Improving the Quality of Information Systems

An agent-oriented approach to domain modelling

Joseph Martin Kavuma

August 14, 2007

Abstract

The crucial role information systems play in the storage, retrieval and structuring, of organisational knowledge and information for manipulation, to support business processes and activities is as important as ever. The efficiency of the process to achieve this for any organisation, greatly impacts on the results of the organisation's efforts to remain competitive, making the quality of information systems critical in determining the effectiveness of the organisation as a whole.

This work embraces the agent paradigm in attempting to better the quality of information systems, to facilitate the modelling process in order to yield domain models of sufficient quality, before they are used in system development. Mainly aiming at reducing complexity in the modelling process, the work entails an investigation of the modelling process, agents, and agent abilities to support and facilitate dedicated communication, between collaborating components within a modelling environment.

Course:	Informatica/Computer Science
Research project:	An agent-oriented approach to domain modelling
University:	Radboud University Nijmegen
Faculty:	Faculty of Science
Research Institute:	Institute for Computing and Information Sciences (ICIS)
Research Group:	Information and Knowledge Systems
First Supervisor:	Dr.Theo van der Weide (Prof)
Second Supervisor:	Dr.Patrick Van Bommel (Ass.Prof)
Graduation Number:	
Date:	

Declaration:

I, **Joseph Martin Kavuma** hereby declare that this thesis is my own work and has never been used before in any institution.
It concludes the research for my masters degree and was carried out at the Information Retrieval and Information Systems research group, under the supervision of Prof.Dr.Theo van der Weide.

Signature .....

Date .....

Supervisor: **Prof.Dr.Theo van der Weide**

Signature .....

Date .....

Dedication

This work is dedicated to the Lord Jesus Christ.
"All the glory goes back to you GOD"

Acknowledgement

I humbly, with sincere gratitude thank the following people for their assistance and support throughout the study.

I would like to thank Prof.Dr.Theo van der Weide for his wise guidance and advice, and above-self support throughout the study.

I would like to thank Dr.Patrick van Bommel for his wise advice, and enthusiasm, that were inspirational in deciding to research about agents.

I would also like to thank Dr.Stijn Hoppwnbrouwers for his help during the research.

Lastly, i would like to thank Ms.Cezanne Klaasen for her unconditional support and understanding throughout the study

Contents

List of Figures

Management summary

Mainly, this research project aimed at identifying ways to improve the modelling process in order to improve the the quality of the models from the process. Most modelling techniques used emphasise predictability of the model, but it is very important that the model and the modelling process remain both predictive and adaptive.

Modelling and models are used by system builders to reduce complexity in the task of automating activities of a given domain. This normally involves down-scaling the problem area to a managable model, in order to study the behaviours of the domain, with the aim of achieving predictability which could help in simplifying the task.

Models are often faced with issues concerning their completeness, precision, formality and compactness in relation to the domain they represent. The modelling process on the other hand involves issues relating to the communication amongest its activities like translation, domain and language ambiguity. The modelling process involves four phases; elicitation, perception, abstraction and representation, that must be achieved collaboratively in a well coordinated environment, if the process and the model are to remain consistent with the domain they are specifying.

The work mainly emphasised the need to perceive the modelling process and the model as separate entities, that are faced with different but related problems, which if properly addressed could improve the quality of the models and hence that of the eventual system. The communication problems in the modelling process as by this work can be better addressed by the introduction of cooperating agents, to model the domain. This would simplify the task through sub-tasking and ensuring sufficient collaboration and coordination.

Chapter 1

Introduction

To remain competitive and profitable, organisations have had to employ and implement Information Systems, aimed at making use of both their external and internal knowledge. Organisational Information Systems are now big and complex systems, and in most cases serve several organisations who may not necessarily share the same group(s) of stakeholders.
The ever increasing sizes and importance of these Information Systems emphasises the current organisational dependency on them, creating a need very much not short of complexities. These complexities are inhabited mainly in these systems' ability (or inability), to make available required resources when they are needed, to the right users, with the most acceptable and achievable accuracy possible.
Luckily, this hardly pauses any problem in circumstances involving "simple" or small Information Systems, but rare are the cases when such systems are "simple" or small.

Organisations need to have control over their information before utilising it, and in order to realise, recognise and attain control over their information, organisations make use of the several information organising, structuring and modelling techniques.
For long, modelling and architecture have been employed by engineers to develop down scaled models of the real-world/domain problem. In this intimacy of the two processes, modelling is subsumed by architecture. The later is more broad and encompassing because it not only involves modelling and designing, but also development, implementation and testing the system under development.
This work, though may have used architecture to describe the proposed approach, it was mainly intended to address issues surrounding modelling, by examining the role played by the modelling process in the Information System's life-cycle, the problems surrounding this process, the methods used to model and the properties of modelling.

1.1 Models and System Development

The advancement of Information System into versatile systems like "webservices", has seen both practitioners and scholars dedicate considerable time into improving these systems' development, performance and efficiency. But unfortunately, still there are some persistent issues surrounding system quality, which argueably are said to stem from the systems' models and the modelling processes that yield them.
Efforts to address such quality issues, are credited for the recent development of several system development approaches, that have achieved remarkable success especially in trying not only to cut costs and increase productivity of development projects, but most importantly to achieve a development process closely associated with the model.
The most recent and popular of these approaches, being the generation of methods commonly referred to as the *"Agile development practices"*, that according to Martin Fowler include;
Extreme programming (XP)[eg. Test Driven Development (TDD)], Rational Unified Process (RUP), Model Driven Development (MDD), Scrum, Crystal, Lean Development, etc.

In Martin Fowler's view, Agile methods when compared to traditional engineering, are not only adaptive as opposed to predictive but they are people-oriented rather than process-oriented. Though Agile development methods are used, still, the system quality issues continue to be a serious problem in the system life-cycle and this is majorly due to the fact that these methods are hardly meant to address the modelling process' complexity, yet they enforce development that is dependant on its outputs (i.e the model). Though this research work did not involve an investigation of available developing practices like those mentioned earlier, their power to achieve good quality results in system development is undeniable.
Borrowing some inspiration and wisdom from the words of the Holy Bible;

> *"Unreliable as the legs of the lame, so is a proverb in the mouth of fools."* [Proverbs 26:7]'

The powerful development tools, can not eliminate the possibility of errors in their final results, because already the models they enforce, are of poor quality before they are used in these practices.
This work mainly intended to contribute to the already on-going research efforts to improve the modelling process, so that better quality models can be obtained, which if used in system development with appropriate practices, would lead to better quality information systems.

1.2 Modelling and Requirements Engineering

The first and most important step in the development of any Information System, is identifying and specifying the system requirements. The process of identifying, specifying and analysing system requirements is what is referred to as Requirement Engineering (RE).

Among the vital processes of RE, is information modelling [BN00]. Majorly, this work focused on modelling as a process, by attempting to determine possibilities of aiding its activities, all with an aim of producing better quality domain specifications (models). For a detailed view of RE the sources [MGC92] and [JEB02] are recommended.

Normally, the development of an Information System is triggered off by a motive/goal, which leads to certain objectives that are hoped to be fulfilled by the development of the system. Certain things, processes, activities or conditions are necessitated for the achievement of the objective(s) and these necessities equate to the requirements of the system.

> **Function:**
> A function is a set of effects (behaviours or properties) desired and utilised by an agent to achieve a goal, possessed by an entity in its environment.[JEB02]

1.2.1 Requirements

According to [MGC92], a requirement is that necessary function or characteristic of a system, or a quantifiable and verifiable behaviour that a system must possess, or a constrained environment within which a system can satisfy objectives, through solving a set of problems of a given organisation.

The [IEE83] glossary defines a requirement as;
"(1) A condition or capability needed by a user to solve a problem or achieve an objective; (2) A condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed documents; (3) A documented representation of a condition or capability as in (1) or (2)."

Requirements provide a clear readable, correct, unambiguous, specific and verifiable description of a given system's required behaviour.
Though as according to the IEEE, requirements are sub-typed into functional, performance, interface, design, implementation and physical requirements, this work only discussed the subject as function and non-functional requirements through brief descriptions of the two as follows.

1.2.2 Functional Requirements (FRs)

Functional requirements are those requirements that define, and are used to specify the system functionality. They specify specific behaviours of a system that define the internal workings of the system, supported by the Non-Functional Requirements.

1.2.3 Non-Functional Requirements (NFRs)

These are the requirements that define the ways and processes, that contribute to the systems' ability to achieve its functionality.
According to [JEB02], NFRs are system or device attributes not confined to a single portion of functionality, that contribute to overall product quality.
In this sense, attributes can include reliability, security, scalability, extensibility, manageability, interoperability, composability, survivability, evolvability, affordability, agility and understandability, and system wide properties like performance, security, availability, modificationability, adaptability, nomadicity, survivability and responsiveness.

Further [JEB02], contends that NFRs are constraints that restrict the types of solutions under consideration and that NFRs are very general in coverage, and considerably variable in desirability by different systems.
When viewed as constraints, [JEB02] refers to NFRs as both FRs and NFRs constraints categorised as interface, performance, operating, life-cycle, economic and political constraints.

1.3 Problem Description

1.3.1 Problem Area

As much as it is agreed upon that models are vital and essential in system building, models are as equally associated with several problems mainly stemming from complexities within the modelling process.
These complexities are present in vital aspects facilitating modelling like; communication, translation, etc and it is these, that require necessary and proper attention or addressing if the modelling process is to be enabled to produce better models, that are accurate domain representations.

The design and development of Information Systems, involves a lot of communication between several parties, who commonly are categorised as; domain experts (DE) and system analysts (SA).
With Information Systems now being applied not only in commercial businesses, but also in critical implementations like space navigation, climate and weather prediction, and defence systems, their importance has never been higher and it most importantly emphasises the need to improve their quality in general.
Researchers, in their contribution towards improving Information System's quality, have argued that the quality of the system critically depends on the quality of the model. But the information modelling process that yields the model, is quite a complex and tedious phase of the development process. And the complexity, greatly depends on a number of variables like the size of the system, the tools and practices employed during modelling, and the experience of the modeller.

1.3.2 Problem Statement

In most cases, the domain expert lacks the technical skills to perceive his/her domain as a system analysts would, and the reverse is true for the system analysts. But since the domain expert provides the requirements for the system analyst to model the domain, there is need for the two to clearly understand and inform each other through a question and answer dialogue. Failure to achieve this dialogue during modelling, was found to be the main cause of the development of poor quality systems despite the use of very efficient development process.

> **Statement of the problem:**
> "The conceptual model of a given domain, determines the quality of the Information System to be developed. But the two parties (DE and SA), involved in the process of specifying, defining and developing the conceptual model, lack enough support from the currently available modelling methods and practices to aid their communication and exchange of knowledge, in order to collaboratively develop models of sufficient quality."

1.4 Research Goal

Motivated by the conviction that poor quality models lead to development of poor quality systems, the study was done mainly aiming at finding ways to improve both the modelling process and the quality of the models from the process. The research was to involve a detailed study of the modelling process, in order to identify the phases or steps of the process that are problemantic and to also identify the complexities of the process.
The detailed study was to help in identifying;

- What modelling exactly means in the perspective of information modelling, the reasons why modelling takes place and the advantages of using models in system development.
- How modelling is used in system development.
- The different methods and techniques employed in information modelling.
- The problems and complexities surrounding the modelling process.

Through a detailed and focused investigation of the modelling process, complexities hindering sufficient communication during the conceptual model development phase, were to be identified. With some of the problems affecting the modelling process identified, stated and discussed, the research was to yield a suggestion, describing a way to support and facilitate the domain experts and the system analysts activities during modelling.

Initially, the use of agents to support modelling was a possible alternative but before their use as a solution, a clear understanding of their nature was only but necessary. The study in this persuit, was to investigate the vital properties and qualities, that make agents a better alternative to those already being used to support modelling. And efforts to achieve this, would entail a relatively descriptive discussion of a given agent platform or architecture, with the aim of showing exactly how modelling processes can be modelled into agents with roles or task to achieve, in a given environment.

Attempts to introduce agency in modelling by the study, were also to explore ways of letting the DE use a language familiar and known to him/her, to express their perception of their domain when specifying the domain properties and to investigate ways of facilitating learning between system analysts in situations were experience would improve the quality of the outputted model.

The research was also to involve a precise examination of the added value and advantages of employing agents into modelling. This would be evaluated by carefully investagating the suggested approach, with an aim of evaluating its efficiency and effectiveness regarding modelling, and the ability to sufficiently support collaboration between the domain experts and the system analysts.

1.5 Research question and Sub-questions

Following the problems of modelling defined in the problem area and the suggested use of agents as a means or part of the means to a solve these problems, a clear understanding of agents as well as modelling, would only serve to concretise the strength of the approach and hence was required.
The following questions were among the tools used to describe, achieve and develop an understanding of agents and modelling, and the way they could affect and aid each other in a bid to curb the problems of the modelling process.

1.5.1 Research question

In what ways can Agents be used to facilitate the process of information modelling?

Sub question 1

What issues of the modelling process need to be addressed in order to achieve models of better quality and what possible means could be used to address such issues?

Sub question 2

What qualities would a multi-Agent system possess that would be of relevant value in support of information modelling?

1.6 Research structure

The structure of the study has been defined and guided mainly by the two sub-questions given above, but it has involved three phases as described below;

1. In order to answer the first sub-question, the following have been done; Mainly by use of literature and interviews, a deeper understanding of the modelling process has been developed, yielding a clear definition and description of the process. Having achieved a clear description, an informed investigation of the modelling process has been carried out, mainly leading to identifying its problems and complexities, that were elementary inspirations in finding a solution to the persistent low quality models.

2. In order to answer the second sub-question, the following have done;
   The second sub-question embraced agents as a possible solution to the problem identified and in order to go about this question, literature review, interviews, experiments with agent development tools, comparisons and analysis', have been used to develop relevant knowledge concerning agent properties, abilities and the environmental characteristics of their implementations.

3. In order to answer the research question, the following have done;
   The two phases above, through analysis, interpretation and idea or concept integration have provided required elements, necessary to evaluate the possibility of modelling with agent support. This process has resulted into answering the research question, and giving its main outcome as, a description of a possible multi-agent system architecture that if developed and implemented, could provide the solution to the problems of the modelling process.

1.7 Research methodology

Literature study; Literature study, has been the main methodology used to both develop an understanding and knowledge about modelling and agents. And to also guide the analysis aimed at marrying the two in order to develop the proposed solution.
The literature sources used for the study have been included in the list of references, only with an exception of one important essay which is described next;
Martin Fowlers' essay *"The New Methodology"* as of 01-June-2007, accessible on his website (*"http://www.martinfowler.com"*), has been used to guide some sections of this research mainly due to his practical background and relevant insights concerning software development practices.
In this work, this source is sited as [Martin Fowler] in the parts where it has been referenced.

The main outcome and benefit of this methodology, was not only the acquired understanding of the main subjects of the study, but also a much needed contribution in developing questions for interviews and informed conclusions.

Interviewing; As one of the methods employed in the research, interviews or question guided discussions, have been carried out with researchers in the field of modelling, in order to broaden knowledge about; modelling as perceived by the business community and the modellers, formalisation of models, problems facing modelling, etc.
The following experts have been approached in this aspect;
Prof.Dr.Theo van der Weide of Radboud University, who in this work has been referenced as Theo
Dr.Stijn Hoppwnbrouwers of Radboud University, who in this work has been referenced as Stijn
Dr.Patrick van Bommel of Radboud University, who in this work is referenced as Patrick

Experimenting with tools; In addition to the above methods, experimentation with several development and modelling tools (JADE, Jdom, Netbeans, Eclipse) were carried out, to examine the possibilities of actually developing and implementing a multi-agent system in a JAVA platform.

These tools in respect to this study, have been used for system designing and describing of the proposed solution with an aim of simulating a possible implementation.

1.8 Structure of the thesis

Chapter two:- Modelling - focuses on the elaboration of the problem statement, by defining exactly what information modelling is, what important steps and concepts it involves, why it is very important in the life-cycle of an Information System, the problems it faces and what can be done or has been done to overcome these problems.

Chapter three:- Agents - introduces and describes the subject of agents, by elaborating on the definition and properties of an agent, distinguishing agents from ordinary software, describing agent abilities, tasks, behaviours and environment.

Chapter four:- Modelling with agents - examines the problems and complexities of the modelling process, elaborates on how agents could be used to overcome these problems and points out those modelling activities that could be better approached with agency.
The chapter further identifies the agents that could be involved in modelling, describes their tasks and a controlled language that the agents would use during communication.

Chapter five:- Multi-Agent modelling system structure - it is in this chapter, that the proposed multi-agent approach is materialised into a system design in form of; usecase, activity, class and deployment diagrams. This designs' quality is investigated for appropriateness to modelling, effectiveness and efficiency. And finally the chapter gives the added value of collaboration in modelling and the advantages of having multiple DE's and SA's.

Conclusion:- The conclusion gives brief summaries of each chapters in this work. It also briefly describes the conclusions reached as a result of this study, about the modelling process, agents and agency in modelling, as guided by the research questions.

Chapter 2

Modelling

Though literally common language defines a model, as a smaller or perharps simpler representation of a bigger real world phenomenon, technically there is hardly anything simple about a model.
Models are in reality complex structures depending on the domains they represent, and the complexity of any modelling process of a given phenomenon, varies with several factors that include the phenomenon's; complexity in nature, perceived context and size.
A real-world phenomenon to be represented formally, is defined as a domain or as the *Universe of Discourse* (UoD) and during modelling, it is down-scaled to a representation developed with the aid and guidance of a definite set of requirements.

A Domain

Usually a domain is defined as a given problem area of interest, but to a modeller, faced with a task to model a given domain by use of a given modelling language L, any unique problem of interest has a unique evolving domain D consisting of all possible statements that would be correct and relevant for solving the problem at hand.[OIL94]

A Model

The model of a given domain D, is a union of the set of explicitly made statements Me and the set of statements that could be from Me in accordance to the modelling language's deduction rules.[OIL94]

2.1 Model Quality

Model quality remains a contiguous item as far as system quality is concerned, according to research work like [OIL94], and others that regard quality as one of the most important model features. Researchers like Terry Halpin, Theo, Stijn and Patrick, argue that the quality of the conceptual model is very critical to the quality of the eventual information system developed.
In accordance to [OIL94], the quality of the model is affected majorly by;

2.1.1 Language-domain appropriateness

This entails exactly how the modelling language, by making and providing statements needed in the domain, appropriately maps to the domain.
A modelling language that does not appropriately map to the domain, guarantees misrepresentation of the domain by the model and thus leads to unsatisfactory and unacceptable results in the eventual developed system.

2.1.2 Language-audience appropriateness

This is said to affect the model's quality based on how the audience perceives the modelling language.
It entails the limit of the audience's ability to learn, understand and use (by expressing themselves and interpreting) the language. In the perspectives of some experts like Stijn, unsatisfactory systems could be eliminated through letting the domain experts do the modelling or at least having the system analysts engineer requirements in a less limited and more natural language clearly understood by the domain expert.
This factor, in agreement with these researchers should guarantee, that the modelling language of choice will not permit use of unfamiliar sentences to the DE by the SA.

2.1.3 Audience-domain appropriateness:

This factor entails the level of formality or ability to attain formality of the domain by the audience.
Some researchers like Theo, argue and contend that the more formal the model is, the easier it is to control and integrate domain changes into its structure.
With the consideration of the above factors, the framework given in [OIL94], categorises model quality properties into; Syntactic, Semantic and Programatic quality properties. Which properties are said to be achieved through modelling activities by a process that yields qualities which indeed are the goals of the modelling process.

2.2 Model properties

The subject of "Model properties" in this section, is approached with a discussion based on the understanding attained from analysing and reviewing information and ideas embedded in the two literature sources [SB] and [OIL94] about model properties.
In [OIL94], a framework that describes the modelling process is given, as a process motivated by a goal(s). [SB]'s work seemed to concur with [OIL94], when they contended that a specific goal is held in mind always when modelling information. They further emphasised, that model-required properties, say its' form, information type, precision, etc are all determined by the goal held in mind.
Below are some of the most often required properties discussed as according to [SB] in relation to the framework given in [OIL94]'s work.

2.2.1 Completeness

Completeness is a must-have model requirement, thus a complete description of relevant aspects of the domain is a compulsory requirement when modelling.
In comparison to [OIL94]'s framework, this is a semantic quality, subject to the feasibility factor, that determines a model's formal semantics' modifiability property. To make sure that a complete model description is attained, activities like statement insertion and deletion are undertaken in an effort to include the left out, and eliminate the unrequited or irrelevant information.

2.2.2 Consistency

Like completeness, consistency too is of the sematic quality category and is subject to a given attainable level of feasibility.
It is a property achieved through consistency checking and is motivated by the goal to attain feasibility, validity and completeness.
In accordance to their work, [SB] contend that a model is required to be consistent not only internally but also with the domain.

2.2.3 Precision

Precisions emphasises the need to have the model give the most accurate representation of the domain possible. According to [SB], precision defines the requirement for the model to describe the domain to a certain abstraction level with enough accuracy.
On comparison with [OIL94], the modelling activity used to achieve precision is filtering and inspection, which could mean that precision is subsumed by expressive economy or structuredness or both. The main goal of this activity is to attain feasible comprehension, a pragmatic quality.

2.2.4 Compactness

As according to [SB], it is stated that a model is a "generative device" with the ability to efficiently describe all relevant information in the domain. The model should be intentional as opposed to extensional.
Modelling activities such as visualisation, explanation and simulation as given

in [OIL94], are aids to the process of efficient description given by [SB]. The "Executability" model property, would logically require some definite definition, to be realised. The main goal driving efforts to determine model compactness, is to be able to attain some comprehension of the domain.

2.2.5 Formality

This probably is the most agreed upon property by both sources. [SB] contend that given the fact that a model is a representation, then clear syntax and semantics are no options but rather required necessities of the process.
In almost complete agreement, [OIL94]'s framework depict model formality as formal syntax and semantics properties, achieved through syntax checking and consistency checking respectively.
Formality is a property determined by the goal to attain syntactic correctness and feasible validity.

2.3 Modelling Applications

As a fundamental activity in RE, modelling is the process of developing abstract descriptions amenable to interpretation [BN00]. Physically, modelling aides efforts aimed at achieving control during the construction of a phenomenon like a house or building, car, ship etc. This allows predictability to the builders, which is very vital in cost, time and task estimation. According to [BN00]'s categories of RE modelling, models are the way through which abstract descriptions of enterprises and organisational data, behaviours, domain, Non-Functional Requirements, etc are constructed.
In this section, modelling was discussed as it is used in the RE process while modelling information of a given organisation.

2.3.1 Enterprise Modelling

The Enterprise modelling process leads to understanding the organisation's structure through analysing and modelling of business rules affecting its operation, constituent members and their goals, tasks and responsibility, and the organisation's needed, generated and manipulated data.
The modelling of the organisation goals is useful in RE, in a sense that it allows the transformation of high-level business goals into operationalised requirements through repeatedly refinement.

2.3.2 Data Modelling

Data modelling defines the several activities and processes, involved or undertaken when exploring and formally representing data oriented structures. Data models usually are categorised as conceptual, logical or physical and are developed through the following processes;

Conceptual Data Modelling: This yields Conceptual models (or domain models), that are used to formally represent a domain's concepts.
They are usually developed before the logical data flow is determined and modelled.

Logical Data Modelling: Logical data modelling outputs models used to represent the functionality of domain concepts as required by their relationships. By use of identified domain constraints, logical models are used to abstract the the logical flow of data as directed by the concepts of the domain.

Physical Data Modelling: Physical data modelling yields models that are used as aids to the process of designing the internal database schema and depict data tables, columns of the tables and the table's relationships.
They basically describe the physical data storage setup.

Since careful decision making is needed about the information represented in the system and how it corresponds to the domain, data modelling in RE provides opportunities to enable understanding, manipulation and management of information [BN00]

2.3.3 Behavioural Modelling

Behavioural modelling deals with abstracting existing and required requirements, mainly realised through modelling dynamic or functional behaviours of a given domain's stakeholders.
It involves modelling and analysing the current physical system in order to determine the current logical system before building the new logical system.[BN00]

2.3.4 Modelling Non-Functional Requirements

Arguably, NFR's are difficult to express in a measurable way and thus more difficult to analyse.
Also referred to as quality requirements, NFR's tend to be properties of a system as a whole, making their verification impossible regarding individual system components.
But never the less, NFR's models are used to construct abstract descriptions of system properties like safety, security, reliability, usability, etc.

2.4 The Modelling Process Overview

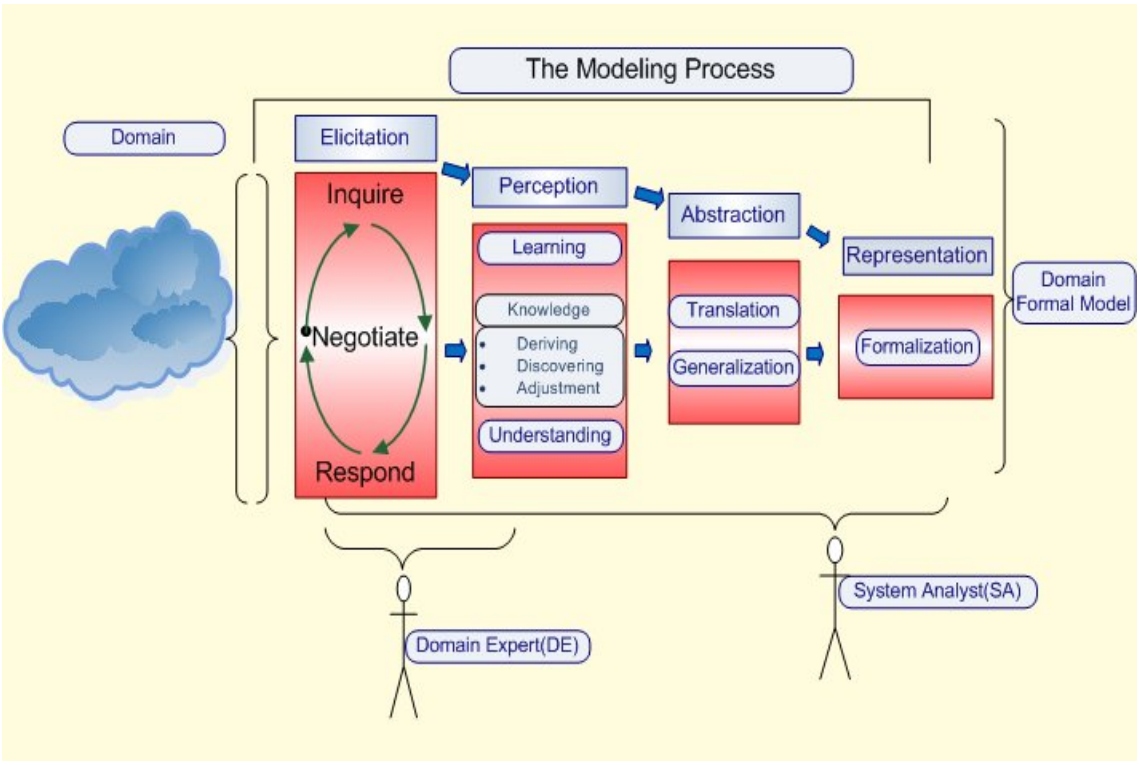


Figure 2.1: Modelling Process Overview

2.5 Conceptual/Domain Modelling

Having introduced modelling and given some of the applications where its is employed, this sectioned has been used to report on a detailed study of domain modelling. It gives a relatively detailed description of some of the known definitions of domain modelling, the phases or stages it involves and the roles involved in these phases.

2.5.1 Known definitions

Modelling is a phase intended at transforming an informal specification into a formal specification, achieved by matching sentence structure on modelling concepts, after discovering modelling concepts and their relationships [PF05].
The sentences' grammatical analysis involves the reduction of syntactical variation to a syntactical normal form. This is followed by abstraction from the sentence structures, hence defining the normal form specification. The sentence structures are then matched onto concepts of the modelling technique used.

The normal form specification, through its elementary and structural sentences, provides an effective way of deriving the underlying conceptual model. But due to some missing elements in the informal specification, like disoriented mutual order of sentences, the sentence structure is normally poor hence creating a need for order reconstruction

1. According to [Hal], during modelling, information is verbalised in terms of elementary facts. These elementary facts, are instances abstracted to elementary fact types that lead to a complete conceptual schema when constraints and derivation rules are added to them.

2. Modelling according to [BN00], is a fundamental R.E activity that involves the construction of abstract descriptions that are amenable to interpretation.

3. As described in [SB], given concepts and relations existent in some domain of interest, Domain modelling is a specific goal driven process done by elicitation of knowledge from a domain expert, which knowledge is then abstracted and represented by use of a modelling language.

2.5.2 Key phases of the conceptual modelling process

The above mentioned definitions have some variations, but mostly approach the subject of information modelling with some relative uniformity in various aspects like elicitation, representation and abstraction.

Abstraction: Literally, abstraction is the action of extracting or removing something from another. It is a process of reducing and factoring out, of irrelevant details from a given domains' informal specification while optimumly leaving behind particular concepts of interest.
Abstraction is the hiding of details while exposing only essential features of a particular concept or object.

Elicitation: This is described as obtaining, producing or "invoking" information or a reaction from a given domain expert, by a system analyst.

Representation: Generally, representation implies multiple presentation of a given real phenomenon by several symbols. With this in mind, representation is used in reference to a hypothetical internal cognitive symbol, that stands for external reality.
In his work, [DM] defines representation as; a formal system for making explicit certain entities or types of information, together with a specification of how the system does this.

Perception: In ordinary language, perception is said to be the sensory way to notice things. It is the ability to understand the true nature of something or an opinion/idea, got/developed as a result of ones' understanding of a given something.
In cognition Science, it is held that perception is that process of sensory information acquisition, interpretation, selection and organisation.

The key words defined above describe four compulsory inter-dependant mechanisms of the modelling process, and with these, a given UoD is defined, described and finally modeled or represented.
As by [PF05], it is contended that the modelling process involves two experts; the Domain Expert and System Analyst, each specialised in their respective fields but work together to produce a model (formal representation) of the UoD.

Both experts though work on the same domain, they realistically perceive the domain distinctively different from each other, and their perception consequently limits their abstraction hence subsequently affecting their representation of the domain.
But critical to note, is that though the system analyst's perception is much different from the domain experts' (for instance it is defined in terms of entities, objects, relationships, constraints, etc while the domain experts' is in terms of customers, client, functions, rules, etc.), it critically depends on the domain experts' informal representation and perception of the domain.
Incompleteness in the domain's informal specification by the DE, prompts questions from the SA.
This process that involves questions and answers, clarifications, acceptances and rejections, is the elicitation phase, and it yields a general description of the domain in the form of a "Requirements List" (RL).

The SA uses the RL to develop and update his/her perception of the domain, which perception guides the abstraction of relevant concepts that are eventually used to formally represent the given UoD.

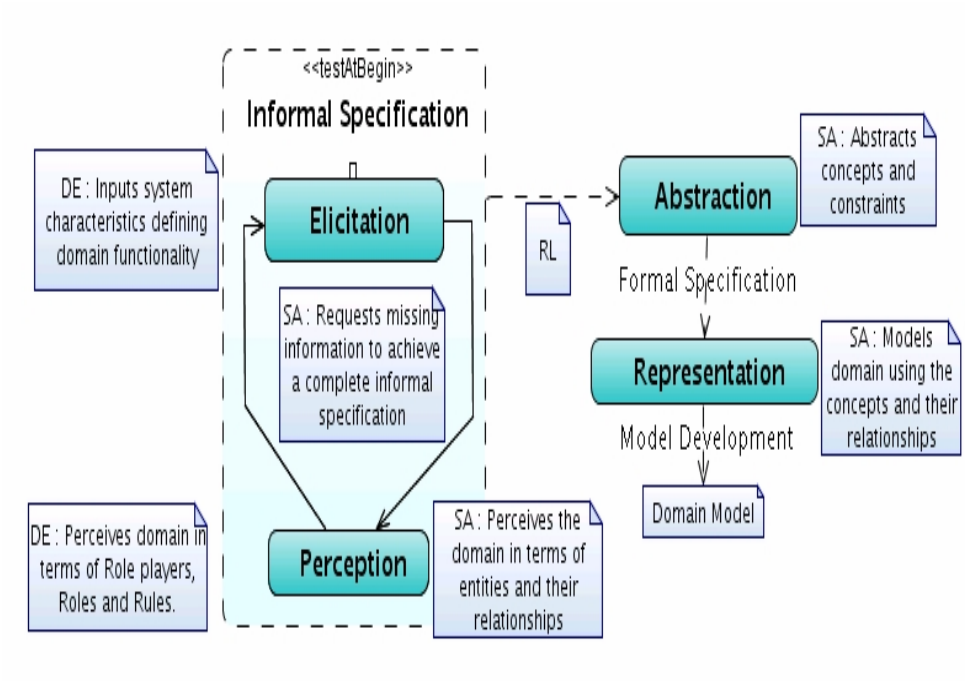


Figure 2.2: Roles in Modelling Process

Note:
In many cases perception development and update happens also on the side of DE when new information is made visible to him/her during the elicitation process.
Abstraction could happen concurrently with elicitation.

Based on the above discussion, modelling as by this work, was defined as;

The process of formally representing abstracted knowledge obtained through elicitation from a knowledgable perception of a given UoD.

2.6 Modelling Techniques/Methods

Practically domain modelling is a subject approached through several methods or techniques like Data Flow Diagrams, System Flowcharts, Workflow modelling, Structured charts, Object Role Modelling, Object Oriented Modelling and Entity Relationship modelling.
This section has been used to give relatively briefly descriptions of three of the methods; Object Role Modelling, Object Oriented Modeling, and Entity Relationship, mainly due to their popular applicability in information modelling.

2.6.1 ORM

In comparison to Entity Relationship (ER) and Object Oriented (OO) modelling methods, Object Role Modelling (ORM) is more expressive, mainly due to its role-based notation, that enables the specification of a wide variety of constraints. Besides, the object types used in the method reveal the sematic domains that bind a schema together, allowing formulation of conceptual queries in terms of schema paths.
According to [Ter98], Object Oriented models often duplicate information by wrapping facts up into pairs of inverse attributes, in different objects and have weak support for constraints.

On the other hand, though ER models have the advantage of giving summarised models as opposed to the detailed ORM models, they are said to be of use once the design process is finished because they are less suitable for formulating, transforming or evolving a design.[Hal99]
ER diagrams are far from natural languages, can not be populated with fact instances, require complex design choices about attributes, lack expressibility and simplicity of a role-based notation for constraints and hide information about the domain.

2.6.2 ER

Originally, the ER model was proposed as away to unify network and relational database views.[Chan76]
From the ER conceptual data model's perspective, the world is viewed as entities and relationships that are associations between entities.
Entities usually, are recognisable concrete or abstract concepts or the data objects about which information is collected.
The individual occurrence of an entity is called an instance, and entities are classified as independent or strong when they rely not on another for identification, or dependent/weak if they do.
Further, entities could be typed as associative if used in a reconciliation of a many-to-many relationship. Where relationships are classified in terms of degree, connectivity, cardinality and existence, representing an association between two or more entities.
The attributes exhibited by entities are either identifiers or keys if they uniquely identify an instance of an entity, or descriptors if they describe a non-unique characteristic of an entity's instance.
Other than its cross-database management system capabilities, simplicity and understandability, the ER-Model can easily be translated to the relational model in which Entities and instances correspond to tables and rows respectively.

2.6.3 OOM

Mostly used with programming languages like Java, C++, Visual Basic.NET, etc, the OO-model is contained and realised in notations like; Rumbaugh, Booch, Coud/Yourden and most popularly the Unified Modelling Language(UML).
Like the ER-Model, the OO-Model is based on a collection of objects with the view that the real world is made of objects of unique identity that provide

services to each other. The objects contain attributes used to represent their state.
Objects are grouped into classes in accordance to their similarities in attributes or services. And these classes form an abstraction hierarchy by use of the **"is_a"** relationship.
The operations, that all objects in a class can do when invoked (message passing) by other objects, are referred to as methods, and they include services and function.
Relationships in this model can be of the form; **"is_a"** representing classification relations, **"part_of"** representing assembly relationships and **"associations"** which represent relationships between classes.
The objects are perceived to be stable though their functions and services may change, which is widely perceived as a stability and maintainability advantage.

2.7 Why model?

There are several reasons as to why real-world items are down-scaled to model representations. These reasons can range from the need for prediction when a new system is being proposed, to the need to attain control when an existing system is being analysed.
In this section, the reasons leading to modelling are briefly discussed as follows;

2.7.1 Reasons for modelling

Most importantly, modelling guides and supports decision making about the information the envisioned system will represent.

Most Information Systems are developed in projects with continuously changing member structures. And usually, real-life system development projects are broken down into sub-projects, intended to collectively develop the different system components procedurally. In such situations, their are risks due to the different people in the different sub-projects.
But by first generating models of the anticipated system that give sufficient details of its' components, relevant information can be derived that if used can help to avoid or contain these risks and also to prevent inconsistencies. Models in this perspective, help to achieve continuity in the system development process and domain representation.

Generally, the use of modelling provides a means to manage, control and reduce complexities in the domain, and that of development process aimed at automating the domain.
Modelling facilitates the RE process by providing means of simulating the requirements, their relationships, attributes and roles. With development paradigms like Model Driven Development, the importance of models is realised beyond requirement specification, to requirement integration, implementation and realisation.
The models provide control over system requirements and to a certain level, control over system development. For reasons like complexity reduction, attaining predictability, efficiency, accuracy of the system development process, modelling

is always part of system building, making it not only essential but also of vital importance in quite a lot of aspects.

2.7.2 Results of modelling

In accordance to [GC04], modelling should not slow down the system development process but instead facilitate it.
A successful modelling process always leads to structuring of the system development process, which could involve steps that are followed in a predictable and controlled manner.

- As according to [BN00], domain models permit detailed reasoning concerning assumptions about the domain and provide opportunities for requirement reuse within a domain.
- According to [GC04], modelling allows developers to create and communicate designs before additional resources are committed.
- Modelling also allows the linking of the design to the requirements which ensures that the right system is being developed.
- Modelling also allows practising iterative development which caters to quick and frequent changes.

2.7.3 Conclusions

As the business organisations continue to grow in size, so do their information system that now have diversified in terms of functionality, scope, stakeholders, etc, in order to cope with the increasing need for coordinated information sharing within the growing organisations.
The bigger these organisations get, the more need and reason to have more control over their information and knowledge. This is vital in coordinating business processes that ensure that they remain competitive in the new horizons that emerge. Unfortunately, the spreading of these organisations also increases the complexity of their domains they define, mainly because it constrains information communication within the organisation.

It is such problems, that make modelling very important when attempting to develop solutions to them, because it provides a means of control over the domain, by breaking the complexity down to managable tasks.
The quality of the models is vital, as it determines the quality of the systems developed to share information within organisations, but itself depends on how the modelling language, audience and the domain align with each other, which have to be chosen with extra care in order to achieve the properties that define a model of good quality.

The complexity in modelling increases with the complexity in the domain, calling for a more structured and systemantic phased modelling approach.
It is common for current modelling practices to yield models that lead to development of systems that are completely what the user did not want, which signifies a gap between the DE and the SA when modelling a given domain.
In order to be able to eliminate problems of this sort, and to improve both the quality of the model and system developed, there is need to improve the modelling process by changing the approach to the modelling process.

The gap between the DE and SA signifies absence of sufficient collaboration and cooperation between them, so by improving the communication in the modelling process, the process could be made less complex irrespective of the domain size.

Chapter 3

Agents

Due to the fact that agency could be used in reference to several things, it is arguably difficulty to find a suitable definition correctly befitting the term "agent" just as it is for intelligence [Goo93].
Agents are said to be natural when viewed in the form of human beings, organised insect and animal societies.

But though difficult it is, to determine both the suitable definition of an agent and what should be called an agent, all items considered to be agents exhibit certain properties, especially in behaviour, that are uniform to all.
This study investigated, discussed and described software agents, with aim of determining how they could be used in support of the modelling process.

Agents were found to communicate with their peers by use of a common language, and could also be viewed as large entities, exhibiting some sort of persistent control.
Software agents are developed with inspiration from human and other natural agents. And like human agents, they have a desire to interact and interoperate [TF], which are the two requirements that together define agent communication. But for software agents to communicate, a common language, a common understanding of the knowledge exchanged and the ability to implement the theory and concepts of the two, are fundamental.
Effective interaction, is considered to be the exchange of mutually understood information and knowledge.

3.1 What is an Agent?

According to [Goo93], the term agent denotes a hardware or software based computer system exhibiting autonomy, social ability, reactivity and pro-activeness properties, that is conceptualised or implemented using concepts that are human like in nature.
An agent is also defined as any entity created with a goal of accomplishing a task(s), and is distinguishable from other agents in the same environment, in order to enable its substitution with another when comparing their performance.

Agents as according to [EH], are a software architecture, described as an abstraction of concepts like methods, functions and objects.
But the agency quality in agent-oriented systems, elevates agents above just abstractions of concepts, given the abilities it avails to these systems reflected in properties like autonomy, behaviours, etc. That enable agents to accomplish tasks on behalf of their creators, a property not exhibited by usual objects.
Technically, agents are implemented as threads executed with specified scheduling priorities. They exercise agency by terminating or starting other agents, reacting and responding to changes in the environment, and communicating and socialising with other agents in the environment, but all this achieved with minimal or no human intervation.

Like their human counterparts though to a lower level, software agents exhibit an ability to learn from their environment and this determines their level of intelligence. When agent learning is proceeded through trial-and-error, the capability to introspect and analyse behaviour and success is implied, but if proceeded by example and generalisation the capacity to abstract and generalise is implied.

Current literature about the subject of agents has seen them categorised as intelligent, autonomous, distributed, mobile, fuzzy and multi-agent system. But all these categories root from certain properties, some of which are uniform to all agents and others applicable to certain agents, that enable them achieve their tasks.
Agent nature, description, development, and implementation consists of concepts like persistence, autonomy, social ability, reactivity, environment, tasks or goals etc. But all these can be classified as properties, behaviours, tasks/goals and environment.

Though rightly contended by several research work like [EH] that agents are still a growing Computer Science research field, some milestones have been reached in this aspect, leading to some of the current agent standardisations, most notably the Foundation for Intelligent Physical Agents (FIPA) standard.
As earlier mentioned, a uniform accepted definition of the term "Agent" as regards software agents is yet to be agreed upon, but some agreement has been reached regarding agent properties. These properties so far are the basis of determining if a given software system possesses agency in its nature and thus qualifies to be an agent system.
As given in the work [EH], Agent properties are categorised as;

Common Properties: Which are described as properties valid for all Agents

Classifying Properties: Which are contended to be properties used for agent categorisation.

The section that follows, has been used to discuss agent properties, based on an analysis and comparison of the definition and description of agent properties as given by [EH] and [MW95].

3.2 Agent Properties

Agents exhibit certain distinctive properties that help identify them as agents, in this section these properties have been explored through a brief description and objectification.

3.2.1 Autonomy

The agent property of autonomy, defines agents as systems with the ability to act without foreign intervention, but with some control over their actions and internal state [MW95]. This implies that agents practice self instruction to perform actions, and are able to go through several process states in the boundaries of their expected or defined nature of actions.
The autonomy property of agents, defines the independency and decoupled execution of tasks undertaken by agents. Which is not only determined and influenced by available resources to the agent, but also depends on its cooperation with the others [EH].
Important to notice from both [MW95] and [EH], is the agent ability to perform tasks independently without foreign help either from humans or other agents. Like for instance an agent that starts and kills other agents depending on the changes in process states as opposed to foreign instructional intervation.
The term "Autonomous Agents" roots from this property, and it embodies the assumption of agents as being self-contained, with the capability to make informed independent decisions that are followed by taking appropriate actions intended to satisfy goals in a given environment.

Mobility

In relation to autonomy, is the agents ability to move around an electronic network, which movement is self initiated without any foreign assistance. Though normally mobility is given as an independent property from autonomy, as perceived by this work, it is an action performed as a result of an informed decision of an autonomous agent.
It involves the relocation of code together with its execution state from one processor to another in order to achieve a specified goal(s).

3.2.2 Reactivity

The notion of reactivity, implies that agents are aware of their environment, so they act responsively in a timely fashion to changes within their environment. Environment is a diversity of some sort in this case, because it could mean physical conditional atmosphere or an activity atmosphere. Where the first, could be a real world physical phenomenon like temprature change or appearance of light, etc. and the later could mean any form of interaction atmosphere, that involves a software agent and other agents (humans, other software agents, etc.).

Three of common properties given in [EH], seem to describe the reactive nature of agents; State and Reflexivity, locality and structural openness.
Where the state and reflexivity reflect a given agents' ability to interpret its environment in terms of its internal knowledge structure, the locality describes the

essential requirement for the agent to realise and adapt to new knowledge within its current given location, and structural openness defines agents as components with flexible structures that can change in terms of behaviour or relations with other agents as time and other factors in the environment change.

3.2.3 Social Ability

This is the quality of interoperatability and interaction, that allows the communication of information messages between agents in a given environment. This requires the interacting agents to use an agent communication language common to both, aid the understanding and the development of the meaning of the exchanged messages of information. This property enables agents to learn from each other and also to notice the presence of each other in the same environment.

Region Affiliation, which according to [EH] is described as the notion of agent systems belonging to regions as specified by their creators and locality, seems to generally describe the agent property of social ability. The knowledge agents possess about their tasks and the resourcess necessary to accomplish these tasks, compels them to a social environment that though individual agent exhibit independency that allows them self-instructing abilities, they also depend on other agents for resources (inputs and outputs) within the environment.
This requires agents to communicate by means of a common language usually or each agent may possess a meta-knowledge to translate inputs to a format it understands or expects.

To enable communication between agents in a social environment, normally available agents with resources to offer, advertise their resources to potential users and by so doing, agents get to know who is in their region, what they have to offer and probably what they may require. Whichever way this may be viewed, it reflects a learning and teaching social environment.

In the classification of properties given by [EH], Communication Behaviour, Negotiation Ability, Delegation Ability and learning Ability of agents describe the idealogy in an agent social environment. These reflect the agent decision-making process, while emphasising social ability in order not to breach the social requirements and expectation of the environment.

3.2.4 Pro-Activeness

This would also be correctly referred to as intelligent reactivity, which means that though agents react responsively to environmental changes, their reactions are goal directed.
This property emphasises the fact that all agent actions are goal oriented, a notion that demands intelligence on the part of agents. Where intelligence according to [EH], is the level of evaluation and learning behaviour of a given agent. It is further contended in [EH], that its through these tasks/goals that agents aquire new knowledge.
This property implies that agents do not react to just any environmental change, but instead to changes that require their action. To ensure that this is achieved,

agents are created in such a way that they know which changes to act to or through their learning process, get to know which changes require their response and action.
The goals that drive agent actions, are what is encompassed in the Role and Service Capacity, which represent functionality for agent task execution and is sub-divided into action type and task type as according to [EH].

Benevolence

In connection to agents being pro-active, is the property of benevolence which is the assumption that agents act to expections, avoiding any possibility of conflicting goals.

3.2.5 Rationality

Under this property, agents are to act in order to achieve their goals or tasks but not to prevent the achievement of these goals. This makes agents almost as obedient as humans, who are expected to be productive and responsible by their society laws, ethics. and codes of conduct.

To achieve rationality, certain properties like Correlation which entails synchronous or asynchronous agent activities, Resource limitation which bounds the agent actions to available resources and to a certain existent Re-Usability, are vital.
Since agents possess some intelligence, they are expected to act as expected but not to over or under react, and by so doing, agent actions lead to goal achievement rather than prevention.

Veracity

This is a property that states that though agents can be autonomous and mobile, all these actions must be under control to ensure absence of intentional information miscommunication.

3.3 Characteristic Nature of Agents

3.3.1 Agent Tasks

It is widely argued that all agents are task/goal oriented, implying that all agents are created with a purpose of achieving a given goal. A task describes what the agent is to achieve in the specified environment [Goo93]. The tasks vary in description and could be; sending a message, coordinating several other agents, terminating other agents, providing information to other agents, etc.
Task specification methods as according to [Goo93], should be able to encode each type of tasks, allow for different types/ways to evaluate agent performace, which should include simple binary and utility evaluations but in line with the environment.

3.3.2 Agent Behaviour

The properties mentioned before, define and influence certain agent behaviours when accomplishing their designated tasks in their specific environments. Goodwin in the source [Goo93], insists on the definition of an agent basing on the tasks it is to accomplish and the environment it is situated in.

Agent behaviours, are agent actions through which agent properties are realised, implemented, applied and utilised in order to enable goal/task achievement and accomplishment in a given environment.
Agent behaviours are not properties, but rather they are a process by which agent properties are transformed into purposeful actions. Agents are obliged to behave in accordance to other agents, their environment (Region and locality) and their tasks or goals. When an agent is chosen for a given task , its properties are defined into reactive actions to appropriate changes, and this translates into its behaviours thus behaviours could rightly be perceived as applied properties.

For Example: A stock monitoring agent with an autonomous property may exhibit a behaviour to automatically make a decision when stock prices fall to a certain level.

Behaviours are critically dependant on the environment and the the tasks the agent is to perform. A given agent can only behave as allowed by the environment.
As according to description of the resource limitation property and correlation given in [EH], the tasks achievable by an agent through certain behaviours, also largely depends on what the environment renders possible.
Agent behaviours include collaboration, communication, mobility, etc.

3.3.3 Agent Environment

The environment, is where an agent operates from (accomplishes its tasks), and it is made up of states subject to change with time, in response to interaction of the agent. An environment is said to be deterministic if it is exactly predictable basing on a known exact state, the interaction with the agent and the forces in its environment. A non-deterministic environment only allows prediction of a set of possible future states. Most important to note is that, when modelling an agent environment, it is very crucial to choose a model able to cater for both deterministic and non-deterministic state changes.

In the work [EH], a brief discussion of the agent Life cycle is given, which is a property normally given little or inconsiderable attention.
The Life Cycle of software agents is as limited as that of other agents. According to [EH], the agent creator is referred to as the "authority" and the environment is approximated to a name space.
An environment is quite broader than a name space though, and is made up almost of everything that defines and supports agent life cycle. It defines the agents involved, their physical geographical locations, their behaviours and even their tasks.

Normally, it is characterised by interfaces that enable behaviours like coordination, mobility, communication and cooperation. Which too facilitate activities like learning and teaching amongest agents within the environment.
Sometimes the environment is definite and this is when it is confined to a specific network, but it can also be infinite and this is when agents can be mobile when executing.
By using of meta-knowledge, appropriate protocols and other tools, agent environments can both be expanded and shrieked as required. But important to note, is that the larger the environment gets, the more complex it gets to manage and maintain.

3.4 The JADE Framework

The Java Agent Development Environment, is an agent framework developed by Telecom Italia in collaboration with several other companies.
JADE is developed and implemented in a java runtime environment, with the capability to run several agents over a network, irrespective of whether the agents are run on machines with different operating systems.
This section has been used to give a cautious description of the JADE framework, based on the [FB05] source, mainly with an aim of highlighting the most important elements vital to the frameworks' way of running and supporting multi-agent systems.

3.4.1 JADE Platforms

Agents in the JADE framework live in a container that provides a runtime support to agent execution.
A container is a machine running and hosting one or several JADE agents, and a collection of containers in a networked domain makes a platform.
Each JADE platform consists of at least one container referred to as the "Main-Container", that is responsible for running, stopping and coordinating other containers in the platform.
JADE, in compliance with the Foundation of Intelligent Physical Agents (FIPA) standard,implements an Agent Management System (AMS), a Directory Facilitator (DF) and the Agent Communication Channel (ACC) resident on the Main-Container in the platform.

When run with the (-gui) command, JADE utilises and provides a graphical user interface (GUI) to; manage several agents and containers, enable debugging, allow intra-platform agent mobility, and provide support for multiple, parallel and concurrent execution of agent activities in a platform.
JADE embodies capabilities that enable Agent Communication Language messaging, and its messages are transported as java objects as opposed to strings.

With the capabilites identified, JADE supports automatic registration and deregistration of agents with the AMS in a given platform, a naming convention compliant with the FIPA standard that requires a unique name for all agents in a platform in the form; agentname@hostname, application defined content

languages and ontologies, and the ability to enable external applications to launch autonomous agents by use of the inprocess interface.

3.4.2 JADE Agents Life-Time

Agents in the JADE framework can exhibit six states;

Agent States

States	**Brief Descriptions**
Initiated	In this state, the Agent object is built, but is yet to register itself with the AMS and cannot communicate with other agents because it lacks a name and an address.
Active	An Agent object in this state, is registered with the AMS, has a regular name and address and is capable of accessing all JADE features.
Suspended	This is the state in which a given agent object is said to be currently stopped. With its internal thread suspended, none of its behaviours can be executed.
Waiting	In this state, the agent is blocked, temporarily waiting or in otherwords its internal thread is said to be sleeping on a Java monitor and only wakes up on fulfilment of a given condition.
Deleted	In the deleted state, the agents' internal thread has terminated its execution and is no longer registered with the AMS so it is said to be dead.
Transit	Only mobile agents exhibit this state while they are migrating to the new locations. The system continues to buffer messages that will then be sent to its new location.

In the JADE framework, agents can execute their behaviours only when in the ACTIVE state. To allow and support change of states for the agents, the Agent Class of the JADE java package, provides public methods (in the form; doXXX()) that enable transition from a given state to another, like; doWait() method that allows transition from ACTIVE state to WAITING state , doSuspend() method from ACTIVE or WAITING states to SUSPENDED.

3.4.3 Implementing Agents with JADE

The Agent class in JADE, is used to create an agent and it implements the agent goals in form of behaviours in its setup() method.
To aid accurate task achievement, other application specific methods and variables can be defined in relation to the task to be addressed.
A task is a behaviour of a given agent in the JADE framework, defined and created by use of the Behaviour class [jade.core.behaviours.behaviour [FB05].

To implement tasks, behaviour sub classes are defined, instantiated and added to the agents task list. And any behaviour defined can be controlled by the two

methods; addBehaviour and removeBehaviour, made available by an extention to the agent class at creation.

Agent Behaviours

A behaviour in JADE is said to be one-shot if it is executed only once, cyclic if it must be executed forever, composite if it is made up of several other behaviours, sequential if it is composite and executes its sub-behaviours sequentially, and then terminates when all are done. It is parallel if it is composite and executes its sub-behaviours concurrently and terminates on a condition, given by any of its sub-behaviours, and FSM if it executes its sub-behaviours in a finite state machine manner.
A behaviour could also be a ticker if it is cyclic and must be executed periodically, or a waker if it is a one-shot but must be executed only once just after a given time out is elapsed.

Agent Behaviours	
Behaviours Class	**Brief Descriptions**
SimpleBehaviour	This abstract class models simple atomic behaviours. It has a reset() method that is domant by default and can be overridden by user defined subclasses.
OneShotBehaviour	This abstract class models atomic behaviours that must be executed only once and cannot be blocked. So, its done() method always returns true.
CyclicBehaviour	This abstract class models atomic behaviours that must be executed forever. Its done() method always returns false.
CompositeBehaviour	This abstract class models behaviours that are compositions of a number of other behaviours (children). Its inside children behaviours define its opera--tions performed during execution but it takes care of children scheduling according to a given policy
SequentialBehaviour	This class defines a CompositeBehaviour that executes its sub-behaviours sequentially and terminates when all sub-behaviours are done. It is ideal when faced with a complex task that can be expressed as a sequence of atomic steps.
ParallelBehaviour	This class too, defines a CompositeBehaviour that executes its sub-behaviours concurrently and terminates when a particular condition on its sub-behaviours is met. It is ideal for use when faced with a complex task that can be expressed as a collection of parallel alternative operations, with some kind of termination condition on the spawned subtasks.
FSMBehaviour	This defines a CompositeBehaviour that executes its children according to a Finite State Machine defined by the user. The FSMBehaviour terminates on completion of one of its children registered as final states.
WakerBehaviour	This abstract class implements a one-shot task that must be executed only once just after a given timeout is elapsed.
TickerBehaviour	This abstract class implements a cyclic task that must be executed periodically.

Agent Communication

Agent task/behaviour implementation requires communication initiated and responded to by use of messages in a given language understood by both the sender and receiver.

Communication protocols: JADE employs several interaction protocols as specified by the FIPA standard for agent communication. FIPA-Request, FIPA-query, FIPA-Request-When, FIPA-recruiting and FIPA-brokering are used to achieve the Rational effect, this is discussed in relative detail in [FB05] in the AchieveRE section.
FIPA-Contract-Net is another interaction protocol that allows the initiator agent to call for proposals from a set of responders, evaluates their proposals and accept or reject them. The protocol employs specific messages like CFP (call for proposal), NOTUNDERSTOOD, PROPOSE, etc.

Agent Messaging

JADE provides the ACLMESSAGE class that allows the messaging process during communication of agents. Agent massages are composed of the fields below, as specified by the FIPA standard;
The Sender, Receiver(s), communication intention (performative) from the senders view, Content, Content, Language and Ontology.
And special fields that control concurrent conversations (like; Conversation-id, replywith, in-reply-to, reply-by, etc.)

3.5 Some applications involving agency

Given the so many abilities and undeniable potential of agents, quite a big number of complex situations have been approached with solutions exhibiting or utilising agency. In this section of, some of such solutions or situations are given and described briefly. And since these are quite comparable in complexity to the modelling process, they were used as inspirations to the use of agency in approaching information modelling, proposed by this work.

Fly-By-Wire : This is an aircraft control system, which more less plays a "big-brother" role to the pilots of most commercial planes now. It is so environment aware and autonomous, making it capable of issuing a warning to the pilot in case of inconsistency in received inputs and observed behaviour, which must be appropriately responded to by the pilot, because failure to do so invokes the system to take over control of the aircraft.

Inventory tracking systems : In the now very competitive customer driven business world, prompt and accurate reaction determines industry leaders. Agents in implementations like Oracle and SAP's driven Customer Relationship Management (CRM), are used to make crucial decisions like ordering on demand, which helps firms to save money that would have been tied up in unwanted stoke.

Aircraft maintenace : In a related work to this study, a clear description in the [OS] source, gives a comparable perspective to this work, by elaborating on the use of agents in coordinating several experts that together repair aircrafts in the US-Air force. By use of the RESTINA multi-agent architecture, they describe an agent solution to a very complex, spread and communication intensive problem, where aircraft mechanics are required to consult geographically spread engineers before they make any repairs.

Communication : Agents have also found use in implementations like mail client applications, automatic system updates and patches installation, instant messaging like Yahoo messenger, intelligent text editors like MSWord and Open Office Writer and Integrated Development Environments (IDE) used in developing.

But in a related application to modelling, especially on the complexity scale, is the current project dubbed the Airport Collaborative Decision Making (CDM) accessible on the site: *http://www.euro-cdm.org/*.
It is an air traffic control project aimed at improving the efficiency of air traffic control.
It is similar to modelling because it is as equally communication and collaboration intensive, and it involves a diversity of specialists' or stakeholders' groups.
Its environment is based on intelligence, autonomy, reactiveness and socialisation. The system ensures that accurate flight information is shared and distributed to the concerned and right users in time, and efficiently in order to avoid errors, which errors could be fatal since it is the air traffic being dealt with.
The main goal of the CDM is to improve the way air traffic management, airlines and airports work together at an operational level.

3.5.1 Conclusions

A process is different from its purpose or goal, and hence could be modelled with a different paradigm from the one used for its goal.
Current modelling paradigms are very powerful, and can properly model the goals of the modelling process but, they have not been as efficient in modelling the process itself.
Because the usual domains that are modelled are made up of agent like participants with roles to play, it should only be logical to approach the process with a similar attempt.

Agents, given the fact that they are autonomous, reactive, social and rational, which characteristics also define organisational environments, provide the best way to improve the modelling process. As given in the example implementations utilising agency, there is no doubt that it guarantees collaboration and process coordination. But because the paradigm itself can lead to other complexities, especially since it is a difficult concept, the tools used to realise agency should be as simple as possible and should allow the possible use of other paradigms, especially when modelling the goal/purpose of during a given modelling activity.

Chapter 4

Modelling with Agents

The discussion given in chapter two not only elaborated on the nature of the modelling process but also revealed persistent problems of the process that greatly comprise the quality of the eventual models.
Chapter three, in a relatively detailed discussion introduced agents and the use of agency.
The agent properties and applications of agency in the given cases, revealed potential agent-abilities that if implemented appropriately, could greatly aid efforts to overcome some of the known and identified complexities of the process to a satisfactory level.

By discussing the complexities of the modelling process, and then proposing the introduction of agency and agents could be implemented in the modelling environment, as an ultimate approach to overcoming complexities, this chapter has been used report the study's attempt to the research sub-question 1.

4.1 Contribution Objectives

After carefully studying the modelling process and the agent paradigm, the study through the objectives mentioned next, the accounts given in this chapter and the one that follows, investigated the modelling problem mainly with an aim and hope of;

- Identifying the complexities surrounding the modelling process
- Using the understanding and knowledge achieved about agents, to propose and suggest an approach that can be used to improve the modelling process through the utilisation of agents and XML.
- Designing and modelling the suggested approach into a system reflecting and simulating the modelling process.
- Clearly describing and explaining the suggested system structure, components, activities and how it can be implemented.
- Examining the added value of introducing agency in the modelling process.

4.2 Modelling Complexities and Problems

As Patrick contends, most (if not all) of the problems affecting the modelling process stem from the process' communication complexity.
This is a problem, that is no doubt emphasised by the fact that usually domains are geographically spread, a property that greatly constrains coordination in the process' environment.
The absence of dedicated channels, means and habits of communication leads to mistakes like making assumption about the domain in order to attain conviction of completeness, which in most cases results into development of unsatisfactory systems that are consequently rejected.

Stated in the goals of the study, was the objective to identify complexities of the modelling process. Four complexities were identified and are discussed next in relative detail, and this all is in relation to the four modelling phases (i.e Perception, Elicitation, Abstraction and Representation).
Later on in this chapter, problems due to these complexities are identified and specified as tasks to be achieved through agent cooperation.

4.2.1 Translation

When defining, engineering and communicating requirements about the domain, specifications are normally translated from several languages by several parties. The DE gives his/her perception (informal specification) of the domain in their language (usually a Natural language) which specification is translated by a party specialised in both the DE's and SA's languages, usually for the SA.
During the verification and validation of the model, there is need for the DE to comprehend the SA perception of the domain in order to be able to clarify on issues of the specification that may not be consistent with the actual domain, hence need for yet another translation.

Given a real-world domain of relative size and several stakeholders, the translation most likely increases proportionally with the domain size, usually resulting into repetitive translations which though are initiated with the aim of achieving consistency and a clear understanding or perception of the domain among other things, usually results into inconsistencies.
This repeated translation, which Stijn refers to as "Successive Translation", is a process held responsible for loss of information along the way from the domain expert to the system analyst and back. And it normally comes about as a result of the DE's inability to understand the SA's language and vise-versa.

Experience Experience as viewed in this dimension, refers to both domain and modelling technique experience.

"Practice makes perfect!"

Inline with the English language saying above, experience is highly contended as a major factor in achieving high quality models, because the more experienced the modeller is, usually the better the model's quality and arguably the less time taken to develop the model.
This comes as a result of being able to correctly identify the modelling technique or method appropriate for the domain, and the accuracy in the definition and specification phases.
The modeller's experience in a given technique, guarantees quality in terms of accurate technique application and even may speed up the process, and also a modeller experienced in given a domain specifies the domain with more accuracy than one of less or no experience.
A combination of both domain and technique experience, leads to applying the technique most appropriate to a given domain, complete domain specification, less translation involved, knowledge sharing and learning between SA's of varying experiences, etc.

Though experience is vital, and in most cases present in form of tacit knowledge held by experienced modellers, certain problems hinder its use to the betterment of the models' quality.
These hindrances are all due to poor communication in the modelling process, since it hardly enables coordination and exploitation of the two forms of experience mentioned above.
Poor communication also makes it hard to appropriately and timely react to changes within the model at different stages of modelling, resulting into inconsistencies between the model, the domain and the processes.

Note: But also since all domains are unique (though could be similar), a modeller's experience in a given domain in relation to another domain, is relative and needs to be used with caution.

4.2.2 Ambiguity

Ambiguity as perceived in modelling, is either Language or domain ambiguity, and is a complexity that causes problems for both the DE's and the SA's when interacting with eachother while specifying the domain.

Language ambiguity
This form of ambiguity is related to the translation problem, but presents deeper challenges and problems though.
With the idea of allowing the DE to specify the domain based on their perception, comes the need to use a more detailed and hence powerfully expressive natural language. This way the DE is allowed a chance to specify the domain how he/she perceives it in the most free and natural way known to him/her.

Because natural languages give too much detail, the SA introduces a more restricted language to eliminate unnecessary details and also to communicate with the DE. But in most cases these languages are completely alien and un understandable to the DE leading him/her to agree (or disagree) to something he/she completely has no idea about.
This is not a DE's problem only, because specifying a domain like a departmental store with its involved business rules and processes, for a SA is as puzzling as it is for a lawmaker faced with UML model specifications.
Ambiguity in languages is embedded in syntax and semantics, and it takes shape in examples like words of multiple meanings, varying sentence constructions and grammar structures, etc.

Domain ambiguity
Apart from exhibiting ambiguous natural languages, domains themeselves are ambiguous.
Patrick contends that a domain can be ambagious when its components lack known definitions, which is a big a problem to the modeller when attempting to specify requirements. Because some of the entities in the domain remain hidden, and hence are not modelled thus greatly impacting the domain specifications' completeness.

4.2.3 Formality

One of the most important reasons why modelling is carried out, is to attain control of the domain. According to Theo, the more formal the model is, the more control you have of the domain.
Control over a domain, to a great extent aids the validation and verification of the model in relation to the domain. Other issues like language and domain ambiguity greatly affect the ability to achieve formalised models, and since formal methods in order to properly work as expected, require fulfilment of certain controlled language specifications, sometimes the expressiveness of NLs is not properly realised. Especially, when domain specifications, in order to achieve formality, are subjected to the restricted nature of formal methods.
Formality remains a vital property of a domain model but a difficult one to achieve especially given the usual real-world domains' sizes.

4.3 Agent abilities to solve the problem

Almost all complexities are communication related, so an approach to improving the modelling process should aim at improving the communication within the process, and it should support and enable collaboration among the modellers.
Both natural and software agents are very collaborative, because it is only through this that they learn from the environment, notice changes and react appropriately when required.
This section has been used to distinguish software agents from usual programs or applications, by comparing both their parties' behaviours and nature when executing;

1. Unlike usual programs, agents react to their environment.
   Agents are aware of changes like an introduction or departure of other agents to and from the environment, the change in behaviour of other agents, etc, throughout their execution time.

   For example: An intelligent text editor like MSWord is away of the user's actions (typing). By this it is able to automatically underlines a misspelled word and even suggest a correct replacement according to its knowledge. And if the word is unknown to the editor, it learns it by adding it to the dictionary.
   But a less intelligent editor like notepad lacks the abilities possessed by MSWord only because it does not employ agency in its execution.

2. Agents are autonomous, because they are cable of making and effecting decisions without the need of foreign intervention.

   For example: Mobile phones and other mobile communication devices, by use of agents can now automatically detect new networks (hotspots, GSM networks, etc) in alien areas and inform the user about the discovery, or automatically connect to the new networks depending on the already configured settings.
   This is a new convenient development in device operating systems, which was absent in earlier operating systems' versions.

3. Agents are also goal-orientated hence purpose motivated. By this they do not simply take in inputs from the user and give an output, but they rather capture those inputs required to achieve the task at hand, and give an appropriate output.
   This is a deviation from the famous; garbage in garbage out (GIGO) notion, commonly associated with automation, because agents to a certain level exercise some reasoning about the inputs before a reaction.

4. Because agents are goal-oriented, they tend to be persistent until they achieve their goal or task.
   Given the fact that agents are made up of threads that will run until a given condition is satisfied, agents can afford to be persistent unlike usual application.

   For example: When a user receives an SMS message but his/her messagebox is full, the phone will give a warning alert message, but the sending agent will keep checking if space has been created in the inbox for message delivery instead of destroying the message.

4.4 Agents' identification, specification and description

A complex process like modelling a domain of a considerable size and scope, would require quite a number of agents that coordinate and communicate with each other, while optimising available resources to achieve accurate domain representation.
The team of agents capable of modelling a domain should involve both human agents and automatic (software) agents.

It was important to identify the phases of the modelling process, that could be approached by use of agents and those that could not, in order to improve the process as whole.
Elicitation, since it involves close collaboration between the DE and the SA, would be better approached with an introduction of agency in coordination of its activities. The anticipated improvement could yield more precise and complete requirement specifications.
Abstraction and Representation also could be better approached with agency, mainly to enable sub-tasking in a team framework, in order to exploit resources like experience, and reducing time taken to model.

Some of the modelling process' components, like DE and SA when modelled with agents, are as according this work, compound agents because they involve both human and automatic (software, hardware) agents.
For reasons of simplifying the explanation, these compound agents were modelled with emphasis put mostly on the automatic (software) component, because this holds the input of this approach. But a clear explanation still, is given to show which part of the agent composition is human and which is automatic.

4.4.1 Domain Expert Agent

Due to limitations of machine ability to learn and reason, the DE_Agent is a composition of both human and automatic agents.
This type of agents, defines those agents well informed about the domain. They are the anticipated system users and implementors, who in most cases only perceive the system the way they use it.
When modelling the system, the human domain experts since are the eventual users of the proposed system, they know exactly how they want the system to function in order to facilitate their roles.
Normally they should and can only express their functionality expectations, using a natural language, in form of an informal specification. Which specification, is later refined into formal functional and non-functional system requirements.

Since in reality the domain involves more than one domain expert, it is logical to have it modelled with the contribution of more than one domain expert.
To attain control over domain experts participating in a given modelling process, all agents started of this type are registered centrally, so that all agents in the environment are made aware of the resources of each individual domain expert agent.

The main role of these agents is to make available all business rules in a certain or specified way, preferably in short clear sentences.
Next to this, they should be able to offer support to the system analysts in form of clarification, correction and withdrawing rules appropriately when trying to achieve a complete domain specification.
It is a vital requirement, that the domain expert agents have a life span lasting as long as the modelling process of the domain and they should have a user friendly interface for the domain expert, that allows the human domain experts to easily input, submit and edit rules.

4.4.2 System Analyst Agent

By the System Analyst agent type, this work refers to those equipped with the technical skills, to model the domain by transforming its informal specification attained during elicitation into a formal model, through abstraction and representation.
Because these agents represent a class of professionals with varying skills, and who produce better models depending on their experience, it is vital to have the process support the presence of more than one of them for a given domains' modelling.

The process of eliciting domain information from the domain experts, involves a question-and-answer dialog between them and the domain expert. The SA's work, is very dependent on the inputs from the domain experts, and they develop their perception of the system from the business rules specified, a process that is then followed by requirement abstraction and later domain representation.
Like the DE_Agent type, the system analyst agent type is a composition of both human and software agents, defined by a close collaboration of the two.

System analysts as perceived by this approach, are of two categories; those that play the main role of directly modelling a given domain and those who model indirectly, by offering support in form of technical advice when he/she holds experience in a given domain or modelling technique.
The modelling process of any domain, should involve system analyst's of both categories and the agent control system, should be aware of which system analyst is playing which role for the given domain modelling task in order to appropriately coordinate available resources.
To achieve their roles, they should have a user interface that aides their activities.

4.4.3 Intermediate Agent

This work so far, supports and recommends a single agent of this kind per modelling process, only because an introduction of several of them could lead to redundancy and unnecessary complexities.
The intermediate agent is an automatic agent, that utilises communication protocols like those specified by the FIPA and other communication protocols to coordinate agent activities between domain experts and system analysts.
It is not given any user interface, because it does not require one. But it should run all through the modelling process.

The main purpose for this agent, is to aid translations and exchange of messages between the different DE and SA agents in the modelling process.
By so doing, it should receive and translate (if necessary) queries, and also receive and forward replies to these queries.

4.4.4 Administrator Agent

The Administrator agent type, like the intermediate type is a single agent per any given modelling process. But unlike the intermediate type, the administrator agent is a composition of both human and software agents and has a user interface that allows effective agent monitoring in the modelling process.

It should be the first to start and it is most convenient to run it on the main JADE container in the agent environment.
The main role it plays is to initiate the modelling process by starting all other agents, and it does this through a close collaboration with the JADE standards AMS, DF and RMA agents, that run on the main container.
It has a life span all through the modelling process.

4.4.5 Retrieving Agent

This is the most diversified agent type, and it is used to refer to all data or information storage and retrieval agents that are used to support the DE and SA agents.
Though now it is an agent retrieving information concerning available files, agent information and controlled language's required needs, it is one that can accommodate potentially more functionality than defined in this work.
It is supposed to support the resource intensive nature of modelling and agency.

4.5 Task specification and description

The JADE framework, inline with the FIPA specification implements agent tasks or roles as behaviours.
The table below gives a brief description of the agent types discussed above, their tasks and behaviours as they could be implemented in an agent system for the JADE framework.

Agent Tasks/Behaviours

Agents	**Tasks**	**Behaviours**
Administrator	Start other agents Kill other agents Create the Rules file Coordinate all agents	createIA createDE createSA createRsFile killAgent
Intermediate	Facilitate communication between DE and SA Create the general domain rules file Integrate changes into the domain rules file from the DE	commDeToSa upDateStructRules UpDateRules
Retrieve	Provide data required by agents Keep record of all agent and files Track and communicate file changes	updateFile createFile receiveRequest supplyInfo
Domain Expert	Provide and input domain rules Provide support in defining domain rules	receiveInq respondToInq
System Analyst	Structure domain rules Identify items reflected in domain rules Model the domain	structureRsFile commSysAnal createSchema

4.6 Controlled Language

As earlier mentioned, modelling at one time involves translation from one language to another, especially in communication that brings the DE and SA together. But due to ambiguities in NL's, translation always leads to misrepresentation problems and/or loss of information as a result of successive translation. To come around this problem, it is required to have and use a language that is understable to all parties involved. The language should be definitely defined, in order to attain control over its words in terms of syntax and semantics, but at the sametime should be rich enough to appropriately and completely define the domain at hand.
The language used for the described purpose, is referred to as a Controlled Language. Normally such languages are brief and precise, and their purpose has been realised in several applications, among which are critical systems' environments like; the military and the aviation communications or traffic control systems.

This work through an agent-oriented approach, to let the DE use a language that allows him/her to express themselves in a detailed NL, introduced the use of the XML format to structure their inputs. Since XML can allow information structuring, that maintains all the DE's specifications but at the sametime remains controllable for the SA, it would eliminate the misunderstanding problems that always happen on either side after translation. A middle agent that utilises the XML format files can be introduced to marry the two views. This way the DE has the freedom to express the domain in their language and so does the SA without any of them disadvantaging the other.
The XML standard is discussed next with relative detail, mainly focussing on the structure of its files in the direction it could be used as by this approach.

4.7 XML as part of the solution

The Extensible Markup Language (XML), is now a well established standard for purposes like information retrieval, storage and document organisation or structuring. It is also commonly referred to as a meta-language, due to its broad range of abilities that enable it to be such a powerful tool of information translation from one language to another.
It is extensible in the sense that a user could define his or her own tags, to include in their document but it is also an extension of the Standard Generalised Markup Language (SGML).
XML was developed and is maintained by the World Wide Web Consortium (W3C). And as by the specification in [WR06], XML is a restricted form of SGML which ensures that XML documents conform to SGML documents by construction.
XML documents are made up of entities for storage units, that are the class of data objects described by XML, which standard also partially describes the behaviour of the computer programs that process its data objects.

The inter-connection and cooperation of organisations, has caused a desperate need for communication between organisational information systems.

Though this is in reality such a tedious experience, one outstanding advantage is the ability for most of the systems to understand text-based data. And it is for this reason that XML, which like HTML, is text-based but yet much more powerful than HTML, has become the standard for data exchange between communicating systems recently.

4.7.1 The XML document structure

For the communicating systems to understand and manipulate a given XML file, the XML document must be well-formed as specified by the [WR06].
This is achieved by first ensuring that all begin tags have corresponding end tags, and also by validating the XML document against a Document Type Definition(DTD) or an XML Schema.
An XML document is hungered on a parent and child nodes approach.
According to [XML03], the XML Document Object Model(DOM) represents the XML document in memory and by so doing, reading, writing and manipulating an XML document is made possible. For the DOM to achieve this, it contains and uses nodes that can be of Document, DocumantType, Element, Attribute, Comment and Text type, which generally specify a given XML document item's nature.

4.7.2 Validating XML documents

As earlier mentioned, an XML document is checked against a DTD or XML Schema to ensure that it is well-formed.
For certain limitations problems, DTDs are inferior in typing ability as compared to XML Schemas.
DTDs and XML Schemas, provide information about datatyping for elements/nodes in the XML document validated against them.
But this functionality is limited in DTDs, where only standard already defined types are used, a weakness eluded by the XML Schema approach that fully takes advantage of the extensible nature of the XML standard.
The XML Schemas are defined by the user and in so doing the user is allowed to determine, and define their datatypes which as according to the [WR06] could be simple or complex element types.

Simple Element types

A Simple type element, is defined by a name and restristrictions properties and can not contain another element or an attribute within its body.
The name property, is actually the type name specified by the element and its restrictions properties basically define its characteristics.

Complex Element types

Unlike the simple element type, the complex type element contains other elements and/or attributes within its body.
A complex element exhibits all the characteristics contained by simple elements.

To validate an XML document, the document is read and compared to its DTD or Schema to ensure that its elements contain data that matches the specifications given by its defining document(DTD or Schema).

4.7.3 XML Webservices

Recent developments in the use and application of the XML standard, have seen the emergency of perharps the most important use of the standard so far, in the form of webservices.
According to [XML03], a webservice is a component that implements program logic, and that through use of standard protocols like HTTP, XML and SOAP provides functionality for desperate applications.

Webservices were of little interest to this study, but their communication nature and how it is facilitated by XML, that were relevant and vital to the efforts, to coordinate the modelling processes' components.

4.7.4 Conclusions

The modelling process' ability to allow sufficient communication can determine the ability to break-down domain complexity, which will lead to a better quality model and system.
Problems and complexities due to communication issues like successive translation, language ambiguities, etc, if reduced by turning the DE and SA into collaborating and cooperating agents, could yield a more efficient and effective modelling process, that will output high quality models.

The means of information sharing within the modelling process, affects the efficiency of agents. Information sharing should be achieved through flexible and spontaneous enough means, to allow abrupt adjustments. They should also be structured means, enough to guarantee credibility of the information exchanged. The use of XML to support and facilitate information exchange, was suggested because it is a well structured and flexible format, capable of garanting credibility of its contents through validation enforced by use of XML schemas or DTDs.

Chapter 5

Multi-Agent modelling system structure

With the discussed agent abilities, properties and applications, it is undeniable that agents can be used to facilitate the modelling process.
But important to consider, was the added value of the notion, of having an agent-oriented modelling process as compared to the current.
To investigate the introduction of agents to the modelling process, the study as reported in this chapter, provided a design for a proposed and possible multi-agent modelling system, entailing an analysis of its efficiency, effectiveness and advantages, and an examination of the feasible use of agents in an information modelling situation.

5.1 System model design

The main goal of work reported in this section, was to sufficiently simulate the modelling process together with its activities as would be perceived in the agent paradigm.
By reflecting and emulating the modelling process in different system designs (UseCase, Activity, etc.), a deeper understanding and knowledge was achieved about how to develop and implement modelling agents, in a very well coordinated and collaborated environment and manner. This mainly aimed at improving communication between several parties involved in domain modelling, which would improve and facilitate vital activities like RE, elicitation, verification and validation, that are crucial quality determinants of both the domain model and eventual developed system.

5.1.1 Use Case diagram

The Use-Case diagram given below, has been used to simulate the modelling process. It consists of actors; *S.A_Agent, Int_Agent, D.E_Agent, Admin_Agent, Retr_Agent* and *Modelling Agent*, that collaborate and coordinate with each other, while performing and achieving the several use cases(activities), as given in the diagram. A detailed account of the actors and their activities follows the diagram and table below;

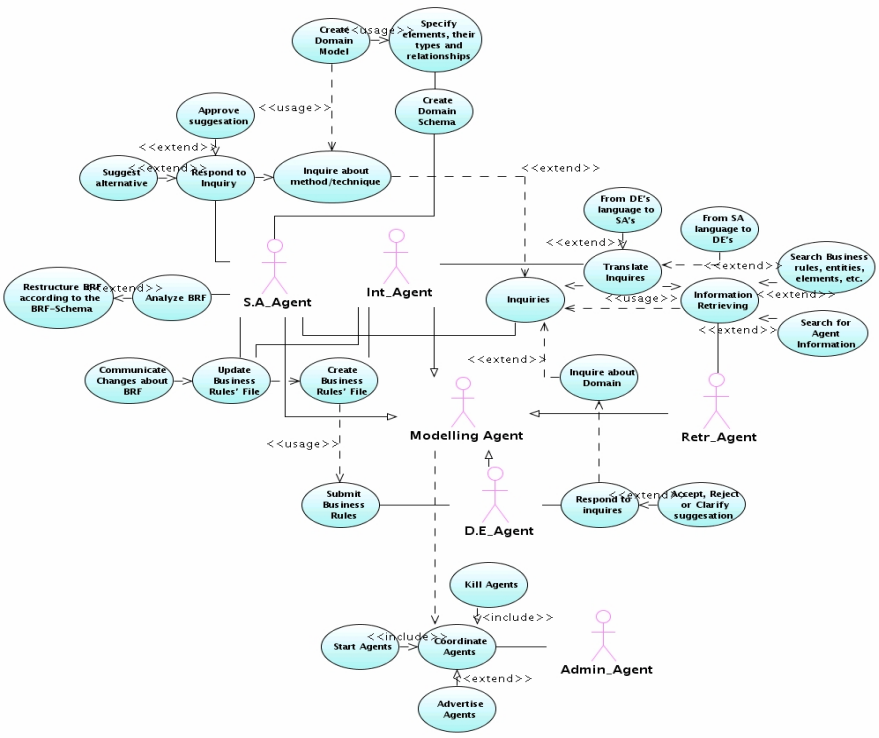


Figure 5.1: Use Case Diagram

Actors and their Use Cases		
Actors	**Use Cases**	**Other Actors**
Admin_Agent	Coordinate Agents Start Agents Advertise Agents Kill Agents	Modelling Agent
Int_Agent	Create Business Rule's File Update Business Rule's File Translate Inquiries	Retr_Agent D.E_Agent S.A_Agent
Retr_Agent	Information Retrieving	Int_Agent D.E_Agent S.A_Agent
D.E_Agent	Submit Business Rules Respond to Inquires	Int_Agent Retr_Agent
S.A_Agent	Analyse BRF Update Business Rules' File Create Domain Schema Inquires Respond to Inquiry	Retr_Agent Int_Agent

The admin_Agent actor's main activity, is to facilitate agent coordination through the "Coordinate Agent" use case.
The activity of coordinating agents, involves the "Start Agents", "Advertise Agents" and "Kill Agents" use cases. These concern all actors, that are of agent-nature started after the admin_Agent, which is why the "Other actors" column bares the "Modelling Agent", for this table row.

The Int_Agent actor, mainly implements three use cases; Create Business Rule's File, Update Business Rule's File, and Translate Inquiries in collaboration with the Retr_Agent, D.E_Agent and S.A_Agent actors.
It facilitates communication in the modelling process, by ensuring that decision making is collaborative and involving for all parties concerned and relevant.
Its sequence of events, is triggered by the D.E_Agent actor's submission of business rules, which are then compiled into a well-formed XML file the Business Rule's File (BRF). In a bid to achieve completeness, the file is subjected to an analysis by the S.A_Agent, who sends queries through the Int_Agent to the D.E_Agent, to elicit more information. When the D.E_Agent actor changes any of the submitted rules, the Int_Agent updates the BRF, and informs the S.A_Agent actor that initiated the inquiry. Changes made in the domains' formal specification by the S.A_Agent, are also communicated to the D.E_Agent, and they too, are reflected in the BRF.

The Retr_Agent actor supports activity coordination, by putting the Retrieved Information into effect, collaboratively with the Int_Agent, D.E_Agent and S.A_Agent.
Actually the Retr_Agent actor is an abstraction (for simplicity reasons) as by this design, representing several other actors involved in data and knowledge storage, processing and retrieving, like data sources; relational databases, XML files, etc.

5.1.2 Activity diagram

The activity diagram below, was used to potray the sequence of events, that would take place in the modelling process as perceived by the agent-oriented approach suggested by this work.

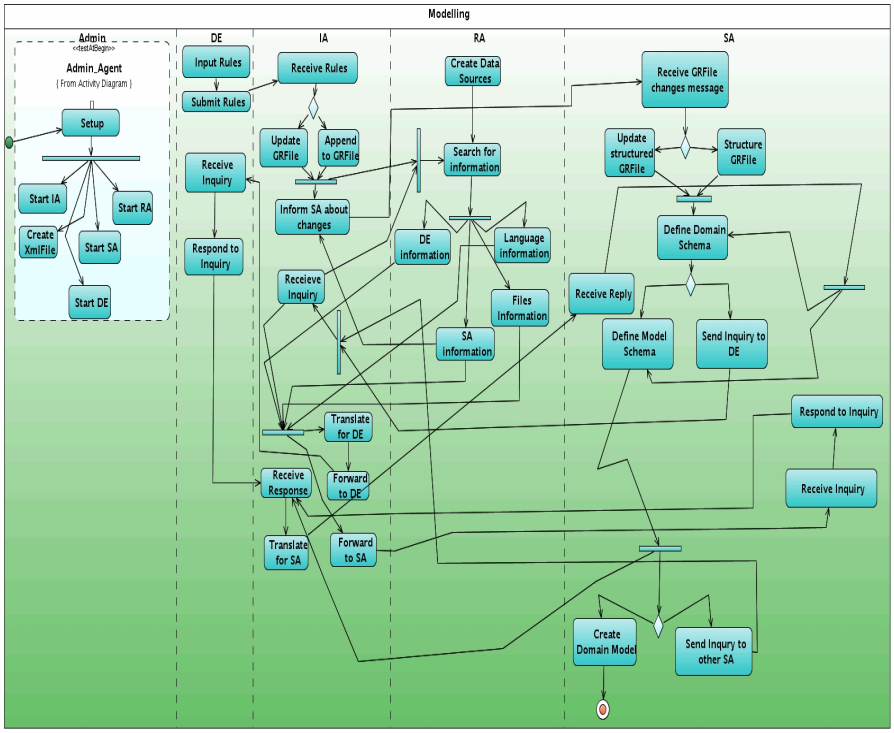


Figure 5.2: Activity Diagram

The table that follows, highlights the components of the activity diagram in three columns, as described below;

The Main Activity: contains the events considered fundamental at the different phases(partitions) of the activity diagram. These phases have been partitioned as; Admin, DE,IA,RA and SA in the activity diagram, and this column represents the major events in each of these partitions.

The Participants: are the different agents, that in one way or another play a role in the given major event in the main activity column.

The Related Activities: is the column that contains those other events, that affect a given major event.
These could be required events for the main event to be achieved, or could be events that require the main event before they can take place, or they could be resultant events after a main event.

Activities		
Main Activity	**Participants**	**Related Activities**
Setup	Admin_Agent	Start IA Start DE Start SA Start RA Create BRsXmlFile
Input Rules	D.E_Agent I.A_Agent S.A_Agent	Submit Rules Receive Inquiry Respond to Inquiry
Receive Inquiry	I.A_Agent D.E_Agent S.A_Agent R.A_Agent	Update GRFile Append to GRFile Inform SA about changes Receive Inquiry Receive Response Translate for DE Forward to DE Translate for SA Forward to SA
Search for Information	R.A_Agent I.A_Agent S.A_Agent D.E_Agent	SA Information Create Datasources DE Information Language Information Files Information
Define Domain Schema	S.A_Agent I.A_Agent R.A_Agent D.E_Agent	Receive GRFile Changes message Update Structures GRFile Structure GRFile Receive Reply Define Model Schema Send Inquiry to DE Respond to Inquiry Receive Inquiry Create Domain Model Send Inquiry to Other SA

The *Setup* event, signifies the beginning of the modelling process as reflected by the system. Its only participant, is the *Admin_Agent* through whom, all system components are initialised by starting the other agents in the process.
This event starts the Intermediate, DE, SA, Retrieve agents, and creates the general file (BRF) that later is filled with business rules by the DE and the Intermediate agents.
The Input Rules activity, mainly performed by the DE and Intermediate agents, initiates elicitation that is followed by the Receive Rules event, which is liable to corrections through a collaborated dialog involving the SA, DE and Intermediate agents. And is intensively facilitated by the retrieve agent, through a prompt availability of information.
The achievement of a complete informal domain specification, triggers the *"Define Domain Schema"* event, that too is liable to correction, achieved when

the SA collaborates with other SA agents through the *Inquiry* related events, facilitated by both the *intermediate* and *retrieve agents*.

5.1.3 Class diagram

The class diagram has been used to reflect the internal structure of the proposed system components, by showing how the different classes will be developed in correspondence to their counterparts, that together will define one system.

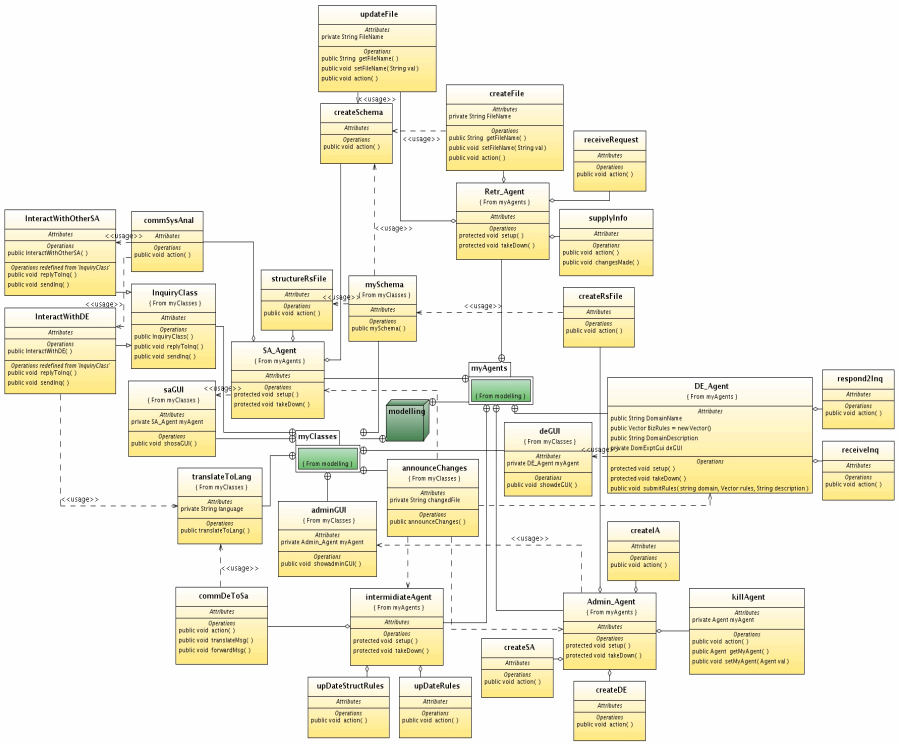


Figure 5.3: Class Diagram

System Classes		
myAgent Package	**Inner Classes**	**myClasses Package**
Admin_Agent	CreateDE CreateSA CreateIA KillAgent CreateRsFile	adminGUI announceChanges mySchema
IntermidiateAgent	upDateRules upDateStructRules commDeToSa	announceChanges translateToLang
Retr_Agent	createFile updateFile receiveRequest supplyInfo	createSchema
D.E_Agent	respond2Inq receiveInq	deGUI announceChanges
S.A_Agent	structureRsFile commSysAnal CreateSchema Inquires Respond to Inquiry	mySchema saGUI announceChanges InquiryClass

The table above, in its first column gives classes from the myAgent package, followed by their inner classes in the second column, with the third column containing classes from the myClass package.
All classes in the myAgent package are of Agent type, so extended from the JADE Agent class, hence containing protected setup() and takeDown() methods that are of void return type.
Their inner classes, which implement their tasks, extend from the JADE Behaviour class and thus, they all contain a public action() method, that returns nothing but implements agent functionality.
The classes in the myClasses package, are those that do not extend from JADE classes, but rather are typical java classes achievable through use of standard java libraries.
The agents classes, that expect input from human agents; *Admin_Agent*, D.E_Agent and S.A_Agent, utilise graphical user interfaces to ease receiving, editing and submitting of inputs by human agents.
The JDOM library is used to create *the mySchema* class, from which several classes are extended. This ensures easy and flexible XML file creation by the Admin, Intermediate, Retreive and SA agents.

5.1.4 Deployment diagram

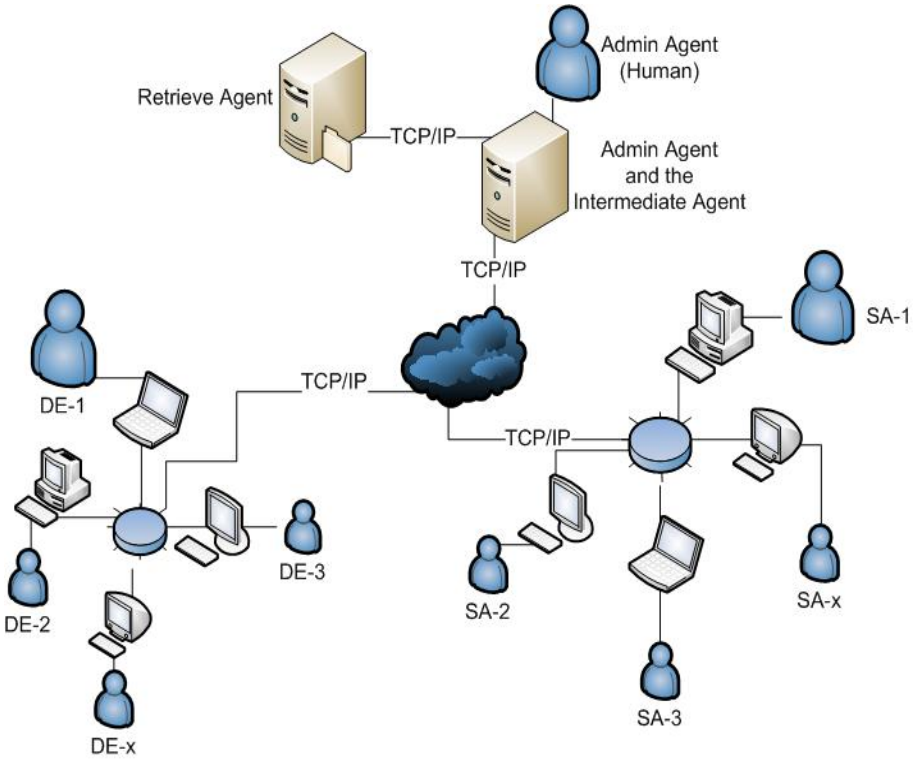


Figure 5.4: Deployment Diagram

According to the deployment diagram given above, the Admin and the Intermediate agents are resident on one server machine. The Admin agent is represented by human and software agent components, because the software agent requires domain specific inputs from the human agent in order to initiate the modelling process.
The Admin, Intermediate and Retrieve agents are all resident on servers, because their services affect and are accessed by several other agents in the environment. The Retrieve agent, since it deals with information processing, storage and retrieving, should be facilitated by a file server, that ensures an appropriate file storage structure, convenient and efficient to access.
To avoid possible increase in complexity, in the implement of the system, socket connections utilising TCP/IP, are suggested for use in this approach but this is not absolute.

As shown in the diagram, both the system analyst and the domain expert environments are of a ring network architecture, consisting of agents with symmetric team leader and team member/player rotating roles, depending on the activity at hand, sub-domain or component being modelled.
The deployment diagram reflects that a domain can be defined and modelled

by a number of domain experts and system analysts. And through sub-tasking, a domain could be sub-divided into sub-domains, where each sub-domain is informally defined by a respective DE. But each sub-domain specification must meet the whole teams' approval, in order to achieve consistence in the domain specification.

Likewise the domain informal specification, is broken down into sub-components on the side of the system analysts. Each component is defined and modelled by a system analyst in collaboration with the others, and at the end of the process, all components are brought together as one formal domain specification.
A given domain expert or system analyst, whose component is being modelled at a given moment in time, in collaboration with the others in the team of system analysts, is the team leader then. And when another component is being modelled he/she is a team member.
This allows switching of roles within the modelling teams, which positively impacts on the precision of the specification by making it possible for the most knowledgeable modeller about a given component, to model it.

5.2 Design Model Investigation

The quality of the given model should take into consideration all the standard model properties discussed in chapter two. The realisation of these requirements was examined through an evaluation and investigation of how the modelling process aligns with a multi-agent system approach.
This section, has been used to give an account of the evaluation of an agent-oriented approach to modelling, aimed at determining its appropriateness with modelling. The effectiveness, efficiency, advantages and disadvantages of introducing agents in the modelling process were evaluated and discussed in detail, with reference to other agent implementations in some known complex applications.

5.2.1 Appropriateness with modelling

So far agents have been used in quite a number of applications, but still a lot is yet to be achieved from them, given their properties and potential.
As according [OS], where it is contended that the many theories and simulations of multi-agent information retrieval systems are yet to yield reasonable real applications.
One reason for the scarcity of agent applications pointed out in [OS], is the lack of sufficiently distributed, rich in information and complexity, real problem domains.

The complexity of information domains, especially those involving information intensive and collaborated scheduled tasks, like air traffic control systems, is unquestionable. Mainly, due to the several stakeholders that not only have different domain perceptions, expectations and use of these systems, but are also in most cases geographically distributed and spread. The distributed nature of such domains, constrains coordination and communication within the domain, greatly increasing their complexity.

When it comes to domains like those of multi-national organisations, the domain due to its spread nature, contains sub-domains that are composed of people who though may have a uniform purpose, have different ethnic background, languages, laws, etc.

Most domains activities in the domains briefly described, are carried out in a collaborative way, similar to the activity of the properties that define the agent paradigm. These activities should affect and effect each other, and should yield to universally collaborated decision making, that considers all stakeholders views, opinions, perceptions and inputs.
It can only be logical to model such domain with a paradigm similar to their nature, because this helps to give a glimpse into how exactly the system might evolve and respond to the changes within the environment.
Since most organisational information and knowledge is distributed, diversified and geographically spread, their domains can be said to be complex enough, to be modelled with agents in order to reduce complexity.

5.2.2 Effectiveness

The beginning stages of modelling are the most messy, tedious and unpredictable mainly due to high levels of human error in these stages, that involve mass introductions of information.
The agent-oriented modelling system, as shown in the deployment diagram, should allow decentralisation of decision making, involving as many stakeholders as managable, right from the start up to the end of the modelling process, defining or contributing to the definition of their respective domain aspects.

In the above given diagram, A is the domain being defined. To reduce its complexity, sub-tasking is introduced through the sub-division of the domain into B, C and D sub-domains. These are then defined locally, but in collaboration with eachother's involved activities and together they define A.

For example: When modelling a university faculty, the faculty is the main domain, its departments are its sub-domain. But even the departments themselves, could have sub-domains in the form of sections, depending on the complexity of the faculty being modelled.

The DE's in A, specify the domain supported by their respective sub-domains, and amongest themselves in A with the guidance of the team leader, qualify a specification as a rule to be entered in the domains collection of rules.
The specification before being qualified and added as a rule, is forwarded to all DE's with relevant knowledge and interest about it, which prevents errors and misunderstandings at later stages in the modelling process.
These DE's hold a clear understanding of the specification, because they possess a detailed view of the input. This enables them to identify inconsistencies within the proposed rule and the domain at hand, as the process progresses.

Though the decentralised decision making approach is intended as a solution, it has great potential to drag on endlessly. This risks the vital quality of achieving completeness, in the domain specification.

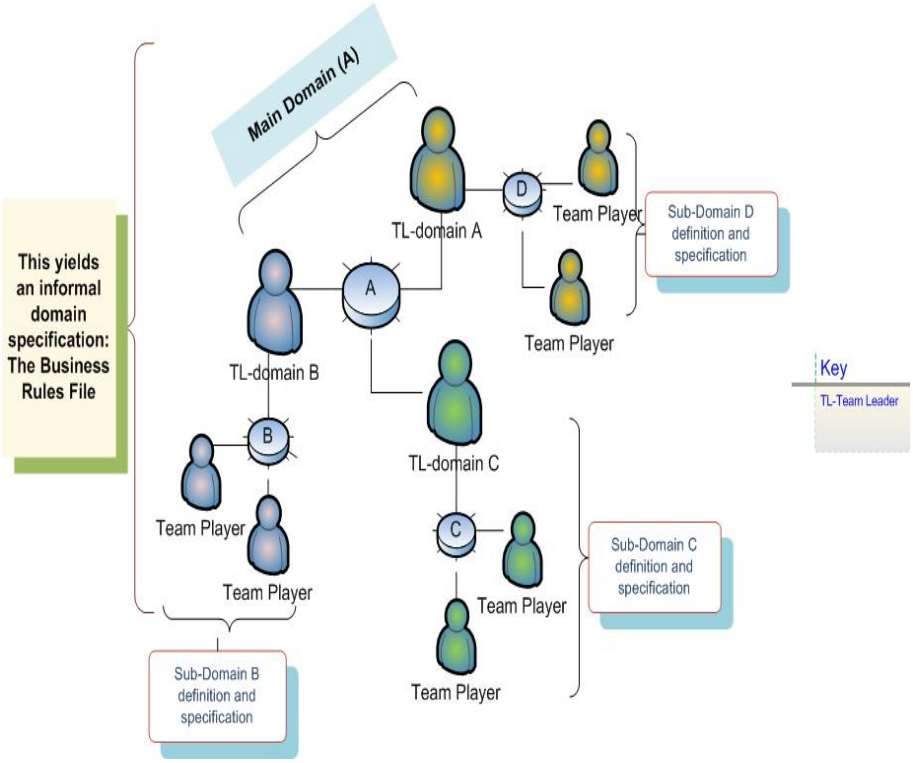


Figure 5.5: DE Sub-tasking

There is need for the system to allow a way of reaching agreement on contended issues, through the use of a voting system that should act as filters, for the inputs into the rules files.

On side part of the DE during domain specification, the process of breaking down the domain into sub-domains, not only yields a complete business rules file, but the yielded file is also consistent with all the sub-domains and precise, given the fact that each sub-domain is represented sufficiently and effectively.

Unlike on the side of the DE's, the domain to the SA is a composition of several components as opposed to sub-domains, derived from the business rules supplied, that can be distributed as tasks to a number of SA's to be modelled. Logically and realistically, modelling should involve less SA's than DE's, which helps to elude unnecessary problems associated with big-teams', and boosts close collaboration within the process, that yields the property of compactness, of the model being developed.

Inconsistencies and precision problems in the domain specification, trigger clarification related actions, leading to regular cross-checking of the specification with domain as defined by the DE's. By so doing, the model is validated and verified component by component, hence ensuring the accuracy of the domain representation.

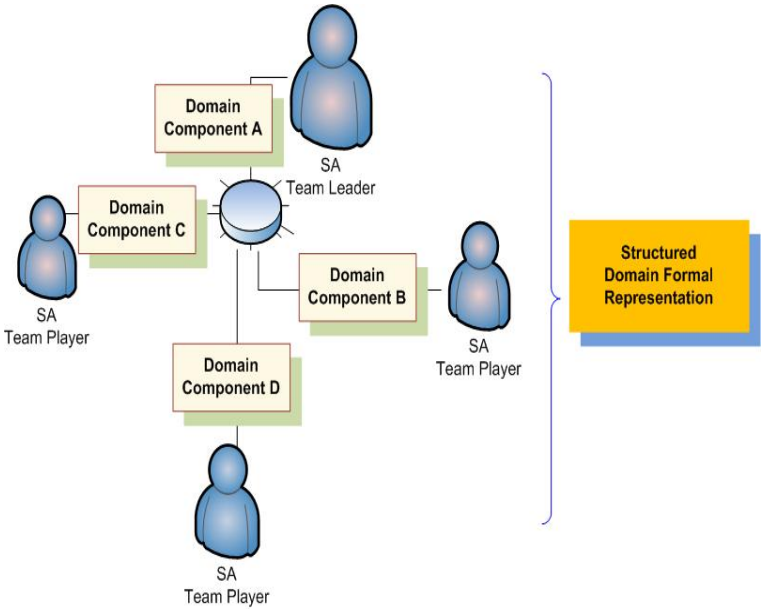


Figure 5.6: SA Sub-tasking

The SA mainly transforms well a formed, but unstructured business rules XML file, into a very structured schema file, intended to mainly abstract semantics availed in the informal specification. This yields the formal representation of the domain problem at hand.

Note: The team leaders are not at the centre of the team, because they are leaders only when their respective sub-domain is being modelled. The roles of each member of the modelling team continuously rotates according to the changes in the environment.

5.2.3 Efficiency

To ensure the quality of the model outputted at the end of the modelling process, the agent-oriented approach suggested is banked on the perception, that modelling is a procedural refinement process.
The process is perceived to start with general and very detailed informal domain specifications, that under go a first formalisation, that transforms them into a general structured file that is later transformed into a format required by the modelling technique of choice.

Elicitation is a phase that actively engages both the SA and DE. But the quality of results of the elicitation phase, highly depend on the quality if the questions asked by the SA and the precision of the specification by the DE.
Because it is hard to guarantee achievement of good questions from the SA, letting the DE to actively participate in the elicitation, in a way that he/she is kept well informed about the specification by the SA, will most likely reduce problems of misrepresentation associated with translation complexities and even

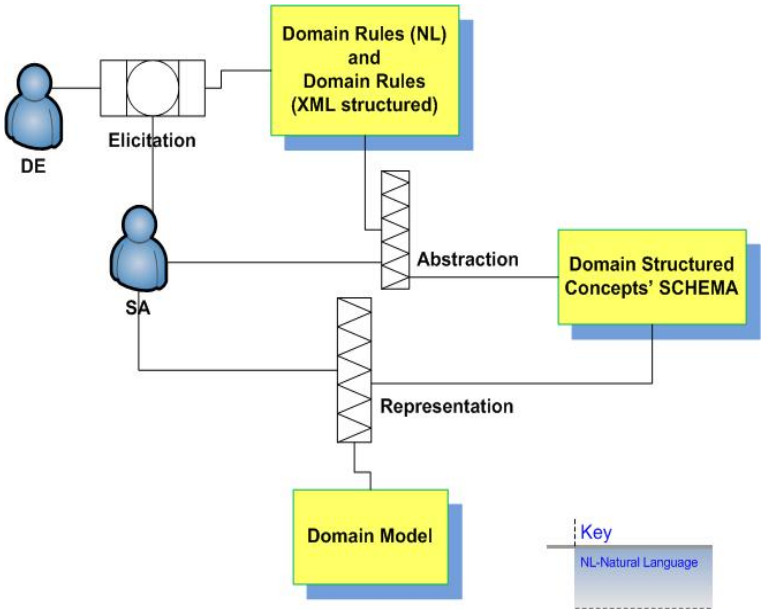


Figure 5.7: Modelling as a filtration Process

invoke the DE to discover and supply information not given in the initial informal domain specification.

The modelling process in order to coordinate its activities, should heavily relies on dedicated communication.
So ensure sufficient communication, the suggested approach will employ the JADE tool, that through its messaging classes makes use of known standards like HTTP, TCP/IP, etc.

The volatility of domain rules and specifications, mainly due to varying perceptions, understanding and experiences, that normally exist in a relatively complex domain, creates a requirement to store domain rules in files of a format flexible enough to support frequent changes.
The system through the JDOM tool, should provide a quick and convenient means of updating domain rules, by storing the rules into a neatly structured and formal XML file format, made up of both simple and complex elements.

The business rules file is of the XML document structure format, defined with the use of simple XML tags, based on a non complex XML schema.
It is from this business file, that the domain schema is developed in a collaborative effort that involves domain experts and system analysts.
The main domain schema is developed first, before anyother sub-domain schema.
It mainly consists of all entities defined in the rules file and all the rest identified during RE activities.
All tags used in component schemas, should be contained in the domain schema and if not, should be added to it after an agreement is reached by the modelling teams.

The refinement process is actually a procedural process, that involves component development through developing individual schemas for each of the componets, which schemas are validated against the domain schema.
A change suggested in a given component schema about the domain, should be reflected in the domain schema and the business rules file. And to ensure that this is achieved, the environment should trigger an action in case of inconsistencies.

The efficiency of the modelling process', will greatly be boosted by an agent-oriented approach that allows several SA's, the opportunity to specialise in domain components familiar and known to them.
By the system allowing a SA concentrate on a given component, it facilitates abstraction, and the team-leader, and team-players approach, ensures that all individual component abstraction is incorporated into one broader domain abstraction.
Still the ability to have more than one SA facilitates the representation phase, by letting experience play a role in the alignment, of the most appropriate modelling technique to the domain at hand, thus ensuring that the domain is properly and accurately represented.

5.3 Cooperating issues

The problems affecting the modelling process were all found to be directly linked to the short comings of the process' communication.
When the DE and SA communicate, due to domain or language ambiguities, and problems like successive translation, so often is the case that the specification hardly matches the qualities required of the model.

5.3.1 Modelling the domain without the SA

One solution to these problems resulting from communication between the DE and SA, according to Stijn, would be to eliminate the SA and let the DE model the domain by themselves. This could be attempted by automating the SA's side through embedding in tools used by the DE. This way the SA is hidden from the DE, and only relies on the DE perception of the domain. This approach for sure could have some advantages mainly in reducing the time taken to model and others.

But Patrick, like this work disputes the notion of eliminating an interactive and collaborative participation involving a SA and DE during modelling.
If the problem is due to lack of proper collaboration, the first option should be attempting to improve and support collaboration, by enforcing cooperation supported by improved dedicated communication means.
The one clear fact is that the SA needs the DE in order to make sense of the domain's components, but so does the DE need the SA in order to represent the domain in a format ready for automation.
The automation of the SA is greatly impaired, by the limitations in the machines' ability to intelligently reason about concepts of the domain, calling for a human to play a role in the SA agent.

A collaborative atmosphere does not only favour a DE-to-SA information exchange, but also a DE-to-DE and SA-to-SA exchanges which enable other vital aspects like learning and team work.

5.3.2 Added value of cooperation

The close collaboration, contribution and participation of the DE and SA guarantees acceptance of the eventual developed system.
Since they work together on the domain model, the DE can express dissatisfaction or satisfaction about the domain representation in the early stages, saving the SA costly and embarrassing situations, that normally are caused by unsatisfactory systems.

Since in real-world projects, the domain is big in size, the SA needs help or support, most importantly from more experienced SA's to define the domain completely and precisely.
An approach allowing several SA's connected and collaborating with each other, if properly scheduled with clearly defined roles for each SA, provides the much required support through knowledge sharing, to the SA particularly when verifying and validating the domain model. An efficient communication enabled environment, enables the SA to seek support from other SA's to verify the model and from the DE's to validate the model.
When a SA attempts to model an unfamiliar domain, because his/her perception of the domain is constrained, in most cases the model achieved is a misrepresentation, that is either an under or over representation of the domain, mainly achieved based on assumptions.
But by allowing the DE to input domain rules, that are later structured into a well-formed XML file, the SA can concentrate entirely on understanding the specified objects and constraints, that are used in defining concepts used to represent the domain, rather than, to define or identify the unknown objects and constraits.

The main added value of introducing cooperation and collaboration into modelling, is the ability to employ sub-tasking, in the organisation and coordination of modelling activities.
Its through sub-tasking, that both the DE and SA, can concentrate on doing what they know best, in their contribution to the specification exercise. And it is only by close collaboration, that results from these sub-tasks are brought together as a model of a unique domain.
The close cooperation ensures that inconsistencies in the specification are detected almost instantly, invoking inquiries and requests for clarification, that ensure the model remains complete, consistent and hence accurate, from the initial stages to the last stages of the process.

Often, model validation and verification is done at the end of the modelling process, which normally is problemantic, especially in cases involving domains of relative complexity and size.
It is because of such, that validation is rarely achieved completely as a quality property required of the model. And it is always assumed, supported by the

assumptions based on the *"Falsification Rule"*, which contends that a specification is valid until proven otherwise.
But cooperation between the SA and the DE, ensures validation and verification throughout all the phases of modelling.
Since the domain is divided into components that are validated individually after they are completed, collective validation and verification can be achieved, which can greatly improve the validity of the developed model.

When faced with a very complex domain, sub-tasking is used to break the domain down, into less complex sub-domains that are defined by different DE's, with respective detailed knowledge about them, enabling the greatly desired control over complexity, that is reason for modelling.
Since experience is very vital in achieving model quality, by sub-tasking and allocating tasks to SA's in accordance to their experience, cooperation guarantees model quality and process efficiency as well.

5.3.3 Benefits of having multiple System Analysts

The sole aim of introducing agents in modelling, was to reduce complexity in the process, and like on the side of the DE, sub-tasking reduces complexity when several SA's agents are involved in modelling.

- By having the right number of SA's modelling the domain in a collaborative forum, the process of modelling is more efficient in terms of accuracy and speed, and is more able to yield a valid formal representation of the domain.
- Since the model's quality requirements, demand that the modelling process remain intentional as opposed to extensional [PF05], sub-tasking enforces deeper and focused abstraction and representation of the domain components.
- Different modelling techniques, are suitable for modelling different domains, and for complex domains like business organisations, that normally consist of several sub-domains, determining the appropriate technique to employ for the right domain component, is a difficult but yet crucial, because the choice of a wrong technique, guarantees an inaccurate representation of the domain.
- By having several SA's, their experience and knowledge could be exploited to further knowledge sharing, learning, and to improve the model quality in general.

5.3.4 Benefits of having multiple Domain Experts

A modelling process involving several DE's, reduces the complexity of the domain to a certain extent during elicitation through sub-tasking, by making domain specification more managable. This way, a more detailed and focused but precise, informal domain specification is achieved, since it is done by the DE, most informed and knowledgable about a given component.

The prospect of having several DE's, allows the DE's to compare eachothers knowledge and perception, and even with that of the SA. Since DE's in many cases have been blamed for not knowing their expectations of the anticipated systems, this collaboration can help them to realise new information about their domain, which they can use to refine and adjust their perception of the domain and thus give a complete specification.

The process of defining specifications in a team, implies that several perspectives have to be in agreement with the specified rules before they are used to represent the domain. The specification is then truly consistent with the domain, because it is properly verified and validated not based on assumptions, but rather on knowledgable and informed perceptions.
As all teams have problems, a team of DE's is like to face difficulties in trying to reach an agreement, due to conflicting information, lack of knowledge or repeated formulation. But approaches like majority voting, to select the best alternative to investigate as first option could be employed to reach an agreement. This investigation is considered concluded, when all participants are satisfied from their point of view.

5.4 Sample Session

To illustrate the use of agents as anticipated by the proposed approach, in this section an example is given and discussed.
The domain is modelled with a perception of the environments as an agent environment, with several distinguished agent players, but all contributing collaboratively, to a sole goal achievement. The example models a growing child's domain, which is defined by a lot of uncertainties in the beginning but eventually evolves into a recognisable purpose.

5.4.1 Example

Taking the example of a young child developing through stages from infancy to adulthood, by trying to make sense of his/her surroundings, the child becomes the SA this domain and the people in his/her life are the DE's from whom he or she elicits information and knowledge.

Agents in the domain					
	Child	**Parents**	**Sibling**	**Teachers**	**Peers**
DE		Mother(Paulina) Father(Ivan)	Brother(Jacob)	Solome Cezanne	Daniel Patricia
SA	Solomon				

The table above gives Solomon (the SA) born to Paulina and Ivan. He has an older brother Jacob, attends a day care centre where he is attended to by his teachers Cezanne and Solome. And while at school, Solomon plays with fellow infants Daniel and Patricia.

The child's' parents give him information and knowledge vital for him to survive in a social setting, by training him to be responsible, disciplined and

respectful. Sometimes his slightly older brother Jacob, may teach him words or habits contrary to his parent's code of conduct, usually leading to disagreements. Often this unwanted information is discarded, in favour of emphasising other more important positive growth aspects.
Solomon heavily relies on the relationship with his brother and playmates, to effectively develop his social and communication skills. And for his intellectual and academic development, relies on his teachers.
The above description, briefly highlights the child's learning process when growing, but mainly the main aim of this collaborative contribution in Solomon's life, is to allow and enable him model a proper and meaningful life, out of these experiences.

Like the modelling process, Solomons' process of growing will contain phases or stages as he progresses in development towards adulthood. These phases generally will be; infancy, puberty, young adulthood and adulthood.

Infancy to Puberty: The phase of infancy to puberty defines a stage of elicitation. During this phase, the child will receive so much information about life from all the agents in the domain, but he is yet to make sense of it because most of it, is still vague and unfamiliar to him.

Puberty and Young adulthood: The stage between puberty and young adulthood is comparable to the abstraction phase. This is when he will try to generalise and sort from all he has been taught, what he should consider relevant and with that, he will attempt making choices like; choosing a career direction, living by himself or taking a first job, etc.

Young adulthood to Adulthood: The stage between the phases of young adulthood all through to adulthood, will reflect the grown child's representation of his perception, of the concept of life as taught to him by the agents in his environment.

All stages are equally important, and weaknesses in any of the stages is clearly reflected in the child's representation, of his understanding of life. And normally, it also reflects on the agent (person) in the child's life, that was the cause of the weaknesses.

But in a realistic setting, learning is a continuous process even after reaching adulthood. This impacts greatly on the growing child's perception of life. For this reason, every now and then, the child's perception of life or certain aspects of life, is appropriately adjusted as new knowledge and information is discovered.
This example, has emphasised the importance of the process in comparison to the goal. Like the way up-bringing impacts on the future life of a growing child, so will the modelling process affect the eventual model and system developed.
All agents need to cooperate and collaborate their tasks, in order to achieve a balance in the process and the quality of the output.
Like a child in the infancy stage given lots of information that they can hardly understand, so is the SA modelling a given domain using a file of specified rules. There is need for him to be able to ask question that will prompt complete,

precise and comprehendable responses, from which concepts can be abstracted, that later, are formally defined as the domain specification.

5.4.2 Conclusions

The elicitation phase in modelling, contains the greatest uncertainties and yet impacts greatly, on the efficiency of the other phases of modelling as a whole.
Perception could be developed and improved by very effective elicitation, in order to facilitate precision in abstraction. But because perception goes hand-in-hand with the learning process, it is a continuous process, that even extends to system development or beyond.

By introducing cooperation in the modelling process, the domain complexity should be reduced greatly, because it presents a chance for collaborative sub-tasking, that by enabling specialisation, team work, collaboration and co-operation, guarantees continuous cross-checking of the process' activities.
Through these, cooperation facilitates an intensional modelling process based on relevant and appropriately aligned experience and knowledge, that is efficient and effective to produce a domain model, of sufficient quality.
All participants in the modelling process should perform their activities, and achieve their goals collaboratively in a team, if the resultant model is to exhibit the required essential qualities.

Chapter 6

Conclusion

Given system development practices like model driven development, it is evident that the quality of the model directly affects that of the eventual system developed. Mainly motivated by the argument that; ***"A model of good quality yields a system of sufficient quality"***, the research was done with the sole aim of finding ways through which information model quality could be improved.
This chapter has been used to give summarised descriptions of observations made in the chapters before and conclusions reached as a result of these observations, guided by the research questions.

6.1 What has been done

In the first chapter the subject, goal and purpose of the study were discussed in a description that gave reasons why information systems, are of critical value to information sharing and organisational success. It's through this discussion that the study registered the need for organisations to attain control over their information and knowledge as very important, and achieved through techniques like modelling.
The chapter also gave an account of the study's endeavours to; identify the exact stages in a given system's life span that utilise and involve modelling (requirements engineering), define and describe the; problem area, research goal, question, structure and methodology.

6.1.1 Sub-question One

The main purpose of including this question as a study tool, was to offer some guidance to the efforts aimed at achieving a deep understanding of modelling as the subject of interest.
As anticipated, a detaild study of modelling, exposed issues in the process that required improvement, among which were;

- The need for more coordination and structuring in the modelling process.
- Domain experts were blamed by System analysts for not knowing what they want, as regards their expections of the system representing their domain.

- The fact that different domains are better modelled using certain different modelling techniques, etc.

What issues of the modelling process need to be addressed in order to achieve models of better quality and what possible means could be used to address such issues?

The two identified categories of participants involved in modelling as given in chapter two, were the Domain Experts (DE's); who hold detailed knowledge concerning the domain, and the System Analysts (SA's) that hold technical knowledge, necessary to formally represent the domain.
They together, were found to perform the task of modelling through four phases identified as; Elicitation, Perception, Abstraction and Representation.
And based on these phases, modelling has been defined by this work as;
"The process of formally representing abstracted knowledge, obtained through information elicitation, from a knowledgable perception about a given UoD."

The study identified issues within the modelling process that were found mainly to be caused by, the lack of sufficient communication between the DE and SA during the four phases of modelling, and also due to the fact that most of the modelling techniques used give no or little consideration to modelling, as a process. This meant that though these techniques have the potential to yield high quality models, they do not, only because the process through which they are employed is poorly coordinated and structured.
The two possible approaches to issues of modelling investigated in this work were to;

1. Eliminate communication between DE and SA by letting the DE model the domain.

2. Improve and facilitate communication between DE and SA by introducing agents into modelling so that the DE and SA collaboratively model the domain together

The first approach would be achieved through automating the SA completely while availing supportive tools helpful in formalizing the domain specification, to the DE. But realistically, it was found very complex to automate the SA, especially since sufficient rationality and intelligent reasoning in machines as compared to natural agents, is yet to be achieved.
The study adopted the second possibility with the anticipation that improved communication between DE and SA would render vital support in reducing complexities within modelling environments.
Since most modelling techniques studied in this work gave little attention to the process of modelling, the agent paradigm was adopted as away to model, facilitate and support the process. A suggestion that inspired the second sub-question.
Modelling was found to be used as a means of attaining control over the problem domain, a tool of generating, discovering and sharing information, and most importantly, a way to break-down complexity in a given domain.

The main conclusion reached in this part of the study was that; modelling of any given domain involves several parties, that must cooperate and collaborate sufficiently in their activities and tasks, in order to achieve their purpose.

6.1.2 Sub-question Two

The main achievement of the study as guided by sub-question one, was the visualisation of the model and the modelling process as being separate entities, faced with different but related issues. This enabled the work to progress with an investigation of the agent paradigm, in an effort aimed at determing how agency could be used in facilitating modelling as a process of activities.

> What qualities would a multi-Agent system possess that would be of relevant value in support of information modelling?

The investigation of the agent paradigm as a possible solution to the issues identified in the modelling process, as given in chapter three involved a detailed evaluation of agent properties. This through a comparison with the modelling process, led to a conclusion that the modelling process' structure of activities and participants, perfectly aligns with the nature of the agent paradigm. Through the agent paradigm, the modelling process could be perceived as a complex enough environment consisting of agents (DE and SA) with tasks and goals to achieve, as dictated by frequent changes within the environment. The main agent properties identified were;

- Autonomy
- Reactivity
- Social Ability
- Pro-Activeness
- Rationality

Agents according to the literature studied were found to have specific tasks at their creation, that they achieve as thier behaviours in a given agent environment.
To evaluate the feasibility of using agents in modelling, the study suggested and discussed an agent framework (JADE), a simple enough to implement with reasonably low effort agent architecture, yet powerful enough to sufficiently support agent communication and coordination.

6.1.3 Full Research Question

The two sub-questions having laid a foundation for employing agents in the modelling process, the modelling issues identified and the agent abilities studied, inspired an agent-oriented approach to modelling, aimed at first improving the process in order to improve the resultant model's quality.

In what ways can Agents be used to facilitate the process of information modelling?

In chapter four, an account of the study's establishment of the root causes of issues in the modelling process was given, in descriptions of; translation, ambiguity and formality complexities.
The fact that real-world domains are spread and distributed, meant that the process of modelling them into formal representations was complex enough a problem, to be approached with agents. The perception of the domain as an agent environment, would provide modellers with agent like characteristic behaviours, who according to this work, were identified to be the agents below;

- Domain Expert Agents
- System Analyst Agent
- Intermediate Agents
- Administrator Agent
- Retrieving Agent

All these agents were perceived to have specific tasks, that were only to be achieved collaboratively with support of dedicated communication means, facilitated by a controlled language.
The study was done with caution, avoiding the development of a new controlled language to ensure that any possibilities of introducing more complexity were eluded. And to achieve this, XML was suggested to be used as a meta-language to facilitate communication between the DE and SA, mainly due to its neat, flexible and very well structured format.

Chapter five gave an account of the endeavours, that were involved in designing and modelling the suggested approach in a direction towards its conceptualisation and realisation.
The suggested approach has been demonstrated with the use of; Use Case, Activity, Class and Deployment diagrams, mainly to provide a clear visualisation of the process design, which could provide ways helpful in attaining control over the process, to investigate its efficiency and effectiveness, and to evaluate the benefits and constraits of having a collaborative and cooperative modelling process.

6.2 What was not done

Originally, the work was to provide a prototype implementation of the suggested system, but due to its immense size and level of difficulty within the given short time allowed for the study, it could only yield a system model, that was used to simulate, determine and evaluate, the benefits of approaching modelling with the agent paradigm.

6.3 Conclusions

Agents can contribute positively in improving the quality of the models by introducing more structure to the pattern of activities of the modelling process. The analysis of modelling techniques based on the literature study, revealed that when properly employed the techniques do not compromise the quality of the model, meaning that it is rather those using them and the process through which they are being used, two vital factors that determine how modellin techniques are used, that compromise the model's quality.
The presently applied modelling techniques including agents, can appropriately represent respective domains with sufficient accuracy but since when used they hardly are meant to consider the process through which they are used, the quality of the models yielded still remains poor, leading to poor quality systems. The absence of sufficient collaboration leads to poor quality models, since it hinders opportunities to; utilise modellers' experience during modelling, analyse, and give feedback in form of comparing and contrasting component specification during the process, which would sufficiently facilitate verification and validation.

By exploiting the agent property of social ability, several DE's and SA's are enabled to communicate collaboratively through an exchange of messages over standard protocols. This facilitates decision making in elicitation, abstraction and representation, reducing the translation, formality and language ambiguities and thus yeilding a more consistent, precise and complete model of the domain.

The agent paradigm can facilitate the modelling process by introducing subtasking to reduce complexities, through breaking down the domain into less complex components, that can be tackled one at a time or distributed among agents, where those most knowledgable about a given component are given the task to model it. The idea of letting the modeller most knowledgeable about a component, to take-up the task of modelling it could reduce on the translation and ambiguity complexities related problems, by ensuring that the DE and SA together agree on the definition of a given component based on their knowledge, as opposed to letting the SA individually define the domain based on a translated informal specification from the DE.

Since agents were found to be autonomous, reactive, socially able, pro-active and rational, introducing agency in modelling gaurantees improvements within the process. Because with such properties, the modelling process will be both adoptive and predictive gauranteeing the much needed control over the complexity in the domain.

If modelling is approached this way, elicitation which contains the most uncertainties can benefit from collaborating knowledge, about the attained informal domain specification whose quality depends on factors like; the scope, precision and quality of the questions asked by the SA to the DE about the domain.
As the model grows through stages, because it is being developed with the contribution of everyone in the team, it remains consistent with the knowledge held about the domain and hence is verified and validated through the phases.

A better collaborated process by ensuring that all decision making, specification and any contribution when defining a domain is achieved as a goal of a cooperating team of agents, guarantees ample and sufficient on-time feedback all throughout the process as it progresses.

By being reactive, autonomous and pro-active both the DE's and SA's aim at achieving accurancy in the domain specification and representation. This way, they contineously check on each others' input while comparing with their own respective domain components in order to make sure the domain remains consistent with the model. This enforces compactness in the model and facilitates learning within the process which reduces the domain ambiguity complexity by generating new knowledge.

The need to improve the quality of information systems, that of their models and the modelling process, is undeniable as given in this work and other research works in this direction before. So the main motivation and goal for this study, has been to suggest a fresh approach to the modelling process, in hope that it will effect improvements in the quality of the domain model generated and hence that of the resulting system. The research work has suggested the use of the agent paradigm for this purpose leading to the conclusion that the modelling process indeed can be improved by the use of agents.

6.3.1 Future Research

This work has contended and concluded that agency can be a solution to the problems of modelling but more challenging is the task to achieve what this conclusion states.
But by providing a design model for a possible system structure of an agent-oriented modelling process, the study has provided a stepping stone from which interested researchers can materialise the idea into a realisable solution.
The study also breifly discussed the possibilty of letting the DE model the domain without the SA, this too could be examined for feasibility purposes.

Bibliography

[BN00] Steve Easterbrook Bashar Nuseibeh. Requirement engineering: A roadmap. *ACM*, 2000.

[DM] L.Vaina David Marr. Representation and recognition of the movements of shapes. In *Proceedings of the Royal Society of London B.*

[EH] Thomas Reinke Erika Horn, Mario Kupries. Properties and models of software agents and prefabrication for agent application systems. In *Proceeding of the 32nd Hawaii International Conference on System Science.*

[FB05] Tiziana Trucco Fabio Bellifemine, Giovanni Caire. Jade programmer's guide. Technical report, IBM, 2005.

[GC04] Eric Naiburg Gary Cernosek. The value of modeling. Technical report, IBM, 2004.

[Goo93] Richard Goodwin. Formalizing properties of agents. Technical report, Carnegie Mellon University, 1993.

[Hal] Dr. Terry Halpin. What is an elementary fact? In *First NIAM-ISDM Conference.*

[Hal99] Terry Halpin. Entity relationship modeling from an orm perspective: Part 1. *Journal of Conceptual Modeling*, 1999.

[IEE83] IEEE. Ieee standard glossary of software engineering termilogy. Technical report, Institue of Electrical and Electronics Engineers, 1983.

[JEB02] David C. Brown Janet E. Burge. Nfrs: Fact or fiction? Technical report, Worcester Polytechnic University, 2002.

[MGC92] Kyo C. Kang Michael G. Christel. Issues in requirements elicitation. Technical report, Software Engineering Institute, Carnegie Mellon University, 1992.

[MW95] Nicholas R. Jennings Michael Wooldridge. Intelligent agents: Theory and practice. *Knowledge Engineering Review*, 1995.

[OIL94] Arne Solvberg Odd Ivar Lindland, Guttorm Sindre. Understanding quality in conceptual modeling. *IEEE SOFTWARE*, 1994.

[OS] Gita Sukthankar Vick Mukherjee Onn Shehory, Katia Sycara. Agent aided aircraft maintenance. In *Proceedings of the Third International Conference on Autonomous Agents (Agents'99)*.

[PF05] Th.P. van der Weide P.J.M. Frederiks. Information modeling: The process and the required competencies of its participants. *Data Knowledge Engineering*, 2005.

[SB] Th.P. van der Weide Sander Bosman. Towards formalization of information modeling dialog.

[Ter98] *Handbook on Architectures of Information Systems,*, chapter 4. Springer, 1998.

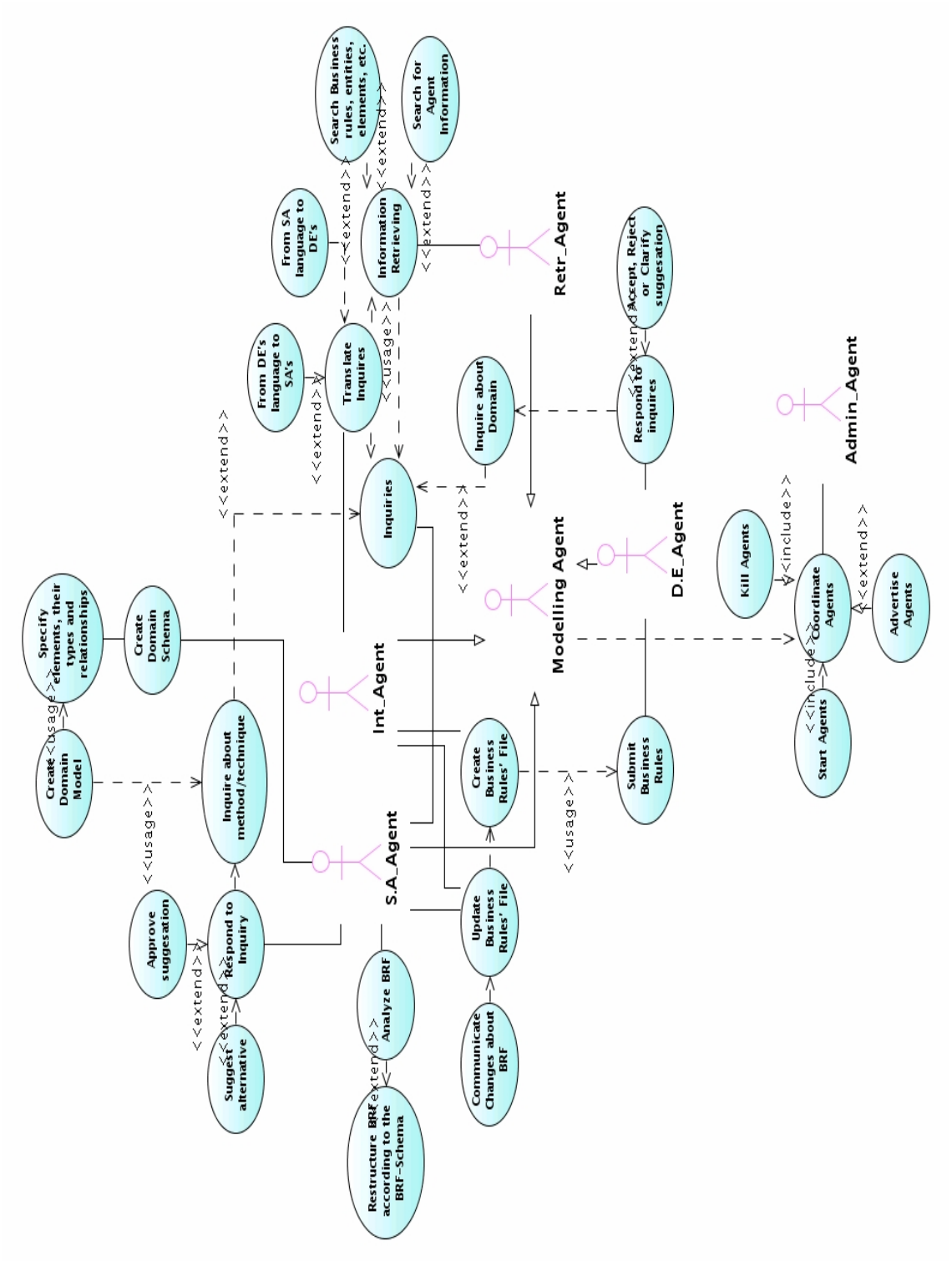
[TF] James Mayfield Tim Finn, Yannis Labrou. 'kqml as an agent communication language'. In *Proceedings of the 3rd International Conference on Information and Knowledge Management (CIKM'94)*.

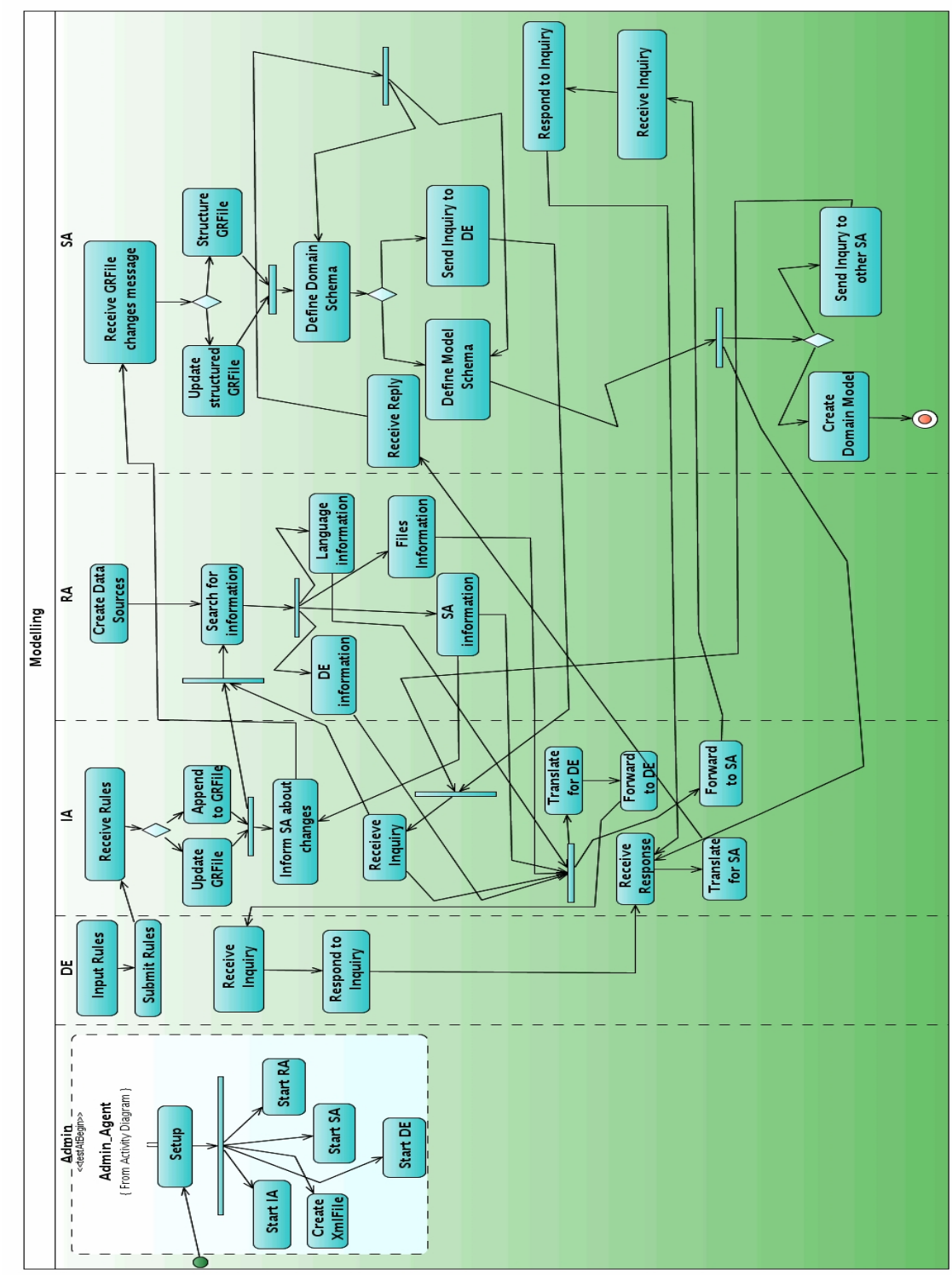
[WR06] Jean Paoli C. M. Sperberg-McQueen Eve Maler Franois Yergeau W3C Recommendation, Tim Bray. Extensible markup language (xml) 1.0 (fourth edition). Technical report, W3C Recommendation, 2006.

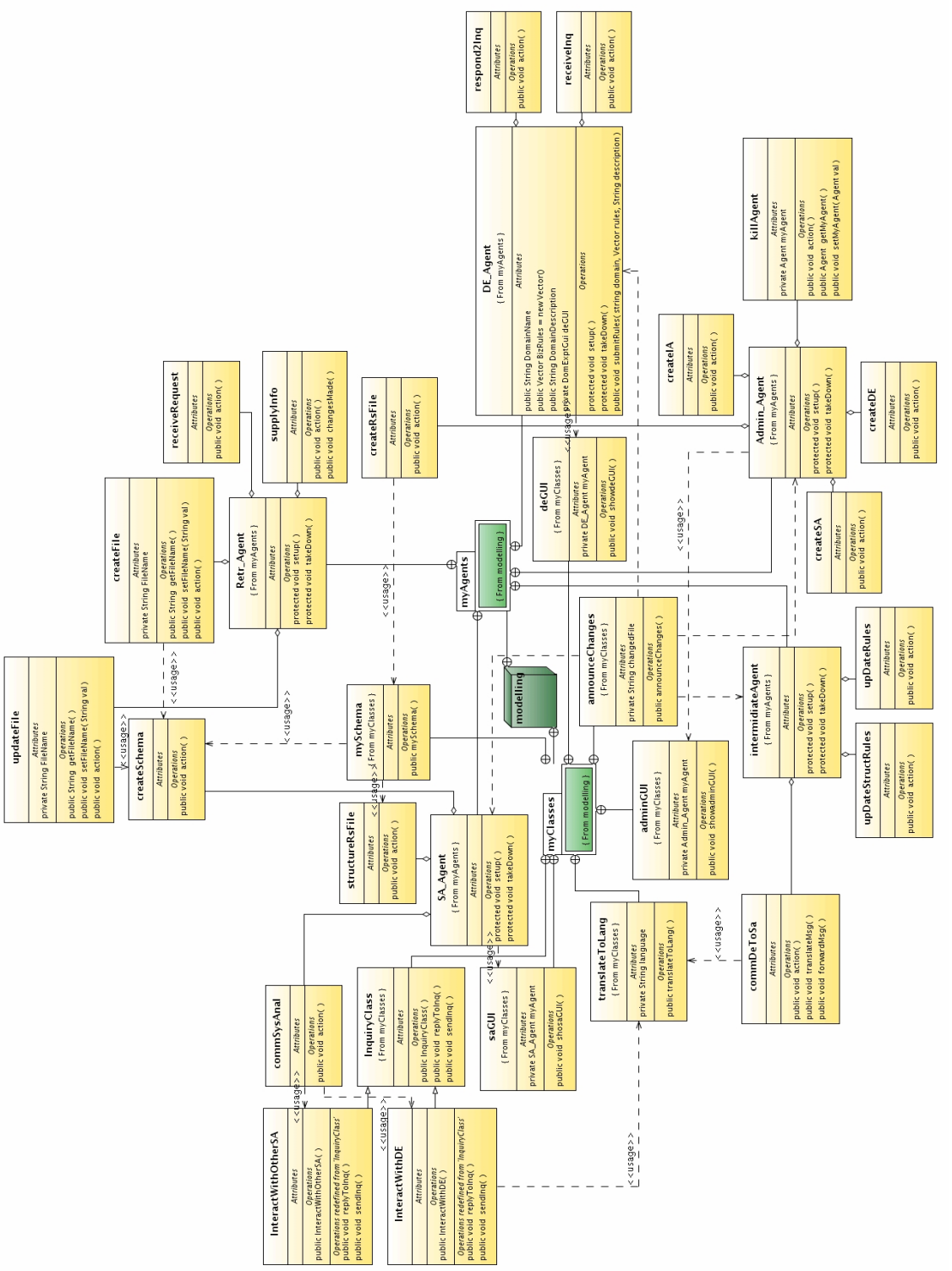
[XML03] *DEVELOPING XML WEB SERVICES AND SERVER COMPONENTS WITH MICROSOFT VISUAL BASIC.NET AND VISUAL C SHARP.NET*. Microsoft Press, 2003.

Appendix A

The Diagrams


Search Business rules, entities, elements, etc.
Search for Agent Information
From SA language to DE's
Information Retrieving
Retr_Agent
Accept, Reject or Clarify suggestion
From DE's language to SA's
Translate Inquires
Inquire about Domain
Respond to inquires
Admin_Agent
Inquiries
Modelling Agent
D.E_Agent
Kill Agents
Specify elements, their types and relationships
Create Domain Schema
Coordinate Agents
Advertise Agents
Int_Agent
Create Domain Model
Inquire about method/technique
Create Business Rules' File
Submit Business Rules
Start Agents
S.A_Agent
Approve suggestion
Respond to Inquiry
Update Business Rules' File
Analyze BRF
Communicate Changes about BRF
Suggest alternative
Restructure BRF according to the BRF-Schema
<<extend>>
<<usage>>
<<include>>


Modelling
Admin
<<testAtBegin>>
Admin_Agent
{ From Activity Diagram }
Setup
Start RA
Start SA
Start DE
Start IA
Create XmlFile
DE
Input Rules
Submit Rules
Receive Inquiry
Respond to Inquiry
IA
Receive Rules
Append to GRFile
Update GRFile
Inform SA about changes
Receive Inquiry
Translate for DE
Forward to DE
Receive Response
Forward to SA
Translate for SA
RA
Create Data Sources
Search for information
Language information
Files Information
SA information
DE information
SA
Receive GRFile changes message
Structure GRFile
Update structured GRFile
Define Domain Schema
Send Inquiry to DE
Define Model Schema
Receive Reply
Respond to Inquiry
Receive Inquiry
Send Inqury to other SA
Create Domain Model


respond2Inq
receiveInq
DE_Agent
killAgent
receiveRequest
supplyInfo
createRsFile
createIA
createDE
createFile
Retr_Agent
deGUI
Admin_Agent
myAgents
createSA
updateFile
createSchema
mySchema
modelling
announceChanges
upDateRules
intermidiateAgent
structureRsFile
SA_Agent
myClasses
adminGUI
upDateStructRules
translateToLang
commDeToSa
commSysAnal
InquiryClass
saGUI
InteractWithOtherSA
InteractWithDE
<<usage>>