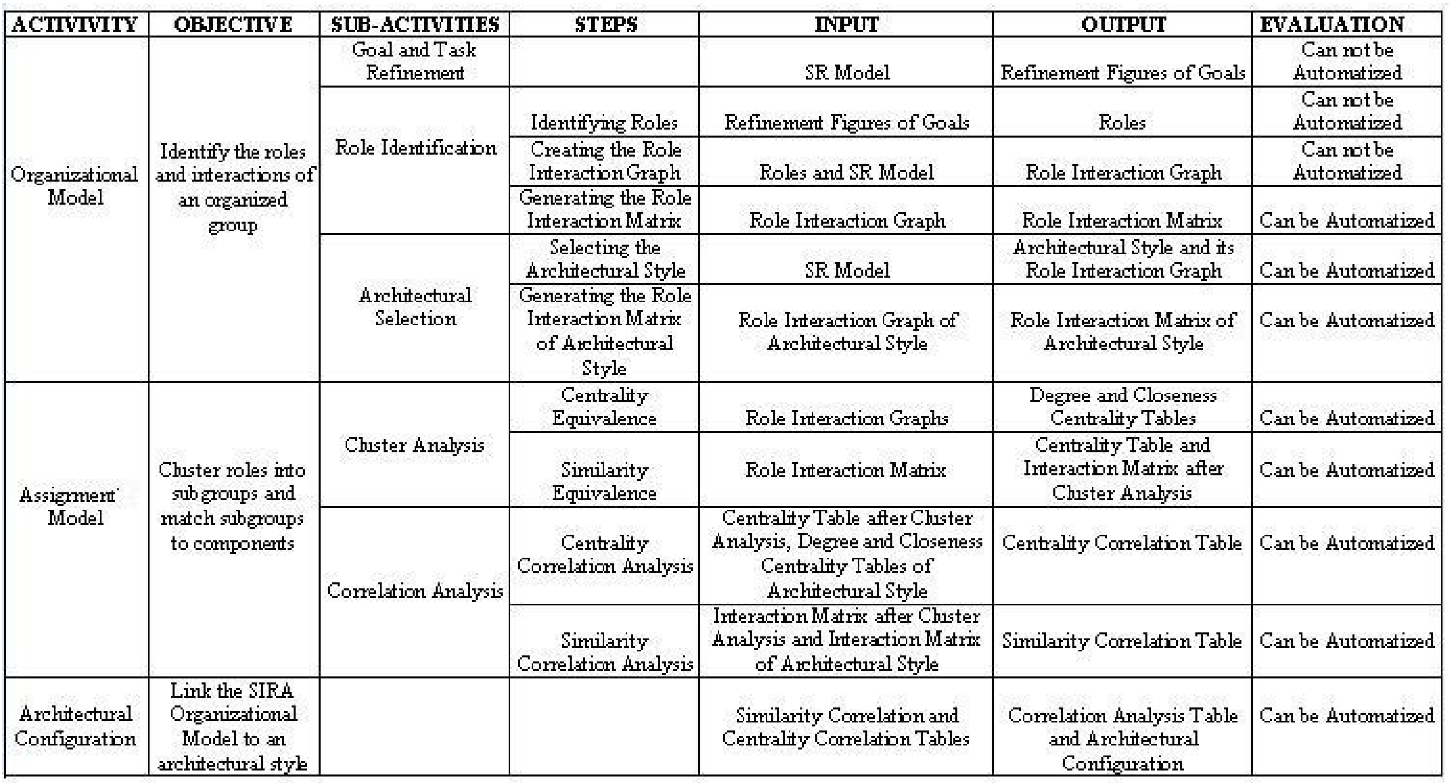
5. Conclusions and Future Work

Understanding the relationship between software requirements and architecture remain as a challenging software engineering problem. The SIRA Process focuses on a systematic way to assist the transition between requirements and architecture. As there is currently a lack of systematic guidelines to help this transition, the purpose of the SIRA Process is very relevant to software engineering. Its main idea is to systematically derive the architecture of a multi-agent system by analyzing its requirements based on organizational concepts.

In this work, the Conference Management case study was applied to each activity of the process in order to evaluate it and determine if it is possible to develop a CASE tool to support it automatically. We came to the following conclusions:

- It was not clear in [3] whether the activities of the process required any human interaction or could be performed by an algorithm. In this work, we came to the conclusion that *goal and task refinement* and *role identification* sub-activities are mainly based on human domain knowledge, experience and intuition and therefore, can not be automatized;
- As explained in chapter 4, the other sub-activities can be automatized as they consist mainly on calculations based on results from previous sub-activities.

To best express these conclusions, Table 18 summarizes all the activities, sub-activities and steps of the SIRA Process and identifies those that can be automatized.



ACTIVIVITY	OBJECTIVE	SUB-ACTIVITIES	STEPS	INPUT	OUTPUT	EVALUATION
Organizational Model	Identify the roles and interactions of an organized group	Goal and Task Refinement		SR Model	Refinement Figures of Goals	Can not be Automatized
		Role Identification	Identifying Roles	Refinement Figures of Goals	Roles	Can not be Automatized
			Creating the Role Interaction Graph	Roles and SR Model	Role Interaction Graph	Can not be Automatized
			Generating the Role Interaction Matrix	Role Interaction Graph	Role Interaction Matrix	Can be Automatized
		Architectural Selection	Selecting the Architectural Style	SR Model	Architectural Style and its Role Interaction Graph	Can be Automatized
			Generating the Role Interaction Matrix of Architectural Style	Role Interaction Graph of Architectural Style	Role Interaction Matrix of Architectural Style	Can be Automatized
Assignment Model	Cluster roles into subgroups and match subgroups to components	Cluster Analysis	Centrality Equivalence	Role Interaction Graphs	Degree and Closeness Centrality Tables	Can be Automatized
			Similarity Equivalence	Role Interaction Matrix	Centrality Table and Interaction Matrix after Cluster Analysis	Can be Automatized
		Correlation Analysis	Centrality Correlation Analysis	Centrality Table after Cluster Analysis, Degree and Closeness Centrality Tables of Architectural Style	Centrality Correlation Table	Can be Automatized
			Similarity Correlation Analysis	Interaction Matrix after Cluster Analysis and Interaction Matrix of Architectural Style	Similarity Correlation Table	Can be Automatized
Architectural Configuration	Link the SIRA Organizational Model to an architectural style			Similarity Correlation and Centrality Correlation Tables	Correlation Analysis Table and Architectural Configuration	Can be Automatized

Table 18. Evaluation table of the SIRA Process.

Also, this work aimed at identifying possible failures and at providing suggestions for improvement as well as providing resolutions to conflicting steps. We came to the following conclusions:

- Although the method used by the SIRA Process to identify how to group roles by comparing their in-degree and out-degree relations is very original, it might be important to consider semantic aspects of the model instead of purely structural analysis;
- Several of the mapping steps were not clearly defined in [3] and applying the Conference Management case study helped coming up with some solutions that were explained in chapter 4.

To illustrate these solutions, the following failures and corresponding solutions can be highlighted:

- In *Goal and Task Refinement* sub-activity, it is not clear to what extent and when goals and tasks should be refined. We came up with the following solution: the developer must refine goals and tasks until enough information is available for the next step of the process, the *Identifying Roles* step. In the Conference Management example, further refinement was not necessary as roles could be identified from the current refinement of goals;

- In *Role Identification* sub-activity, it is not clear how to identify the roles and how to create the role interaction graph. The solution given for the Conference Management example was based on domain knowledge and was explained in 4.2.2 and 4.2.2.2 sub-sections, but due to lack of space we do not detail it here;
- In *Cluster Analysis* sub-activity, it is not explained how to proceed when roles do not communicate with other roles and therefore, their farness values can not be calculated. Future work is required to deal with this issue as this work could not come up with a solution for that. In the Conference Management example, this missing information was not relevant and there was no problem in carrying on with the process;
- In *Correlation Analysis* sub-activity, it is not explained how to combine the similarity matrix based on in-degree and the similarity matrix based on out-degree to generate the similarity correlation table. The solution given for the Conference Management example was based on how the NFR framework deals with conflicts (negative correlations) and harmony (positive correlations) of non-functional requirements in software engineering and was explained in 4.3.2.2 sub-section, but due to lack of space we do not detail it here;
- In *Architectural Configuration* activity, once again, it is not explained how to combine the similarity correlation table and the centrality correlation table to generate the correlation analysis table. The solution given for the Conference Management example was

based on how the NFR framework deals with conflicts (negative correlations) and harmony (positive correlations) of non-functional requirements in software engineering and was explained in 4.4 sub-section, but due to lack of space we do not detail it here.

Some aspects of the SIRA Process require further work. Considering this, future work could take the following directions:

- Consider the semantic aspects of a model when grouping roles. To do so, it might be necessary to have even more human participation;
- Study *Goal and Task Refinement* and *Role Identification* sub-activities in order to determine the best way in which they should be carried out as they can not be automatized;
- Resolve the conflicting decisions that could not be resolved by this work, such as determining farness values when roles can not communicate with all other roles;
- Apply the SIRA Process in many different applications to obtain more information about its weaknesses and strengths;
- Develop tools to guide the *Architectural Selection*, *Cluster Analysis* and *Correlation Analysis* sub-activities, as well as the *Architectural Configuration* activity, as they can be automatized.

REFERENCES

[1].L. Bass, P. Clements, and R. Kazman, *Software Architecture in Practice*. 2nd edition. Addison-Wesley, 1998.

[2].L. Bass, P. Clements, and R. Kazman, *Software Architecture in Practice*. Boston, MA: Addison-Wesley, 2003.

[3].L. Bastos, *Integration of System Requirements and Multi-Agent Software Architecture*. 2005

[4].J. Castro, M. Kolp, J. Mylopoulos, *Towards Requirements Driven Information Systems Engineering: The Tropos Project*. 2002

[5].G. Kotonya, I. Sommerville, *Requirements Engineering: Processes and Techniques*.

[6].M. Galster, A. Eberlein, M. Moussavi, *Transition from Requirements to Architecture: A Review and Future Perspective.* In: Proc. Of the Seventh ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing. 2006

[7].A. v. Lamsweerde, "*Requirements Engineering in the Year 00: A Research Perspectiv*e" 22nd International Conference on Software Engineering, Limerick, Ireland, 2000.

[8].J.G.Hall, M.A.Jackson, R.C.Laney, B.A.Nuseibeh, and L.Rapanotti, "*Relating Software Requirements and Architectures using Problem Frames*", IEEE Joint International Requirements Engineering Conference (RE'02), Essen, Germany, 2002.

[9].A. v. Lamsweerde, "*From system goals to software architecture*", *in* Formal Methods For Software Architectures, vol. 2804, Lecture Notes in Computer *Science*, M. Bernardo and P. Inverardi, Eds., Springer Verlag, 2003.

[10].Loucopoulos, P. and Karakostas, V. "*System Requirements Engineering*". McGraw-Hill Book Company, 1995.

[11].Perry, D. E. and Wolf, A. L. "*Foundations for the Study of Software Architecture*". In: ACM SIGSOFT Software Engineering Notes, 17(4), pp. 40 – 52. 1992.

[12]. Garlan, D. "*Software Architecture*". Encyclopedia of Software Engineering, John Wiley & Sons, Inc.

[13]. Katia Sycara in: http://www.aaai.org/AITopics/html/multi.html#ks

[14]. 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), and pp. 63-83.

[15]. Stone, P. and Veloso, M.: "*Multiagent Systems: A Survey from a Machine Learning Perspective*." Computer Science Department, Carnegie Mellon University.

[16]. L. Chung, B. Nixon, E. Yu, J. Mylopoulos, *Non-Functional Requirements in Software Engineering*, Kluwer Publishing, Dordrecht, 2000.

[17]. 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 Multiagent Systems, Kluwer Academic Publishers, volume 8, Issue 3, pp.203-236.

[18]. Zambonelli, F., Jennings, N., Wooldridge, M.: *Developing Multiagent Systems: The Gaia Methodology*.

[19]. Pearson Correlation Coefficient in Microsoft Excel (2005).

[20]. Silva C., Araújo J., Moreira A., Castro J. ,Tedesco P. Alencar F., Ramos R., *Modeling Multi-Agent Systems using UML*.

[21]. 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.

[22]. 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.

[23]. Chung, L., Nixon, B., Yu, E and Mylopoulos, J.: "*Non-Functional Requirements in Software Engineering*". Springer, 1st edition (October 31, 1999), Volume 5, pp. 129-137.

SIGNATURES

Recife, 29 de Março de 2007.

Jaelson Freire Brelaz De Castro (Supervisor)

Turah Xavier de Almeida (Student)