

**A REQUIREMENT MODELLING FRAMEWORK FOR REAL-TIME  
MULTI-AGENT SYSTEMS**

A thesis submitted in fulfilment of the  
requirement for the award of the degree

**DOCTOR OF PHILOSOPHY**

From  
University of Technology Sydney

By  
**AMIR ASHAMALLA**  
B.Sc., M.Sc.

Supervisor: Ghassan Beydoun  
Co-Supervisor: Asif Gill

Faculty of Engineering and Information Science  
2017

## **CERTIFICATE OF ORIGINAL AUTHORSHIP**

I certify that the work in this thesis has not previously been submitted for a degree nor has it been submitted as part of requirements for a degree except as fully acknowledged within the text.

I also certify that the thesis has been written by me. Any help that I have received in my research work and the preparation of the thesis itself has been acknowledged. In addition, I certify that all information sources and literature used are indicated in the thesis.

This research was supported by a Australian Government Research Training Program Scholarship.

Signature of Student:

Date:

## ACKNOWLEDGEMENT

I would like to express my deepest gratitude to my supervisor, Professor Ghassan Beydoun, for his endless support and guidance throughout this study. I would also like to thank Dr. Asif Guill from University of Technology Sydney, Professor Graham Low and Dr. Paramesh Nandan, from the University of New South Wales, for their support.

I would also like to thank my wife Marianne Abdelmesih for her support and encouragement which enabled me to accomplish this important milestone in my life and complete this dissertation, even when I almost gave up after 8 years of hard work.

To my daughter Rebecca Ashamalla, son Jonathan Ashamalla and mother Dr Aida Sami, thanks for all your support and encouragement.

Finally, I would like to thank all my teachers, tutors, work colleagues and managers. You have taught me everything I have learned, and your support and guidance have paved the way for me to earn such a prestigious degree and achieve my dream.

Copyediting and related services were provided by Jera Editing Services.

**PUBLICATIONS BASED ON WORK PERFORMED IN THIS THESIS**

1. Ashamalla, A., G. Beydoun and G. Low (2011). Towards agent-oriented approach to a call management system. *Information Systems Development*, Springer New York: 345-356.
2. Ashamalla, A., G. Beydoun, G. Low and J. Yan (2012). "Towards Modelling Real Time Constraints." *ICSOFT 2012 7<sup>th</sup> International Conference on Software Paradigm Trends*. SciTePress Digital Library: 158-164.
3. Ashamalla, A., G. Beydoun and N. Paramesh (2014). "Real-Time Task Attributes and Temporal Constraints.". *AMCIS 2014 American Conference on Information Systems*. Savannah, Georgia, August 2014.
4. Ashamalla, A., G. Beydoun and G. Low (2017). Model driven approach for real-time requirements analysis of multi-Agent systems. *Computer Languages, Systems and Structures Journal (COMLAN)*, Elsevier. [To appear]

## TABLE OF CONTENTS

CERTIFICATE OF ORIGINAL AUTHORSHIP.....	ii
ACKNOWLEDGEMENT.....	iii
PUBLICATIONS BASED ON WORK PERFORMED IN THIS THESIS .....	iv
List of Figures .....	viii
List of Tables .....	x
ABSTRACT.....	xii
Chapter 1 INTRODUCTION .....	1
1.1 Background and Motivation .....	1
1.2 Multi-Agent Systems and Real-Time Requirements .....	2
1.3 Thesis Goals and Significance.....	4
1.4 Thesis Structure .....	7
1.5 Chapter Summary .....	8
Chapter 2 Background and Literature Review .....	9
2.1 Introduction .....	9
2.2 Real-Time Multi-Agent Systems (RTMAS).....	11
2.2.1 Benefits of RTMAS Technology.....	13
2.3 RTMAS Applications and Implementations .....	15
2.4 RTMAS Requirements Engineering .....	18
2.5 AOSE Methodologies and Modelling Languages .....	20
2.5.1 AOSE Methodologies .....	21
2.5.2 Modelling Languages .....	25
2.6 Discussion.....	27
2.7 Summary .....	28
Chapter 3.....	30
RESEARCH DESIGN .....	30
3.1 Overview of Design Science Research .....	30
3.2 Phase 1: Problem Identification.....	33
3.3 Phase 2: Identifying Initial Set of Modelling units .....	34
3.4 Phase 3: Synthesis of the Modelling Units Deployment Process.....	35
3.5 Phase 4: Validation of Modelling units and Concomitant Process .....	35
3.6 Summary .....	36
Chapter 4 Identifying the modelling Units .....	38
4.1 Synthesis of the Modelling Units .....	38

4.2	Call Centre Management Domain .....	48
4.2.1	Call Centre Management Background.....	48
4.2.2	Confirming the Suitability of a MAS Architecture for CMC Requirements .....	53
4.3	Validating the Modelling units in the Requirements Analysis of the CMC MAS .....	55
4.3.1	Call Management Centre Requirements Analysis .....	55
4.3.2	Step 1: Identifying Actors in CMC.....	56
4.3.3	Step 2: Identifying Tasks for Each Actor in CMC.....	57
4.3.4	Step 3: Identifying RT Constraints for Role Tasks in CMC .....	61
4.3.5	Step 4: Identify Agents and Ensuring Tasks Do Not Overload Any Single Agent. 62	
4.3.6	Step 5: Applying the Modelling units .....	64
4.4	Conclusion .....	72
CHAPTER 5 SYNTHESIS OF THE RT MODELLING PROCESS.....		74
5.1	Introduction.....	75
5.2	Meeting Scenarios .....	77
5.3	Modelling units' Integration .....	82
5.4	Modelling units' Dependencies .....	87
5.5	Proposed Process .....	89
5.5.1	Identifying the Modelling units Set for the Process: .....	90
5.5.2	Identifying Candidate Sequences .....	90
5.6	Simulating The Candidate Processes .....	114
5.6.1	Individual Modelling units' Simulation .....	116
5.6.2	Sequential Modelling units' Simulation.....	119
5.6.3	Random Modelling units' Simulation .....	120
5.6.3.1	Random Simulation Tests .....	123
5.6.3.2	Random Simulation Results.....	125
5.7	Conclusion .....	127
Chapter 6 RT Modelling framework in an iPhone Application.....		129
6.1	Introduction.....	129
6.2	Application Requirements and System Goals .....	131
6.3	Integrating RT Requirements within the iPhone Calendar.....	135
6.4	The Pcal Application .....	136
6.5	Application Testing and Validation.....	139
6.6	Results .....	141
6.7	Threats.....	146

6.8	Summary .....	<b>Error! Bookmark not defined.</b>
Chapter 7 CONCLUSION .....		149
7.1	Thesis Summary .....	149
7.2	Thesis Contributions .....	151
7.3	Thesis Limitations and Future Work .....	152
7.4	Concluding Remarks.....	153
Appendix A.....		155
Call Management System SR Diagrams .....		155
Appendix B .....		162
Table 4.2 Sources of Modelling Units .....		162
Bibliography .....		177

## LIST OF FIGURES

Figure 2-1 Real-time approaches .....	13
Figure 2-2 No of publications, identifying how to represent constraints, per year.....	27
Figure 3-1 Research phases.....	33
Figure 4-1 SD diagram illustrating CMC actors dependencies .....	57
Figure 4-2 SR for the outbound calling system agent.....	60
Figure 4-3 SR for the performance monitor agent.....	60
Figure 4-4 SR for the RM, RM after call and RM pre call agents.....	63
Figure 4-5 Outbound calling system agent with the modelling units .....	69
Figure 4-6 Performance monitor agent with the modelling units .....	70
Figure 4-7 RM, RM pre-call and RM after call agents with their modelling units .....	71
Figure 5-1 Meeting attendees travel and arrival times to a scheduled meeting.....	79
Figure 5-2 Attendee “A”, “B”, “T” Travel options.....	81
Figure 5-3 Relationships between real-time constraints .....	89
Figure 5-4 Set 1 representing the first set of 5 modelling units .....	92
Figure 5-5 Set 2 representing the 2nd set of 5 modelling units .....	93
Figure 5-6 Set 3 representing the 3 <sup>rd</sup> set of 5 modelling units .....	94
Figure 5-7 FC1 The 18 real-time modelling units’ process .....	96
Figure 5-8 FC2 The 23 real-time modelling units’ process .....	99
Figure 5-9 FC3 The 12 real-time modelling units’ process .....	103
Figure 5-10 Logic1 The 23 real-time modelling units’ process.....	106
Figure 5-11 Logic2 The 23 real-time modelling units’ process.....	110
Figure 5.12-16 Individual success rate test results for T1-T4 and all test comparison	126
Figure 6-1 Pcal application diagram .....	133
Figure 6-2 Pcal application home screen listing a month’s meetings.....	136
Figure 6-3 Deleting a meeting .....	136
Figure 6-4 Edit meeting screen .....	137
Figure 6-5 Location search screen .....	138
Figure 6-6 Location map display .....	138
Figure 6-7 The framework showing the process and the used 18 modelling units.....	140
Figure 6-8 Pcal application users density per country, as per Google Analytics e.g. 1000 users from USA, i.e. the darker the blue colour the more users per country .....	141
Figure 6-9 The total number of events and their categories, actions and labels.....	142



Figure 6-10 Detailed Google Analytics per Pcal application event.....	145
Figure 6-11 Summarised Google Analytics per Pcal application event .....	146
Figure 6-12 Apple’s IOS 8.1 Travel Time Alert settings option .....	147
Figure 6-13 Apple’s IOS 8.1 Travel Time option.....	147
A-1 Outbound calling system SR diagram .....	155
A-2 SR with modelling units.....	156
A-3 Performance monitor SR diagram .....	157
A-4 Performance monitor SR modelling units diagram .....	158
A-5 Relationship manager SR diagram.....	159
A-6 Relationship manager after split SR diagram .....	160
A-7 Relationship manager after split SR modelling units diagram .....	161

## LIST OF TABLES

Table 4.1: Systematic review results showing number of papers from each source .....	39
Table 4.2: The modelling units and their references *References as per appendix B ...	42
Table 4.3: Feature ratings on the call centre domain .....	53
Table 4.4: Potential agent roles, tasks importance and appropriateness .....	54
Table 4.5: Relationship manager, performance monitor and outbound system tasks.....	59
Table 4.6: Relationship manager real time tasks before subdividing .....	61
Table 4.7: Identifying tasks with time requirements for PM Role.....	61
Table 4.8: Identifying tasks with time requirements for the OCS Role.....	61
Table 4.9: Identifying tasks with time requirements for the RM Role after subdividing	62
Table 4.10: Identifying tasks with time requirements for the RM agent .....	63
Table 4.11: RM time requirements template .....	65
Table 4.12: Performance monitor time requirements template.....	66
Table 4.13: Outbound calling system time requirements template.....	67
Table 4.14: The proposed 23 modelling units' icons.....	68
Table 5.1: Meeting attendee, transport method and meeting agents identified tasks .....	82
Table 5.2: The agents and tasks .....	82
Table 5.3: The 4 modelling units' sets .....	90
Table 5.4: The proposed modelling units' processes .....	91
Table 5.5: Attendee "1" meetings .....	115
Table 5.6: General lecture non-attendees.....	117
Table 5.7: Results when each agent had 1 meeting only .....	118
Table 5.8: Adding more meetings to each agent results .....	118
Table 5.9: IVF and TS effect on the meetings success rate. ....	118
Table 5.10: Results of 99 events on meetings 2-100 .....	119
Table 5.11: Success rate for 223 events .....	119
Table 5.12: The 3 randomly chosen agents.....	120
Table 5.13: Meetings held by the 3 randomly chosen agents .....	120
Table 5.14: The randomly chosen events.....	121
Table 5.15: Total successful and unreachable meetings .....	121
Table 5.16: The 10 random not attending agents.....	121
Table 5.17: The 100 random not attending agents.....	122
Table 5.18: Summarised simulation results .....	125

Table 5.19: The Meeting success rate in resolving calendar conflicts.....	126
Table 6.1: Database fields mapped to existing iPhone calendar fields .....	134
Table 6.2: Pcal application versions.....	139
Table 6.3: Google events modelling units' events hits and percentage .....	143
Table 6.4: Detailed Google Analytics per Pcal application events.....	144
Table 6.5: Summarized Google Analytics per Pcal application event.....	145

## ABSTRACT

Real-time constraints are a subset of abstract temporal constraints, which are a class of constraints that are often placed on real world tasks during a problem-solving activity. Violating temporal constraints can produce consequences of unknown severity. Real-time constraints research is extremely useful in environments that require a high degree of availability and reliability, which are the main characteristics of real-time multi-agent systems (RTMAS). Domains currently using RTMAS include, but are not limited to, rescue systems, scheduling applications, electricity, infrastructure systems, flight control systems, marine systems, automotive systems.

This thesis synthesises a framework to support RTMAS requirements analysis to enhance system design identifying real-time and fault tolerance requirements in the early phase of the software development life cycle. The framework consists of a sufficient set of constraints and an associated process to identify and apply the modelling units. The analysts identify the applicable modelling units during the system analysis phase of the sought RTMAS. A design science approach was applied to construct the framework systematically. The framework was validated incrementally as it was constructed using a call centre case study, a meeting scheduling application and an iPhone scheduling application. These case studies have illustrated that the early identification of the real-time constraints and their even distribution among different agent, significantly reduce the chance of an agent failing. These also enhance the system stability and redundancy by providing an extra level of fault tolerance at the agent and task level, as well as at the overall system level.

## CHAPTER 1

### INTRODUCTION

Software systems can fail when requirements constraints are overlooked or violated. This thesis advocates a model-driven approach to ensure real-time requirements constraints are accounted for in the design of distributed intelligent systems (aka multi-agent systems). This chapter introduces the goals of the thesis and sets the research background.

The chapter is organised as follows: Section 1.1 presents the thesis background and motivation. Section 1.2 presents multi-agent systems and real-time requirements. Section 1.3 presents the thesis' goals and significance. Section 1.4 provides an overview of the thesis structure. The chapter is summarised in Section 1.5.

#### **1.1 Background and Motivation**

With the increased complexity of software systems, software development has become more reliant on model-driven development. Instead of requiring software developers to detail how a system is implemented, in a model-driven approach software models specify the functionality and architecture of the system to be used (Colin, Thomas et al. 2003). Modelling describes or specifies a system and its environment for a specific purpose (Alhir 2003). Software modelling processes typically involve a number of phases, e.g. analysis, specification, design, implementation and testing. Each phase would create its own model (system representation) and bring the software system closer to realisation. Each phase represents the software system from a different abstraction point of view, and collectively represents the system more effectively. For example, the analysis phase in developing multi-agent systems captures system goals, then refines these into agent goals and respective role descriptions. Later in the design phase, goals and roles are further analysed to identify agent tasks and agent classes that are closer to the system implementation (DeLoach et al. 2001). In Object Oriented systems development, where objects are at the centre of the modelling activities rather than actions and logic, layering the abstractions of various phases is done according to three views: Computation Independent Model (CIM), Platform Independent Model (PIM) and Platform Specific Model (PSM) (Estefan 2007).

The class of systems of interest in this thesis is agent-oriented systems or multi-agent systems (MAS). These are systems where “agents” are at the centre of the modelling processes, and these are again typically supported by their own methodologies, e.g.

BEAST (Carrera et al 2014), ICTAM (Elsawah et al 2015), SAVS (Nakashima et al 2014), SODA (Cossentino et al. 2014), PASSI (Cossentino and Seidita 2014), MOBMAS (Tran et al. 2008), AUML (Červenka et al. 2005), secure Tropos (Mouratidis and Giorgini 2007), Tropos (Bresciani et al. 2004), GAIA (Wooldridge et al. 2000) and others. Various methodologies focus on different technologies or phases, e.g. Gaia methodology focusses on the analysis and design of agent-based systems (Wooldridge et al. 2000) while TROPOS introduces techniques for validating early requirements via a model checking approach (Giunchiglia et al. 2003). Moreover, methodologies are typically tightly coupled with one or more modelling language(s) to describe the types of intermediate products generated during the various phases of development. For instance, UML is typically tied to RUP and its variants; i\* (i-star) is typically tied to the requirements analysis of agent-oriented systems. As the use of Object Oriented (OO) languages is an established technology with a quite sophisticated Integrated Development Environment (IDE), there is some debate whether OO languages can actually be used for the implementation phase of agent-oriented systems (Riemsdijk, Mehdi et al. 2006). In other words, the requirements analysis phase is more tied to the abstractions and the modelling paradigm employed than to the implementation phase. This thesis focusses on supporting the requirements analysis phase for the relatively new technology of agent-oriented systems. More specifically, the focus is on supporting the analysis of real-time requirements for agent-oriented systems.

This research also focusses on multi-agent systems, as these systems heavily rely on negotiation, cooperation and coordination between tasks that are distributed among different computer systems. In what follows, the significance of this research is highlighted. The significance of agent-oriented systems is first described. The chapter then highlights the increasing importance of identifying the real-time requirements of agents as early as possible in the development process of agent-oriented systems.

## **1.2 Multi-Agent Systems and Real-Time Requirements**

Multi-agent systems (MAS) are composed of multiple agents. A single agent is a software component that is situated in the system's environment and is capable of autonomous action in that environment. Agents autonomously sense their environment and respond accordingly. Other than the definitional properties of autonomy and situatedness, agents typically interact and cooperate to meet their goals, which need to be aligned with the system's overall requirements (Beydoun et al. 2009). That is, agents are typically designed to meet local objectives as part of the overall design objectives of the distributed

system (this is the MAS that they inhabit). Proper coordination and cooperation between agents that possess diverse knowledge and capabilities underpin the successful achievement of global system goals that cannot be otherwise achieved by a single agent working in isolation (Vincent Conitzer 2007). MAS are often employed when a centralised system solution is not feasible such as coordinating manufacturing (Koji Iwamura et al. 2009), network management (Moon Hae et al. 2003), and electrical load balancing (Broster et al. 2005). Time awareness is part of situatedness and is designed into agents. That is, agents observe time in their interactions and decision-making. Their reasoning processes take time into account and thus, the final outcome of the reasoning process is partially dependent on time (Soh et al. 2005).

Modelling agent real-time interactions has gained focus in the last ten years. Such focus highlights the link between modelling agent interaction and modelling how agents can meet their deadlines, and how to overcome any ensuing agent faults by creating redundancy in the MAS. Agents (and tasks) in these systems might not always be aware of other agent's (and tasks) availability and response times, e.g. task A waiting for task B result, where task B has failed; hence, task A would wait indefinitely unless there is a timeout or notification process in place. Such issues are currently resolved by timeouts. In most cases, the timeouts are fixed times independent of the actual process duration or system load. They are designed to force the application code to return data within this time limit (Zahariev 2009). Notable recent examples of modelling real-time agent interaction in MAS are seen in: The London Underground project (Basra et al. 2007), search and rescue application (Micacchi and Cohen 2008), target tracking (N.A. Sabour et al. 2008), construction management (Zhang et al. 2009), power management (Colson and Nehrir 2013) and online learning (Agudo-Peregrina et al 2014).

Real-time attributes of plans, actions, events and messages are required to model real-time constraints of MAS tasks, e.g. the London underground project (Basra et al. 2007) uses real-time attributes of messages and actions taken by other trains to avoid collision. Other applications include cases such as search and rescue tasks (Micacchi and Cohen 2008), where real-time aspects of actions are used to avoid obstacles in rescuing victims in real-time target tracking (N.A. Sabour et al. 2008). These and other related efforts illustrate that agents can indeed efficiently interact in real-time to re-plan/reschedule required tasks in a way that fits unexpected events or changes in the environment. A principal aim for modelling requirements for real-time systems is thus fulfilling time constraints (Attoui 2000, Garousi et al. 2009). When developing a model

for Real-Time Multi-Agent Systems (RTMAS), the relative priority and deadline of a task are equally important. From this perspective, a real-time agent is “an agent which achieves a maximal set of high priority goals by their deadlines” (Vikhorev et al. 2009). As such, modelling real-time agents requires analysing delay tolerances and assessing alternative courses of action, e.g. an elevator door modelled as an agent has a minimum and maximum time to open or close doors i.e. when a floor button is pressed, the elevator doors do not immediately close as there is a minimal delay required. There is also a maximum time for the doors to close. When this maximal delay is exceeded, an “*alternative course of action*” is to raise some sort of alarm. This is typically in the form of producing beeping sounds and the doors are then forced to close. After the doors close, the elevator has a minimum and maximum time to start moving i.e., the elevator doesn’t move up/down straight after the doors are closed; as it must wait until the minimum time elapses, and then starts moving. If the elevator cannot move and the maximum time elapses, “*an alternative course of action*” is again taken, such as opening the doors once again and stopping the elevator to let people out, declaring the elevator is stuck or out of action. This simple example illustrates how certain real-time constraints would help identify problematic tasks, while other constraints would set “*an alternative course of action*”, thus creating a more robust system. Currently, agent methods do not easily enable analysts to make these distinctions, let alone identify real-time tasks in the midst of requirements analysis and elicitation. This thesis was aimed at filling this gap. Specifically, it provides a list of constructs to assist analysts in identifying real-time tasks and specifying their relevant and critical attributes. It also provides a process that interleaves the use of the constructs in a typical agent oriented system analysis phase.

### **1.3 Thesis Goals and Significance**

The research presented takes the view that the earlier we model real-time requirements in the software development life cycle, the more reliable and robust the resultant system will be. Furthermore, the more likely it is that an appropriate balance between competing time requirements will be achieved.

Surprisingly for systems that are supposed to be decentralised and distributed, a common modelling approach to deal with real-time constraints has been to create a monitoring agent (master agent) (Neto et al. 2009; Attoui 2000). This requires agents to report their status to the monitoring agent, which ensures they are completed within their allocated time. The monitoring agent initiates a redundant task if an agent charged with a



task does not report completing it within the required timeframe (Neto et al. 2009). This clearly presents a single point of failure and is contrary to the decentralisation key feature of MAS and its engendered appeal. This thesis seeks to maintain decentralisation in fulfilling real-time constraints by ensuring that this responsibility is appropriately allocated to individual agents, and this begins with the requirements analysis phase of MAS development.

The thesis aims at providing a modelling framework to facilitate the identification of a sufficient set of activities to be carried out by software modellers determining when a task is said to have failed to meet its real-time constraints. The framework ensures that the tasks that have failed can be distinguished from those that are likely to succeed, even if they are a bit late, e.g. ensuring that the latter ones are provided with more resources, or delaying the dependent task to prevent a cascade of failed tasks. The framework is application domain independent. It depends on two sets of criteria: the first set provides the knowledge to identify the success or failure of the task to meet its real-time constraints; the second set provides the possible set of available behavioural actions. A task taking too long to complete should be regarded as a failed task when a real-time constraint applies. Receiving the right answer too late becomes the wrong answer (Gokhale et al. 2004). Runtime errors and exception handling in the development phase typically require a different set of tasks to be initiated when an error occurs (Westley and George 2004). If a mission-critical task is taking too long to complete, it can lead to unwanted consequences, e.g. dialling an emergency number then having to wait for an hour for it to ring cannot be regarded as successful. The fact that the response time was too long (one hour) means the task has failed, as it did not meet its real-time constraint. This is different from fault tolerance: “the application service must continue even if parts of the control system have failed” (Kopetz 2000), where the latter focusses on the behaviour of the task after reporting a failure in order to start an alternative task to fulfil the application goals. This research focusses on modelling the real-time constraint redundancy during the analysis phase. The goal is not to address fault tolerance issues; rather, this thesis is more concerned with synthesising a reliable and a precise analysis process to ensure that the system modeller captures real-time constraints and the concomitant required agents’ behaviour. This thesis relies on using modelling criteria to identify a set of alternative actions to be taken once a task has been identified as having failed to meet its real-time constraints. This set of behaviour actions can range from logging an error to starting an alternative task. There has been some focus in recent years

on message exchange, negotiation and MAS fault tolerance, while not much has been done on modelling the real-time MAS in the analysis phase.

A more detailed view of the environment is required to specify real-time aspects of the agent activities that need to be taken into account while performing the task. For example, a plan may include real-time constraints on the sequence of actions, and on the duration, the deadlines and the resource states. A real-time constraint on a set of entities is defined by a condition that must be satisfied by the entities over time. Real-time constraints are fundamental to the descriptions of real-time tasks such as: dial a number, wait for the ringtone, or speak. Tasks both individually and collectively must satisfy the implied real-time constraints. For example, dialling a number must finish in a few seconds, and within a reasonable period of time the ring tone must be heard. In addition, each task must finish within a pre-calculated time (Estimated Duration). Violation of these constraints is often not accepted, so it is necessary to specify them explicitly. Accurate identification of the violation of a real-time constraint requires taking into account task dependencies. For instance, task A may be simply waiting for its required input from another task (task B), which is the problematic task (not task A). In the context of agents within MAS, this dependency may be compounded and takes the form of a chain of dependencies of tasks and agent goals (Neto et al. 2009) i.e. all agent features must be considered and modelled with their time-related features (Cabri et al. 2003). This research in essence promotes further context awareness of agents as advocated in (Barbosa et al 2012). To represent the salient features of the environment and the required agent interactions that are relevant to identifying real-time constraints on agent's actions, this thesis emphasises the need to include further support for modelling languages to support RT requirements.

As identifying the problem is the first step towards fixing or avoiding it, the modelling framework developed in this thesis will help developers better understand the problem and give them more insights as to the different aspects and effects of the problem (Selic 2003). This will avoid future problems that might arise as a result of not meeting real-time constraints. This can also assist in identifying bottlenecks, and better distributing workload between agents.

This research not only creates more reliable MASs but also more redundant and more self-healing software. By self-healing software, this means that the MAS, in general, can overcome failures of certain tasks by identifying and rectifying the failed tasks whenever possible. Not all failed tasks can always be rectified, neither do tasks always

fail. Currently, error detection mainly happens in run time using “try and catch” commands (Dalessandro et al. 2007), with no standardised analysis or planning for error identification and correction. This thesis proposes a modelling framework for real-time constraints identification and error correction for MAS. Applying the framework to multiple domains and applications will validate that the framework is not domain or application-specific.

## 1.4 Thesis Structure

This thesis comprises the following seven chapters:

**Chapter 1 *Introduction*:** This chapter introduces the thesis, starting with the background and motivation. It then outlines the contributions of the thesis, and their significance.

**Chapter 2 *Background and literature review*:** This chapter reviews the relevant literature and research in modelling real-time constraints and MAS domains. This also highlights the missing gaps that the thesis fills.

**Chapter 3 *Research design*:** This chapter discusses the research design and methodology followed in this thesis.

**Chapter 4 *Identifying the modelling units*:** This chapter presents the steps followed in conducting a synthesis process to identify the real-time modelling units to be modelled. It then discusses the review findings and the need for a process to apply these modelling units within the software development life cycle, which is developed in Chapter 5.

This chapter also validates the identified modelling units using a call management case study. The validation begins by applying the identified constraints set. It then discusses why the constraints are sufficient, demonstrating the case study models and findings. This thesis validation is spread across three chapters, as it was too long to be included in one chapter and part of the validation was used to develop a synthesis process to deploy the modelling units.

**Chapter 5 *Synthesis of the RT modelling process*:** This chapter presents the process proposed to implement the modelling units. It then validates each modelling unit and the process using a calendar scheduling simulation case study. This simulates user delay for scheduled meetings and how the proposed framework attempts to enable them to attend the meetings, either by taking alternate routes or transport methods to attend a meeting, or by rescheduling the meeting if possible.

**Chapter 6** *RT modelling framework in an iPhone application*: This chapter presents the analysis of a calendar scheduling iPhone application case study, its outcomes and findings. This is a rich domain that highlights key features of any MAS. Highlighting how real-time constraints have been applied for a mobile phone (iPhone) scheduling application and the benefits of applying the constraints in the analysis phase. This will further validate that the set of constraints is domain-independent.

**Chapter 7** *Conclusion and future work*: This chapter summarises the research findings and limitations, then proposes future work to extend this research.

## **1.5 Chapter Summary**

This chapter has outlined the research goal of this thesis, providing a brief background to the research problem addressed. The chapter also presented the thesis structure, contribution and significance to the MAS modelling community in general, and specifically to the RTMAS and the RTMAS modelling community.

## CHAPTER 2

### BACKGROUND AND LITERATURE REVIEW

This chapter reviews the current work related to this thesis, then highlights an existing research gap in RTMAS requirements analysis, which this thesis aims to fill. The chapter starts by looking at related work in the areas of RTMAS, requirements engineering, agent-oriented software engineering, and examines more closely work dealing with the real-time dimension in these areas. It is organised as follows: Section 2.1 introduces the chapter focuses on agents, multi-agent systems and real-time multi-agent systems; these are elaborated in Section 2.2 together with the benefits of real-time multi-agent systems (RTMAS) technology. Section 2.3 describes some of RTMAS applications and implementations. Section 2.4 presents current RTMAS requirements engineering practices. Section 2.5 summarizes and compares several existing agent-oriented software engineering (AOSE) methodologies and their modelling languages. Finally, the chapter is summarized in Section 2.6.

#### 2.1 Introduction

The Oxford dictionary defines an agent as “*a person or thing that takes an active role or produces a specified effect*”. A software agent similarly takes an active role in producing interactions within the environment of the system. This thesis aims at early identifying time constraints for such software agent roles, e.g. how long does it take to complete an active role? which facilities later transform into task time constraints? Currently, time constraints are implemented on tasks, where a “Task usually refers to a clearly defined piece of work, sometimes of short or limited duration” (Kellogg, M. 2016); that is, tasks are required to complete within a certain time period. However, this research proposes identifying these time limits in the early analysis phase, so as to identify constraints for the agents’ overall roles as well as individual task time constraints. To better understand agent role time constraints and the aim of this research, let’s start by stating the agents’ definitions from Botti et al. (1999, 2004, and 2008):

- An agent is a system situated within and part of an environment that senses that environment and acts on it, over time, in pursuit of its own agenda and so as to affect what it senses in the future.
- An agent is an encapsulated computer system that is situated in some environment and is capable of flexible, autonomous action in that environment in order to meet its design objectives.

- Agents are active, persistent components that perceive, reason, act, and communicate.

Now that this thesis defined agents, what differentiates them from other software systems and objects? The main difference can be summarised in the following slogan “*Objects do it for free; agents do it because they want to.*” (Wooldridge and Ciancarini 2001). This is not the only agent characteristic, as the list below allows agents to work individually and independent of each other, yet communicate to achieve certain global goals that cannot be otherwise achieved by a single agent working in isolation (Beydoun et al 2009).

- Agents are situated: they can sense their environment, through sensors, and have a set of possible reactions to be performed through their effectors.
- Agents are designed for a specific purpose or goal, which is a part of the bigger system plan.
- Agents are autonomous: they operate independently to achieve their goals. That is, they control both their own behaviour and their internal state. An agent is not limited to react to external stimuli, it is also able to start new communicative acts of its own.
- Agents are proactive: they do not wait for orders to achieve their goals.
- Agents are reactive: agents follow predefined plans to achieve their goals; if things do not go as planned then agents perform alternative plans and actions.
- Agents are cooperative: they coordinate and cooperate together, by communicating their goals and plans, to achieve their goals.
- Agents are non-deterministic: they do not control their environment; however, many agents can affect their environment by executing certain actions and plans.
- Agents are social: they can interact with other agents to achieve their goals.

Agents are also reflective, which includes being time aware “*When an agent is time aware, it observes time in its decision making and actions. Its reasoning takes time into account, and thus, the outcome of a reasoning process is partially on time*” (Soh et al. 2005). In this thesis, a Real-Time Agent (RTA) is an agent with real-time constraints in some of its responsibilities; by extension, a Real-Time Multi-Agent System (RTMAS) is a MAS with at least one RTA. Systems of this type require the inclusion of real-time representation in the communication process, management of a unique global time, and the use of real-time communication. This definition is adopted in this thesis, as it extends

the agent meaning to include real-time constraints. As such, the next section introduces RTMAS, which is the main focus of this research thesis.

## **2.2 Real-Time Multi-Agent Systems (RTMAS)**

RTMAS are MAS with time constraints. Hence, when agents consider time in achieving their goals (that is, are time aware), then they are considered real-time agents. Within RTMAS, an agent typically is tasked with its own goal derived from the overall systems requirements. Each agent is expected to pursue this goal which is allocated by the system analyst at design time in a manner that would ensure that system goals are satisfied. Agent local actions maximise expected utility in leading closer to achieving their local goals. Agents embody a stronger notion of autonomy than objects, and in particular they decide for themselves whether or not to perform an action or a request from another agent, according to their local goals and likelihood of achieving them. This agent autonomy property suits the decentralised architectural requirements of modern distributed systems, which makes them more suited in tackling the emerging complexities of modern software scenarios (Zambonelli et al. 2005).

Software systems generally, and particularly RTMAS, can have both real and non-real-time data input/output requirements. Hsin-wen et al. (2005) proposed scheduling only the real-time data in any system. Since scheduling both will affect the overall system performance, some real-time data items may miss their deadlines, while Saehwa et al. (2000) grouped all real-time transactions together in one thread and set priorities for them. This facilitated meeting the entire thread real-time deadline, rather than individual task real-time deadlines, thus allowing the system to focus on the goal's deadline rather than sub-task deadlines.

The main obstacle for real-time systems is the response time of the active units and their reaction speed, which is vital for many applications, such as multimedia and distributed systems. Real-time systems are explicitly required to guarantee a response time. In some systems there are physical processes with slow dynamics compared with the execution speed of the command part. On the other hand, there are physical processes with fast dynamics; this means that the process part must have much more stringent temporal constraints for the execution of tasks. This may lead to the temporal suspension of a task in favour of the execution of another higher-priority task. This is called task pre-emption (Attoui 2000). That is, high priority and critical tasks are given more time to complete within their real-time deadlines. Prioritising tasks allocated to agents underpins

successful real-time agents' modelling. If an agent is unlikely to meet its deadline for one of its tasks, for example, 50% of work is completed in 90% of the task's estimated duration, then such a task should be allocated more resources (such as CPU and memory) to enhance its chances of meeting its deadline.

When developing a model for RTMAS, the relative priority of the task should be taken into account, as well as the task deadline. From this perspective, Vikhorev et al. (2009) defined a real-time agent as "*an agent which achieves a maximal set of goals i.e., the largest number of high priority goals by their deadlines*". Task priority is a prominent aspect of real-time modelling, where tasks are given priority based on their real-time requirements (Lisa Cingiser et al. 2001). A priority can be assigned based on earliest deadline first (EDF). If, during processing, another higher priority task with a closer deadline arrives, the system will process the higher priority task first while notifying the lower priority task initiator that the task execution has been postponed, as earlier discussed in task pre-emption. Another technique is time-boxing (Roger Pressman 2010) where, if projects can't meet their deadline, each task in the project schedule is time-boxed by creating a schedule per task from its delivery date (deadline) then stopping any work done on this task once it hits the deadline. Any uncompleted work would be completed later if required, time permitting.

For modelling real-time systems, identifying the real-time interactions between the agents is critical. This requires specifying agents that need to interact in real-time and illustrating how they would interact, which criteria should be met, and how the agents should communicate to re-plan in case of any change in the environment. Only critical real-time tasks need alternatives; if non-critical real-time tasks fail then the system will continue working without them, while if critical real-time tasks fail, then the system could stop or fail as a whole. Creating alternative routes for the critical real-time tasks introduces redundancy in the system, which is another prominent requirements in a RTMAS. In summary, most RTMAS research has followed the following two approaches:

1. Meeting deadlines and finding alternative solutions/plans to implement, when the system cannot meet the deadline; that is, identifying the real-time requirements and constraints, and creating a redundant subsystem that would effectively prevent the whole system from failing in case one agent or task failed to meet its real-time requirements.



2. Modelling real-time agent interaction and communication, which in some cases reflects stationing, while in others reflects the messaging protocols and algorithms needed to maintain balance between the time taken to negotiate and coordinate among agents, and the time taken to actually complete a task. The time taken to negotiate and decide on the best way to complete a certain task or alternative task, plus the time taken to actually complete the task, should in total be less than the real-time requirements for that task (Villaplana 2005). Adding to the above any network delay in message exchange, and the agent response time, it becomes critical to identify which part of the task is required to meet the real-time requirements, and also the number of tasks with real-time constraints per agent as illustrated in figure 2-1.

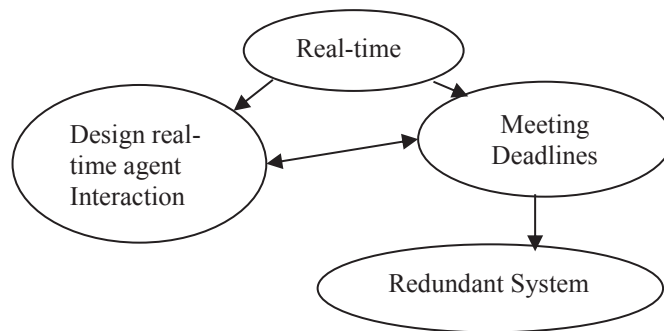


Figure 2-1 Real-time approaches

The two approaches are not mutually exclusive, as responding to a certain stimulus should be done within a certain timeframe. Hence, this thesis focuses on both modelling the interaction between real-time agents and modelling real-time agent redundancy; that is, how the system identifies a failure (real-time deadline not met) and how it will resume working (redundancy).

To put into focus the significance of this thesis, we need to ask what are the benefits of RTMAS and why does this research focus on RTMAS? To answer these questions, this thesis first examines the benefits of the RTMAS technology. Hence, the next section highlights notable features of RTMAS implementations, which together emphasise the importance of such technology and why it came to be the main focus of this research.

### 2.2.1 Benefits of RTMAS Technology

System response time is the primary complaint in many applications. Generally, response time is the time between users performing an input action such as clicking a mouse or hitting a key, and the time the software responds with the desired output or action. System response has two characteristics: response time and variability. A long response time

makes users frustrated and stressed. Variability refers to the deviation from average response times. Low variability allows users to establish a rhythm even if the response time is relatively long, which allows users not to be always wondering if something went wrong (Roger Pressman 2010).

Timing analysis significantly reduces development and maintenance costs, as timing related errors are identified and corrected early in the software development life cycle (Bohlin et al. 2009). Denaro et al. (2004) noted that “*researchers and practitioners agree that the most critical performance problems depend on decisions made in the very early stages of the development life cycle, such as architectural choices.*” Hence, to avoid future performance problems it’s better to tackle them in the early stages of the development life cycle, such as in the analysis phase, using use-cases for example to validate the model. This thesis aims to enhance RTMAS design by identifying the real-time and fault tolerance requirements during the requirements analysis phase enabling the following improvements:

1. Redundancy: The proposed analysis and modelling approach provide system redundancy by creating a highly available system that is capable of self-healing.
2. Reliability: A redundant system is itself a reliable system. This is underpinned typically by the system’s reliability in identifying a failed task. The proposed analysis and modelling approach ensures system tasks would always complete, acknowledging task success or taking an action to overcome any potential failure when possible. This further strengthens system reliability.
3. Avoiding time locks is an important consequence of identifying tasks failing to meet their real-time constraints (deadlines). Time locks are the state when the system cannot execute anything beyond a certain bound. With time locks conditions, verification of correctness properties becomes unreliable (Gómez 2009). The proposed analysis and modelling approach would always pre-check deadlines, allowing enough time to ensure all critical deadlines are met, and avoiding any potential time locks.
4. Monetary savings: Designing RTMAS for qualitative characteristics is more important than for quantitative characteristics, as the cost of hardware (e.g. CPU, memory, motherboards...etc.) is negligible when compared to the cost of lost production, software development, human resources time. However, many real-time systems implementations require more hardware to make the system faster and capable of meeting its real-time requirements (Wolfgang et al. 2004). The

thesis' proposed approach (modelling units set and concomitant analysis process) reduces the lost productivity of real-time systems by reducing their downtime, which translates into monetary saving and increased revenue.

This section introduced RTMAS, highlighting their important features and the benefits of using the RTMAS technology. The next section further focuses on RT systems benefits by discussing prominent and current RTMAS applications and implementations. This is to further highlight their benefits and importance in resolving problems and building such systems.

### **2.3 RTMAS Applications and Implementations**

RTMAS typical applications span different organisational settings: monitoring and analysis (Cheng 2012), e-commerce trading environments (Rodriguez 2003 , DiPipp et al. 2001), evacuation planning (Wang et al. 2012), network robustness and network load balancing, air flight control, mobile robot control (Navarroa et al. 2006; Badano 2008), radar target signatures (Goldman 2000), rescue and systems (Micacchi and Cohen 2008), personal digital assistance (Bresciani et al. 2004), intrusion detection (Shamshirband et al. 2013). RTMAS have often been implemented in the following domains:

- 1) Agents for air traffic control: This is one of the most-used RTMAS fields today, where agents guide planes to land on specific runways, while other agents coordinate the work needed once a plane lands e.g. stairs, luggage carriers, security officers. (Chen and Cheng 2010; Bicchi and Pallottino 2000).
- 2) In scheduling applications, e.g. the London underground "Tube", RTMAS were used to reschedule train timetables based on delays, train breakups, scheduled track maintenance, and other. (Basra et al. 2007).
- 3) Intrusion detection (Shamshirband et al. 2013): This is used as a network security system against hackers and various cyber-attacks, which requires distinguishing misuse and abnormal behaviour.
- 4) Search and rescue: Here, robots can identify objects as fire, victim and obstacle, then create a plan based on their identification. For example, if they identify fire, they would try putting it out; if it's an obstacle, they would go around it; at the same time victims would be rescued. (Micacchi and Cohen 2008). For new tasks, the agent would create its own execution plan and perform the task.

To illustrate the modelling activities in formulating an RTMAS solution and its limitation of not having any real-time modelling units, this thesis examines the search and rescue domain in further details here. The model includes the victim's health where the victim agent identifies the victim's state which could be (rescued, dead or alive), and used three main agents (worker, coordinator and victim):

- I. The worker agent is in charge of the following tasks:
  - a) Worker messages, all communications should be channelled through the coordinator agent. Here, three types of messages are sent to the coordinator (heartbeat, notification and panic).
  - b) Planning by model worker agents is divided into three parts:
    - i. Simple reactive planner; this is to respond to simple actions that do not require much planning.
    - ii. Plan library, consisting of six existing plans that are further broken into three parts (initialisation and generation, execution, consistency checking).
    - iii. Sending the new approved plan to the plan executor.
  - c) Sensors: agents receive periodic updates from their sensors, where update frequency is controlled by the "sensor update cost" option.
  - d) Motors: agents move to the next cell on the desired map cell after confirming they are within range and safe.
- II. The coordinator agent is in charge of the following tasks:
  - a) Models workers, and helps them select other workers, to perform tasks through exchanging the following messages:
    - i. Coordinator messages: does not prioritise messages or tasks, as it assumes they are all important.
    - ii. New mission: messages sent to assign or reassign workers' top level goal.
    - iii. Revoke autonomy: message to take control of the worker to apply a new plan or goal.
    - iv. Restore autonomy: message to release the worker.
  - b) New task: reassigns an already controlled worker to a new task.
  - c) Resume task: stops a worker from working on the current task and resumes working on a previously suspended task.
  - d) Planning: identifies which worker to perform the new task, based on evaluating the candidate workers and selecting one or more to perform the task.

- e) Global state representation: information is divided into worker specific or global, including updated map information and new object discovered.
- III. The victim agent models the health of a victim to be rescued. The victim agent also identifies the victim state, which could be rescued, dead, alive.

Since the current implementation does not provide any awareness to the agents of their real-time constraints, this creates some limitations to the system, e.g. the *priority* to save the victim based on the victim's status is not taken into consideration. This is clearly time dependent, as victims who are in a *critical* condition should be saved first. This can be done by creating deadlines to rescue those victims first. The use of modelling units to identify deadlines in requirements modelling could produce such a system. "hard/soft deadlines are defined by Gasouri et al (2009) as "*A hard RT constraint on an operation enforces that the operation must complete within the specified time frame or the operation is, by definition, incorrect, unacceptable, and usually has no value. On the other hand, in the case of a soft RT constraint for an operation, the value of the operation declines steadily after the deadline expires. Tasks completed after their respective soft RT deadlines are less important than those whose deadlines have not yet expired*". Another modelling unit that could be used during requirements modelling is *checkpoints*. As proposed by Sasikumar (2004) and Roger Pressman (2010), this can be used to save the current search state when the rescue has to stop searching. This allows agents to resume the search from where they had previously stopped. While the task status, as described by Brazier et al. (2000) means that agents have a lifecycle, which changes from migrating, to suspended, then activated state; it is only able to perform actions, when it is in the activated state: the agent is unable to perform actions in the suspended state, as it is waiting and inactive. This raises the question of agent survival, which is how long the agent will keep working; this can be identified by one of two options:

1. The agent has a specific task; once completed it should stop functioning, e.g. once all victims are rescued then the agents should stop searching.
2. The agent keeps monitoring and working in the background to accomplish a set of tasks, when these have been accomplished it stays working, looking or waiting for more tasks to accomplish, e.g. the agent would keep awaiting for further victims to rescue.

As any software development process, a multi-agent system development process starts with requirements gathering and analysis. This thesis focuses on modelling units

such as *priority, critical task, deadline, checkpoint and task status*. that can be identified in the requirements analysis during the development of such time dependent multi-agent systems. It focuses on supporting the requirements analysis phase to ensure that real-time requirements are well identified in advance and well ahead of system design and implementation. As such, it is instructive to examine the current practices in requirements analysis of RTMAS to highlight the research gap on which this research focuses. This is done in the next section.

#### **2.4 RTMAS Requirements Engineering**

The first phase of developing RTMAS is articulating its requirements in order to undertake an appropriate agent-oriented analysis, where a requirements “*is a description of a system property or properties which need to be fulfilled*” (Goknil et al 2011). Requirements analysis leads to understanding the business impact of the software, what customers want, and how end users will interact with the software. Its main objectives are: 1) to describe customer requirements, 2) to create a basis for software design creation, and 3) to define a requirements set to be validated once the software is built (Roger Pressman 2010). The analysis outcome evolves to give a better understanding of the business problem that can be transferred to features and attributes of a software system. This requires developing a common vocabulary and assigning specific meanings to various business concepts (Ghaisas and Ajmeri 2013). Requirements generally consist of functional and non-functional requirements. Functional requirements represent requirements that add value to the system users, while non-functional requirements represent limits, constraints or impositions on the system to be built (Chung and Nixon et al. 2012). In other words, functional requirements describe what the system will do, while non-functional requirements will describe how it will do it. Non-functional requirements include, but are not limited to, system usability, security, simplicity, customisability and adaptability (Chung and Nixon et al. 2012).

The underlying synthesis processes of requirements gathering is encapsulated in the practice of requirements engineering (RE), which is defined as the part of software engineering that is concerned with the identification, expression, validation and analysis of goals and constraints for a software system. RE offers the required tools, methods and techniques to link customers’ desires to systems specifications that can be used to design and build a software system satisfying the stated requirements (Ambriola and Gervasi 1999; Sadraei et al 2007). RE uses a number of approaches for requirements gathering,

like goal-oriented approaches that are used to improve the outcome of RE activities by bridging communication gaps between different stakeholders and developers, and for validating and verifying RE activities (Xu et al 2011; Shen et al 2014; Beydoun et al. 2014; Lopez-Lorca et al. 2011; Lopez-Lorca et al. 2016). RTMAS requirements gathering starts by listing the potential stakeholders (actors), their goals and social dependencies. A goal analysis is then performed to break down the goals to finer goals. A means-end analysis is performed for dependencies relating goals to actors. The outcome of this activity is an artefact that contributes to the requirements documents and design documents.

Requirements evolve during the software development life cycle, hence they might change from the initial specification; changes include additions, omissions and modifications of some initial requirements. Each phase of requirements gathering provides essential feedback to the next phase in terms of the design quality and the design artefact under development. The final artefact is considered complete and effective when it satisfies the system requirements and constraints (Pires et al. 2011). Challenges to RE include dealing with inconsistencies or incompleteness of the requirements specified, which results from information being gathered from multiple sources. Incorrect requirements specification risks the project is cost and quality, usually leading to faulty, expensive software. RE does not, and cannot, lead to a complete requirements gathering if the system is in service and must evolve (Siegemund et al. 2011).

RE research aims at improving and validating requirements gathering; however, it fails in providing sufficient support for requirements metadata, which currently is up to the requirements engineer to define (Siegemund et al. 2011). Examples include differentiating between “requirements conflict” and “requirements inconsistency”. Requirements conflict refers to contradicting requirements where, for example, the fulfilment of one requirements requires excluding another requirements, while requirements inconsistency refers to situations where the co-existence of certain requirements relations causes a conflict, competition, obstruction or clashing with another requirements (Goknil et al. 2011). Current RE deficiencies include the need for higher level abstractions, verification of requirements consistency, undetected conflicting requirements, inadequate identification of requirements relationships, and insufficient requirements knowledge coverage of risks or obstacles. (Siegemund et al. 2011). To overcome some of these deficiencies, researchers proposed a number of enhancements, such as (Miller et al. 2011) proposed withholding design commitments, delaying system

boundary definition and delaying requirements “sign-off”. (Lopez-Lorca et al. 2016) propose using additional knowledge sources (ontologies) to interleave consistency and completeness checks with analysis.

This thesis emphasises the widely-recognised need that non-functional requirements, e.g. time related requirements, should be considered at the earliest stages of the system development life cycle. This facilitates moving towards real-time system design and early timing requirements error detection (Hassine and Rilling et al. 2010).

In the context of MAS, once systems’ requirements are collected, developers require a methodology to develop and implement the gathered requirements into a MAS software system. The next section examines the current practices in agent-oriented software engineering to discuss some methodological advantages and limitations and highlight the significance of this research to the agent oriented software oriented community.

## **2.5 AOSE Methodologies and Modelling Languages**

Developers use one or more methodologies to develop agents, MAS and RTMAS. Methodologies define various modelling languages, steps, techniques and models to produce MAS (Argente et al. 2011). However, there is no single universal methodology, as each methodology has limited applicability, for example, to a specific domain or application (Beydoun et al. 2006; Tran and Low 2008) making them deficient in at least one area e.g. agent internal design or agent interaction design.

Methodologies typically define their abstractions and their work products through a meta-model, which frequently presents a different meaning to each phase (Cossentino et al. 2014; Tran, Low and Beydoun 2006). With an appropriate notation, a meta-model can be used to create a modelling language that can be used to create various agent work products. A methodology also prescribes the processes that developers need to follow to generate the various work products. There have also been efforts to create modelling languages and meta-models without their processes. These modelling languages are also used by developers to support their analysis. In what follows, I describe notable methodological efforts in AOSE and then present several notable modelling languages that have been created without concomitant processes (i.e. those modelling languages do not belong to any specific methodology).



### 2.5.1 AOSE Methodologies

There are few dozen AOSE methodologies in use. Notable methodologies include, but are not limited to, GAIA (Wooldridge et al. 2000), Cassiopeia (Collinot et al. 1996), TROPOS (Giunchiglia et al. 2003), MOBMAS (Tran and Low 2008), O-MaSE (Deloach and Garcia-Ojeda 2014; Garcia-Ojeda et al. 2008), MESSAGE (Garijo et al. 2005), PASSI (Cossentino and Seidita 2014), MOBMAS (Tran, Beydoun and Low 2007), MAS-CommonKADS (Iglesias et al. 1996), Adelfe, Prometheus and INGENIAS (Beydoun and Low et al. 2009), ADELFE, ASPECS, ELDAMEth, GORMAS, INGENIAS-Agile, O-MaSE, OpenUP, ROMAS, SODA (Cossentino et al. 2014), and Radical Agent Oriented Process/Agent Object Relationship RAP/AOR (Wagner and Taveter 2004). This section further summarises and compares some of the notable methodologies highlighting their key characteristics and differences.

GAIA methodology focuses on organisational abstractions, modelling the macro (social) and micro (agent internals) aspects of MAS (Wooldridge et al. 2000). GAIA methodology is best described in Zambonelli et al. (2003) and has been extended to allow implementing the designed models using the Java Agent Development “JADE” framework (Bellifemine et al. 2001) and GAIA2JADE (Moraitis and Spanoudakis 2006), in which the requirements are gathered first, then the analysis is conducted, and where the roles and interaction models are fully elaborated. It is in the design phase that the agent services and acquaintance models are developed; then comes the architectural phase, aimed at defining the system organisational structure. The final design phase specifies the MAS in detail (Zambonelli et al. 2005).

Weyns et al. (2007) recommended including the environment in the GAIA methodology, giving five perspectives to it as an explicit part of the MAS “(1) *the environment as a container and a means for communication*, (2) *the environment as an organizational layer*, (3) *the environment as a coordination infrastructure for cognitive agents*, (4) *Markovian environments*, and finally (5) *task environments*”, where the environment is a first-class abstraction with the following dual roles:

- A. The environment provides the surrounding conditions for agents to exist, which implies that the environment is an essential part of every multi-agent system.
- B. The environment provides an exploitable design abstraction for building multi-agent system applications.

Another widely used methodology is Tropos (Giunchiglia et al. 2003), which is a comprehensive agent-oriented methodology that recognises the interaction between software systems and humans or organisations (Cossentino et al. 2014). Tropos adopts i\* requirements modelling in five phases (Morandini et al 2014; Beydoun et al 2006):

- 1) Early requirements analysis (identifying stakeholders and their intentions).
- 2) Late requirements analysis (identify interactions and dependencies between the system and environment expressed in functional and non-functional requirements).
- 3) Architectural design (detailing the system agents).
- 4) Detailed design.
- 5) Implementation and testing.

While the Cassiopeia methodology, developed by Collinot et al (1996), defines agents in 3 steps:

- 1) Identify the elementary behaviours that are implied by the overall system task.
- 2) Identify the relationships between elementary behaviours.
- 3) Identify the organisational behaviours of the system, for example, the way in which agents form themselves into groups.

Modelling and Analysis of Real-time and Embedded Systems (MARTE), developed by IBM and approved by the Object Management Group (OMG) (Chise et al. 2009), extends the UML by explicitly referencing clocks, for example, the Idle clock (idealClk) (Demathieu et al. 2008), where clocks are classified into three families: coincident-based, precedence-based and mixed constraints (Mallet 2008). The frequently used constraints in these three families are: “isPeriodicOn”, “alternatesWith” and “sampledOn”. MARTE consists of three packages, each of which targets the general, schedulable and performance analysis. MARTE also uses constructs to express non-functional properties, time-related constraints and platforms, which are gathered as part of system requirements yet expressed in a limited natural language. MARTE overcome this limitation by formalising the requirements design models using UML, enabling relating model-driven engineering to real-time and embedded domains (Demathieu et al. 2008). MARTE offers the following four fundamental pillars and notations:

#### Pillar 1: QoS-aware Modelling

- High-level application model “HLAM”: for modelling high-level RT QoS, including qualitative and quantitative concerns.

- Non-Functional Properties “NFP”: for declaring, qualifying, and applying semantically well-formed non-functional concerns.
- Time: for defining time and manipulating its representations.
- Value specification language “VSL”: The Value Specification Language is a textual language for specifying algebraic expressions.

#### Pillar 2: Architecture Modelling

- Generic component model “GCM”: for architecture modelling based on components interacting by either messages or data.
- Allocation model “Alloc”: for specifying allocation of functionalities to entities realising them.

#### Pillar 3: Platform-based Modelling

- Generic resource Model “GRM”: for modelling of common platform resources at system-level and for specifying their usage.
- Software resource model “SRM”: for modelling multitask-based design.
- Hardware resource model “HRM”: for modelling hardware platforms.

#### Pillar 4: Model-based QoS Analysis

- Generic Quantitative analysis model “GQAM”: for annotating models subject to quantitative analysis.
- Schedulable analysis model SAM: for annotating models subject to scheduling analysis.
- Performance analysis model PAM: for annotating models subject to performance analysis.

Other modelling methodologies propose creating an agent manager “master agent” that ensures that tasks are completed on time (Ephrati and Rosenschein 1992). Although the master agent is considered a single point of failure, which is against one of the main characteristics of MAS, it has gained publicity in recent years and started to be accepted in RTMAS. Another model was presented by Zambonelli et al. (2001), which depends on initially broadcasting the agent’s set of tasks and relying on other agents to participate and negotiate this set of tasks. The MAS design is represented as an organisation structure where each agent is identified by its role. The role is what the agent is expected to do in an organisation where each agent can either initiate a change flow or participate in a stage. This gives agents the flexibility to initiate an alternative stage, if it could not complete the one it is participating in, after notifying all other participating stage agents. By having the

initial protocol, there will be no need for a global controller (master agent) as all agents will be notified and will be able to participate in the newly initiated protocol. The aforementioned methodologies might be considered general in their view; however, other than MARTE there are a number of methodologies related to this thesis and targeting real-time systems, such as:

- A. PEARL methodology which has the ability to use software specifications as program prototypes and extend them to fully functional programs (Gumzej et al. 2001).
- B. Real-Time Structured Analysis (RTSA) which is an extension of the traditional Structured Analysis (SA) method. RTSA allows capturing and portioning complex real-time systems into three elements:
  - 1) External top level behaviour.
  - 2) Real-time functional design options, that is, control, timing, and synchronisation aspects of the system functions.
  - 3) Real-time implementation behaviour, that is, response time, delays, queue lengths, and other aspects of the system behaviour as embodied in the hardware, software and human resources and resource architectures (Karangelen et al. 1994).
- C. Rong-he et al. (2009) intelligent system architecture, where in the uncertain and dynamic circumstances a systematic modelling method can be used in the analysis and design of real-time multi-agent systems in particular.

There are also some frameworks which are worth noting, as they are related to this thesis; for example, RADE (Role-based Agent Development framework), which is used when agent goals cannot be directly related to constructs, as a one-to-one relation; hence, there is a need for a theoretical framework to help resolve such complex relationships (Cossentino et al. 2014). Complex relationships are developed due to agents operating autonomously in MAS where each agent has a role in the system's requirements fulfilment. A single agent may take one or many roles, and many individuals may also occupy the same role (Zhang and Xu 2006). RADE applies role-mapping where an agent is considered a part of a system tasked with the overall requirements, while role is defined by the following four attributes:

- 1) Responsibilities: These determine functionality which is divided into two types; liveness properties that represent something good happening and safety

properties that represent nothing bad happening. Maintaining and ensuring an acceptable state of affairs, where a request is always followed by a response (Wooldridge and Jennings et al. 2000; Georgeff and Lansky 1986).

- 2) Permissions: These are the rights associated with a role as resources available for the role to utilise and realise responsibility.
- 3) Activities: These are computations associated with the role that may be carried out by the agent, without interacting with other agents, thus they are private actions.
- 4) Protocols: These define the way agents interact with other agents in their roles.

Another approach similar to RADE is DARX (Marin et al. 2001), where each task can be replicated unlimited times and with different replication strategies. DARX includes group membership management to dynamically add or remove replicas. It also provides atomic and ordered multi-cast for the replication groups' internal communication (Zahia et al. 2001). Furthermore, the PABRE (Franch et al. 2013) framework is designed to reuse requirements patterns by using meta-models that describe the main concepts around each requirements.

Regardless which methodology they use, developers will always require a modelling language to describe and represent the generated work products. As earlier discussed, such modelling languages can also be used independently of the processes prescribed in a methodology. The next section examines some MAS modelling languages, also highlighting the modelling language preferred in the validation component of this research.

### **2.5.2 Modelling Languages**

Modelling is a description of the system and its environment for a specific purpose; hence, a model is the abstract representation of a domain with concepts and relationships between these descriptions and the real world (Othman and Beydoun 2013). Modelling requires a number of skills that span the range from knowing the aspects described in a model to the ability of encoding such knowledge into formal statements (Ghidini et al. 2009). Modelling requires a language to express and represent it, hence this section presents several such existing languages.

There are many languages to model MAS and RTMAS, for example the Agent Unified Modelling Language (AUML), Agent Modelling Language (AML) and i\* (pronounced as i-star). AUML and AML are extensions of the widely used Unified

Modelling Language (UML) (Červenka et al 2005). Agent Modelling Language (AML) defines three modelling concepts to model MASs: agent type, resource type and environment type. The agent type is used to specify the type of agents. The resource type is used to model the type of resources within the system, that is, physical or informational entities. The environment type is used to model the type of the system's inner environment. Zhang (2006) proposed modelling the real-time feature, of multi-agent real-time systems using the Unified Modelling Language, "UML", by extending stereotypes, tagged values, and constraints. This is based on Papasimeon and Heinze (2001) using stereotypes to extend UML for the Jack language (Picard 2003) Stereotypes Definition and AUML Notations for ADELFE Methodology with Open Tool. Hull et al. (2004) modelled real-time embedded systems with UML, by representing DORIS (a method extensively used in the aerospace industry) using UML, where they represented the interaction between different parts of the system by a path end, which represented the path that data would flow in. AUML is not preferred as an Agent Oriented (AO) modelling language, as semantic problems appear when the agent is being a subtype of a UML component or classifier (Beydoun, Low et al. 2009), for example, where UML forces programmers to translate goals into other software notations such as classes, attributes and methods (Bresciani et al. 2004). This research does not have such problems as it uses the agent modelling language *i\** which does represent goals, hence there is no need to translate them into any other software notations. This research also contributes by defining a process for applying the modelling units in the software development life cycle. The modelling units and the process identified constitute a MAS real-time requirements framework. The framework thus consists of an identified set of modelling units and a process representing their checking sequence and their interdependencies.

The *i\** model was designed specifically for modelling agents and MAS by Eric Yu (1995), and has gained acceptance by the modellers. The *i\** model consists of two components: The Strategic Dependency (SD) model which models the different agents and the relationships between them, and the Strategic Rationale (SR) model which models the different tasks each agent has and the different proposed alternatives to accomplish these tasks. In this thesis, the *i\** model is used to represent MAS agents and the relationships between them. Our early requirements phase generates a high-level description of system goals and roles expressed in the *i\** model. The choice of *i\** as a modelling language is based on previous experience (Bresciani et al. 2004) which has shown that *i\** is a good language in which to express MAS requirements. In particular,

the i\* “actor” lends itself to readily modelling the actors and agents in the proposed systems, which comprises a number of Actors (Agents and Roles) (Beydoun et al. 2009). It is instructive to highlight the novelty of the work presented in this thesis from other RT-UML work. When using RT-UML, the authors were limited by the UML tags, guidelines and industry standards, which constrained the ability to extend it with the proposed modelling units, as will be further discussed later in this thesis.

## 2.6 Summary

The literature review has illustrated an important research gap regarding modelling real-time constraints and standardising their identification and rectification process in RTMAS. In particular, despite the fact that a number of researchers have identified how to represent constraints, as summarised in Figure 2-2 and will be detailed in Chapter 4, there has never been, to the author’s knowledge, any work that produces a domain agnostic modelling framework to identify the full set of constraints that would help diagnose or rectify failed real-time agents. There have been notable attempts on this, but they were restricted to a specific domain, in the automotive industry (Konrad et al 2005), or to specific notations (Hassine et al 2010). This is important to ensure quality distributed agent based systems. Ensuring a domain agnostic approach is critical for multi-agent systems, as real-time constraints assist analysts in identifying bottlenecks, and enhance system redundancy and reliability. Modelling real-time constraints can further highlight an agent’s workload: which is beneficial because identifying overloaded agents and redistributing the workload to other agents reduces the agents’ risks of failure.

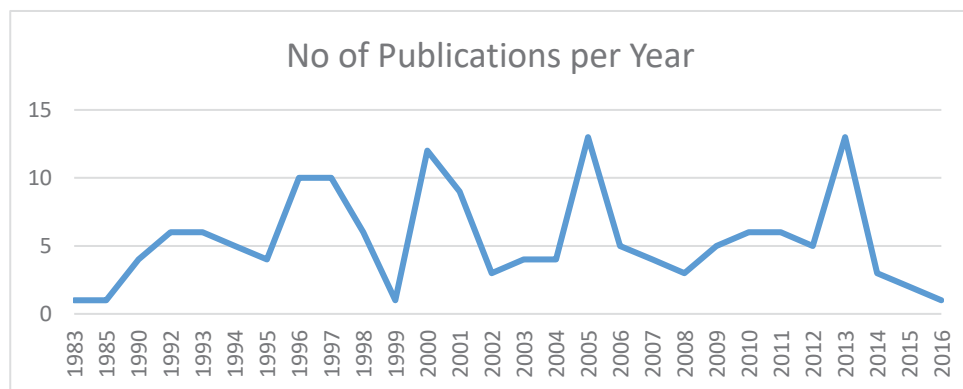


Figure 2-2 No of publications, identifying how to represent constraints, per year

From the above, the importance and usability of real-time constraints is clear; what is also clear is: the non-existence of a comprehensive real-time constraint research and the lack of focus on usability in modelling. This is the gap that this research aims to fulfil through this thesis and is discussed in further details throughout the remaining chapters.

The use of real-time constraints has mostly revolved around creating a redundant system, and not around identifying late or failed tasks. For example, most approaches focus on distinguishing whether deadlines are hard or soft, and how to start an alternative task once a deadline has lapsed. The majority of approaches focus on a single type of constraints or a small group of constraints, as illustrated in Chapter 4. Konrad et al. (2005) and Hassine et al. (2010) were the only researchers noted in the literature who attempted to identify and group the real-time constraints. This thesis is different from their work as they did not model the constraints, and their identified constraints were not meant to identify task failures or recovery actions. Konrad's research was also not related to multi-agent systems, while this thesis focuses on developing a framework of a recommended constraint set and an implementation process to identify task failure and facilitate task recovery for MAS.

The main contribution of this thesis is a framework to support developers in identifying real-time constraints for RTMAS during the requirements analysis phase of development. The framework is model driven and consists of a set of modelling units and an accompanying process to deploy them systematically. The modelling units enhance the analysis tasks of RTMAS requirements engineering. The process improves the completeness of the requirements of the system and its subsequent reliability. This work is critical, as the first step in resolving a failed task is to identify when the task fails. The details of the framework and how it is developed and described in later chapters. However, it is relevant to outline how this thesis fits within the existing research by discussing the following three research aims:

1. To identify a sufficient set of criteria that would help developers, analysts and researchers clearly diagnose and rectify when the task, goal or agent has failed.
2. To develop a process that implements the sufficient set of criteria within the software development life cycle.
3. To validate the research findings.

## **2.7 Summary**

This chapter highlighted the research gap that the thesis aims to fill. First, it presented a background on various RTMAS and their benefits, outlining some RTMAS implementations. Then it discussed RTMAS requirements engineering, AOSE methodologies and modelling languages, highlighting the research gap that this thesis aims to fill in terms of early real-time constraints identification for RTMAS. The chapter



also discussed constraint analysis to ensure avoidance of bottlenecks and agent overloading leading to stable, reliable and redundant systems. This is important for RTMAS as they are very prone to time errors due to their complexity agent dependency, and time sensitivity.

## CHAPTER 3

### RESEARCH DESIGN

This chapter details the design of the research and justifies the structure of the thesis. The research methodology followed is *design science research*. Section 3.1 overviews design science, focussing on design science in information systems research as applied in this thesis. Employing design science yields four phases: problem identification, modelling units' identification, synthesis of a deployment process, and validation of both the modelling units and their deployment process. Each of these phases is discussed in further details in Sections 3.2 to 3.5. Finally, Section 3.6 summarises and concludes the chapter.

#### 3.1 Overview of Design Science Research

Design Science (DS) solves real-world problems by creating innovative artefacts and supporting pragmatic research (Simon 1996 and Cross 2007). DS differs from natural and social science. The latter tries to understand reality, while DS tries to create things that serve humanity (Peppers, Tuunanen et al. 2007). Business managers may also view DS as a way to increase the organisation's profitability, asking questions like "*What can IT artefacts do?*". To answer such a question, artefacts and events that lead to a more desirable IT system are created, which is the main focus of DS research in information systems. This also supports the challenge facing IT professionals: how to describe, design and develop an artefact that would shape a firm's future and increase its profitability (March and Storey 2008).

The research presented in this thesis can be classified as an Information Systems (IS) design science research, as it aims to identify new IS artefacts to facilitate the creation of distributed agent-based systems. The research uses IS artefacts to improve the creation process of real-time multi-agent systems. In this sense, this thesis applies DS as advocated in Hevner et al. (2004). In this IS context of DS, artefacts can be generally defined as constructs, models, methods, and their instantiations enabling representation, analysis and development of new systems. The DS artefacts as used and appearing in this research are:

1. DS constructs: These are words generally formed from the use of natural language to describe and communicate the IS problems and describe their solutions. In this research, they are a standardised set of modelling units that enhance the description of requirements analysis and communication, leading ultimately to developing better real-time MAS.

2. DS models: These are generally a representation, using the constructs, to describe and resolve IS design problems. Models are formed from a coherent collection of the constructs. They facilitate understanding the problem domain and connect the solution to the problem. In this research, models complement existing requirements models. For any given IS development, they are formed from a subset of suitably selected constructs and relations identified in this research. Various representations will be used to express these models, including  $i^*$  (Yu 1995), flowcharts and formal notations.
3. DS methods: These are generally a collection of prescriptive artefacts that together outline some guidelines to resolve problems, creating appropriate models. In this thesis, this will be a process to guide software analysts to select the appropriate modelling units to support their analysis.
4. DS instantiations: These are artefacts created to validate any of the three above artefacts. In this research, the realizability of the modelling units in the requirements analysis of an actual MAS application is confirmed in a call management analysis study. This is the first instantiation of the modelling units. Later, the process to use them is instantiated in a number of applications. Both types of instantiations are used to support the validation and to further refine the DS artefacts created in this thesis (the modelling units and their deployment process).

Building and evaluating artefacts are actually two complementary IS design science activities. Once the DS for IS artefacts are built they should be evaluated as to their effectiveness in how they resolve the identified problem. This is a bi-direction process, as the validation enhances the refinement processes and provides feedback that enhances developing processes to create the artefact. As earlier described, this research aims at standardising a set of modelling units that can be used to identify and represent the MAS real-time requirements. This set will assist in real-time requirements analysis of such systems to enable more effective implementations of them. For instance, if successfully discovered and implemented, such requirements will lead to systems able to identify long running tasks, declare them as failed when necessary, and initiate self-healing and recovery mechanisms. The usage of the modelling unit set naturally requires a supporting process that can be used during the MAS software development life cycle. Both the

modelling units and the associated process to support their deployment need to be sufficiently general to be usable in a multitude of AOSE methodologies.

The research in this thesis follows design science as practiced in IS. The research is organised into the following four phases (as shown in Figure 3-1):

- Phase 1 defines the research problem and its relevance. This was completed in Chapters 1 and 2. The problem is to create a framework to support the requirements analysis of RT-MAS. This can be stated in the following two research questions:
  - o What is a sufficient set of real-time constraints (modelling units) that can be used to support and enhance the representation of extant requirements models?
  - o How to effectively deploy and use the above modelling units in an actual MAS requirements analysis process?
- Phase 2 will identify the initial set of modelling units to answer the first research question above. This phase will be carried out in Chapter 4 of this thesis in the form of a synthesis process. This initial set will then be validated and refined in an actual MAS application (a call management system).
- Phase 3 will develop the process required to systematise the use of the modelling units and enable a software developer to generate an enhanced set of requirements models. This phase will be undertaken in Chapter 5. Multiple processes will be created and simulated. A best-of-breed process will be chosen and refined. This will address the second research question above. In choosing and validating the process, the set of constraints (modelling units) will also be further validated and refined. The threat against domain dependence will also be mitigated, as different domains will be used.
- Phase 4 will further validate and refine both the set of modelling units and the process to apply them. In this phase, two validity threats will be further mitigated against: the first will be against the dependence on the application domain, and the second will be the usability by different software developers. Finally, the

usability and enhanced effectiveness of the RT MAS developed will be illustrated. The remainder of this chapter details each of the above research phases.

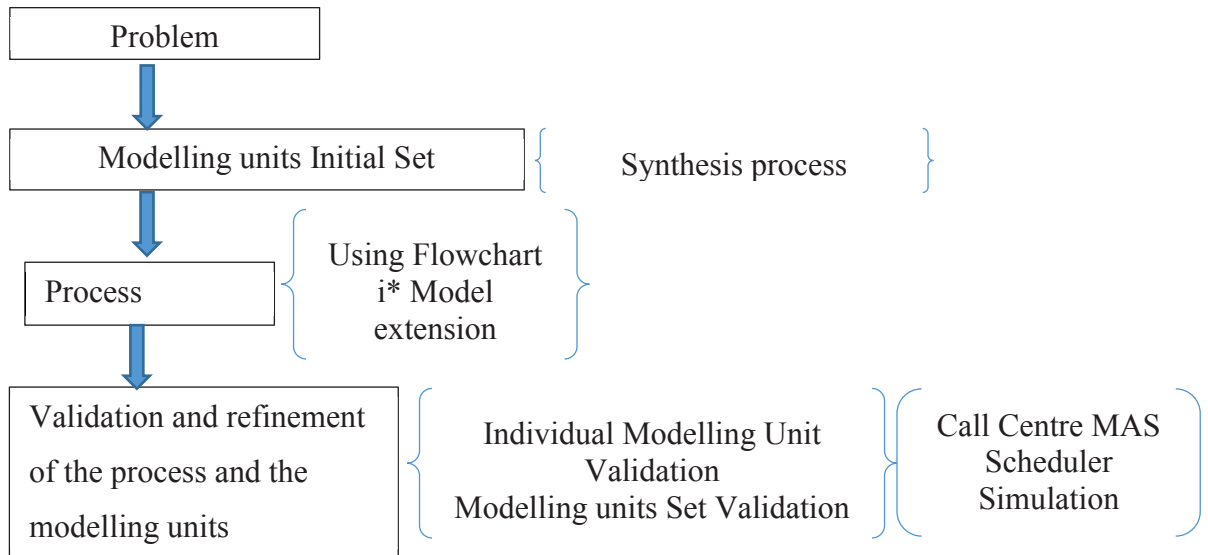


Figure 3-1 Research phases

### 3.2 Phase 1: Problem Identification

The first phase of design science research is generally to define the research problem, which can be elaborated as one or more research questions. This process identifies how relevant the research is to the science body, by highlighting the research benefits/merits and the value added to the science body. In these terms, this thesis research aims to create a framework fulfilling two goals, which correspond to the two research questions described above:

- 1) Identify a sufficient set of modelling units that enhances requirements analysis for real-time multi-agent systems.
- 2) Create a process to enable a software developer to deploy the identified set of modelling units to develop software constructs to manage RT-MAS failed tasks and recover from their failures.

To understand the framed research problem, it is important to highlight the difference between ‘*an acceptable delay*’ and a ‘*failure*’ of a given task. If a person double-clicks an icon, and nothing happens for 5 minutes, then the person expects that something is faulty and considers their double-click to have failed. Most of the time,

double-clicking succeeds on the 2<sup>nd</sup> or 3<sup>rd</sup> attempt. However, in some cases it does not work at all. The obvious indicator of a failure is that it takes longer than expected (rather than some estimated response time). Such an expectation is built on a task-estimated duration, which is based on experience of average time of previous executions. This research aims at including this, and other measures to identify a long running task (failed task), beyond the estimated duration, based on task priority and criticality. For instance, dialling an emergency number is a lot more critical than dialling a friend's number for a social call. Another modelling unit would be, for example, "*how many times can we retry dialling the number*"; if this is the only call an agent can make (e.g. a prisoner making his one and only allowed call), then the agent would generally wait longer for someone to answer the phone.

### 3.3 Phase 2: Identifying Initial Set of Modelling units

To answer the first research question, described in Phase 1 above, a synthesis process was undertaken to identify a set of modelling units that can enhance the requirements analysis gathering process; these modelling units are marked as *the initial set*. The synthesis process is based on a rigorous literature review as advocated in Kitchenham et al. (2009) and also used by other agent modelling researchers, e.g. Kardas (2013).

The synthesis process aims to create an initial set, as presented in Chapter 4. The modelling units set needs a solid basis. In DS terms, these units are constructs used to develop "DS models". The thesis does not arbitrarily recommend or develop its own modelling units. The modelling units are integrated into MAS requirements models to represent RT requirements constraints. The use of the synthesis process partly addresses these research challenges: 1) finding enough modelling units; 2) whether the modelling units be considered a sufficient set; 3) how to validate the units in terms of the reliability of their source and their usability.

The synthesis process presented in Chapter 4 identifies 23 modelling units from verified sources. This initial set addresses the first research question. This set undergoes an initial validation using a call centre MAS domain. In this application, the MAS assists end-customers (EC) by routing calls and allocating calling duties to the most appropriate relationship manager (RM) – call centre workers receiving and making calls. The system accounts for the knowledge/skills and availability of RMs to maximise effectiveness (i.e. improved customer service). This set undergoes further validation at the same time as the deployment process is synthesised and validated. These further validations include

additional applications (a calendar simulation and an iPhone application). The synthesis of the process is first undertaken in Phase 3 of the research, as detailed in the next section.

### **3.4 Phase 3: Synthesis of the Modelling Units Deployment Process**

This phase aims to create a process to deploy and use each modelling unit of the identified set. By definition, a process is a sequence of actions that has inputs and outputs. Each action checks values of the inputs and processes them to produce the designated outputs. The process developed in this thesis guides the developer to identify RT requirements that represent the inputs and express them in terms of modelling units. Ultimately, the output requirements developed will differentiate between late and failed tasks, depending on engendered properties of the tasks. This thesis' aim is to guide the developers to identify and represent the range of constraints for various agent tasks.

The process needs to facilitate the integration of the modelling units into the software development life cycle, especially during the analysis phase, enabling system analysts' better understanding of delays and task status, such as *soft constraints cannot be violated forever*; however, there is a maximum number of times that a constraint can be violated (e.g. a student cannot always be late for a lecture as there would be a maximum number of times that they would be allowed late or absent from that lecture after which an action would be taken against them).

The process needs to be represented to enable easy use, and at the same time the representation needs to be precise without being open to multiple interpretations. For example, a flowchart with formal notations can be used. The process will undergo further refinement in the subsequent research phases as determined by the experimental results.

### **3.5 Phase 4: Validation of Modelling units and Concomitant Process**

This research, as typical for design science research, uses the validation phase to refine and improve the artefacts created, i.e. the modelling units and the associated deployment process. In the validation, different application domains are used to mitigate against any domain dependency. The first preliminary validation focusses on the semantic adequacy of the modelling units. In other words, the validation aims to ensure that RT requirements can indeed be expressed using the modelling units identified in the synthesis process. This validation is undertaken in the context of requirements analysis of a multi-agent system in a call centre to profile and match customers with employees based on specific criteria. End-Customers (EC) receive and make calls to the call centre to receive the service or

product the call centre is offering. The proposed MAS will mix and match the skills and available Relationship Managers (RM) to increase call centre sales, customer satisfaction and profits (Ashamalla et al. 2011). The system routes the calls to the appropriate RMs based on the EC and RM skills, background, demographics and performance.

The second round of validation is simulation-based and exploratory meant, to identify a deployment process. This includes testing subsets of the modelling units set with different sequences. To ensure the exploration is bias-free, random combinations of the modelling units were generated. The random set of combinations represent the usage of the modelling units in different sequence, and of course in different combinations. In this validation stage, the application domain is a MAS to schedule a large group of people attending meetings without scheduling conflicts. Each person has one or more meetings to attend, then the system proceeds with simulating meetings and attendee's events, such as being late, cancelling meetings. This emphasis that as the number of dependencies increases, capturing the constraints using the modelling units becomes critical to the system. This simulation platform is used to evaluate the most effective process that sequences the use of the modelling units to best outcome. The number of meeting clashes is used to determine the outcome of various processes. It should be noted that the meeting scheduling problem can be easily mapped to other problems. This mapping will be discussed in Chapter 5. In other words, using the scheduling problem as a domain to investigate the process of applying the modelling units supports the generality of the outcome.

The third phase of the validation evaluates an actual MAS developed from requirements expressed using the modelling units and the associated process. It is an actual iPhone mobile scheduling application that extends the meetings and events calendar and alerts users when they are running late for their meetings. It has been provided to a number of users, and publicly available on the App Store, to validate the modelling units in day-to-day meetings and events. The application imports all meetings and events from the user's iPhone calendar then extends it by adding the modelling units set.

### **3.6 Summary**

This chapter has reviewed the research methodology, highlighting the design science approach and how it is applied in this thesis. The research is organised into 4 phases:



problem identification and solution proposal, synthesis process, processes development, and finally application and simulation validation.

These phases are sequenced in a way where the output of each phase is further developed and refined in the subsequent phases. The first IS artefact, the first version of the set of modelling units, is developed and refined in Phase 2. However, Phase 3 further refines and validates this while at the same time creates the second IS artefact. The process to deploy the modelling units. In phase 4, The process and the modelling units are validated in three different validations across two different domains. During each validation, the modelling units and the associated process are enhanced. The validation concludes with a successful development of an iPhone application available for public download from the App Store.

## CHAPTER 4

### IDENTIFYING THE MODELLING UNITS

This chapter targets Goal 1 (defined in chapter 1) by identifying a set of 23 modelling units using an appropriate synthesis process based on a rigorous review of the literature.

The work shown in this chapter constitutes Phase 2 of the research described in Chapter 3. It creates the initial version of the first component of the design science artefact that this research aims to produce. The synthesis of this first component is rooted in an evidence-based synthesis process review. The identified modelling units are used to help MAS developers identify when a task is prone to failing to meet its real-time requirements and determine the necessary actions to recover from such a failure. This chapter also provides an initial validation of the modelling units in the requirements analysis of a call centre application. This constitutes a first validation of the semantic efficacy of the set of modelling units. This brings a level of confidence in the modelling units to proceed to synthesising the second component of IS artefact emanating from this research. This will be the analysis process used in grounding the modelling units for a given MAS application. Its synthesis will be undertaken in Chapter 5.

The chapter is organised as follows: Section 4.1 presents the synthesis of the modelling units. Section 4.2 The chapter presents the call centre management domain as a suitable domain for using MAS. Section 4.3 validates the modelling units in the requirements analysis of the call management centre MAS. Finally, the chapter is concluded in Section 4.4.

#### 4.1 Synthesis of the Modelling units

The synthesis process is based on a rigorous literature review as advocated in Kitchenham et al. (2009). As recommended in Budgen and Brereton (2006), the survey of the literature considers the nominated domain in an objective manner, including all relevant arguments, not only the ones supporting the thesis argument. The synthesis process was undertaken using Scopus, Web of science, ProQuest, ScienceDirect and Informit databases covering the last 20 years. The synthesis process was completed between 2010 and 2015. New publications were constantly added as they became available. Research papers which describe real-time modelling units were referenced. Table 4.1 below lists the sources of the papers and a count of how many papers were identified in the source.

Table 4.1: Systematic review results showing number of papers from each source

	<b>Journal Name</b>	<b>ISBN / ISSN</b>	<b># of Papers</b>
A1	<i>ACM SIGPLAN Notices</i>	1581135270	1
A2	<i>Active and Real-Time Database Systems</i>	9783540199830	1
A3	<i>Artificial Intelligence</i>	43702	1
A4	<i>Automata, languages and programming</i>	3029743	2
A5	<i>Collective Robotics</i>	3540647686	1
A6	<i>Computer</i>	189162	2
A7	<i>International conference on Software engineering</i>	1581139632	1
A8	<i>Embedded and Real-Time Computing Systems and Applications</i>	23251271	23
A9	<i>Embedded Systems for Real-Time Multimedia</i>	9781479912858	3
A10	<i>Euromicro Conference on Real-Time Systems</i>	9780769526195	2
A11	<i>Autonomous Decentralized Systems</i>	769510655	2
A12	<i>Genetic and evolutionary computation conference</i>	1595930108	1
A13	<i>Hardware/Software Codesign and System Synthesis</i>	1450307159	2
A14	<i>Computer Communications</i>	1403664	1
A15	<i>Information and Software Technology</i>	9505849	1
A16	<i>Integration, the VLSI Journal</i>	1679260	1
A17	<i>International Journal of Cooperative Information Systems</i>	2188430	1
A18	<i>Object-Oriented Real-Time Distributed Computing</i>	9781467377096	10
A19	<i>International Workshop on Object-Oriented Real-Time Dependable Systems</i>	769523471	11
A20	<i>Journal of Parallel and Distributed Real-Time Systems</i>	7437315	4
A21	<i>Journal of Database Management</i>	10638016	2
A22	<i>Multiagent System Technologies</i>	3642161774	1
A23	<i>Real-Time Technology and Applications Symposium</i>	818685697	1
A24	<i>IEEE Transactions on Computers</i>	189340	1
A25	<i>Symposium on Principles of database systems</i>	897913523	1
A26	<i>NPSS Real-Time Conference</i>	9781467310826	1
A27	<i>Languages, Methodologies and Development Tools for Multi-agent Systems</i>	3642133371	1
A28	<i>Real-Time and Embedded Technology and Applications Symposium</i>	9781467386395	3
A29	<i>Real-Time and Multi-Agent Systems</i>	1852332522	1
A30	<i>Real-Time Applications</i>	818641304	1
A31	<i>Real-Time Systems Symposium</i>	10528725	10
A32	<i>Real-Time Systems</i>	9226443	32
A33	<i>International Journal of Distributed Sensor Networks</i>	15501329	1
A34	<i>Automatica</i>	51098	1
A35	<i>IEEE Transactions on Software Engineering</i>	985589	2
A36	<i>Technical Report. Massachusetts Institute of Technology, Cambridge, MA, USA</i>	7437315	1
A37	<i>The Computer Journal</i>	104620	1
A38	<i>Autonomous agents and multi-agent systems</i>	13872532	3
A39	<i>International Journal of Agent-Oriented Software Engineering</i>	17461375	7
A40	<i>Journal of parallel and distributed computing</i>	7437315	1
A41	<i>Joint Modular Languages Conference</i>	978354045213	1
A42	<i>IEEE Transactions on Industrial Informatics</i>	15513203	1
A43	<i>Information Processing Letters</i>	200190	1
A44	<i>Massively Parallel Computing Systems</i>	818663227	1
A45	<i>ACM Transactions on Embedded Computing Systems</i>	15399087	1
A46	<i>Hard Real-Time Computing Systems</i>	1461406757	1
A47	<i>Principles and Practice of Constraint Programming</i>	3029743	1
A48	<i>Journal of Scheduling</i>	10946136	1
A49	<i>Timing analysis of industrial real-time systems</i>	818670053	1

The synthesis process is 6 steps:

**Step 1 – *Collecting references and initial inclusion criteria*:** The process starts by searching on the keyword: “real-time constrain”. This keyword also covers “real-time constraint”.

**Step 2 -- *Filtering*** The abstract and the title of each search result are used as an initial rough indicator to identify relevant papers. Those are papers that specifically deal with real-time constraints in multi-agent systems.

**Step 3 – *Identifying candidate constraints*:** If a real-time constraint is identified, then it is extracted from the paper. These set of candidate constraints will be detailed and analysed into an operational set of modelling units to be used during the requirements analysis. For each of the candidate constraints discovered in this step, their operational definition along with its reference are noted. Generally, a paper reference can either state a constraint directly by its exact name, or indirectly by a similar name or definition identified from a paper previously encountered. For indirectly stated constraint, a note about the synonymous name and/or the new definition/explanation is also added (the output of this step is shown in Table 4.2). The output of this step is the input for Step 5.

**Step 4 – *Revisiting popular sources*** Literature sources that provided a large number of candidate constraints are highlighted in this step. For these sources, Step 2 and Step 3 are executed again; that is, all publications of the highlighted journal/conference that provided a large number of candidate constraints are visited to identify candidate constraints as per Steps 2 and 3 above. This step identified an extra 33 references. This step is essentially a quality check to ensure that any missed important papers out of these sources which do not fit the original inclusion criteria (i.e. without the keywords “real-time constrain”) are revisited. The output of this step complements the output of Step 3.

Both outputs of Steps 3 and 4 are now the inputs for Step 5. The output is a set of 88 candidate constraints. These are listed in Table 2, with their references and various definitions and/or explanations. The remainder of the process will reconcile these various definitions for constraints of related concerns and will merge them as appropriate into a single modelling unit. This is done in Steps 5 and 6.

**Step 5 – Grouping units.** As noted in Step 3, in some cases similar constraints were used under different names. The first step towards this is to categorise the 88 constraints into groups of similar concerns, based on their names and definition. Each one of these groups is named as a modelling-unit to be included in the final set. In all, 23 groups (modelling units) are identified as shown in the first column of Table 2.

**Step 6 – Reconciling definitions.** This step reconciles the various definitions within each of the 23 identified groups (the output of Step 5). The unification of all definitions within each group becomes the unified definition of the final modelling unit to be used in the requirements analysis. The unified definition in some cases is based a dominant definition. In others, it is based on a number of merged definitions. The influence of an existing definition on the final definition depends on how commonly used the definition is (as per number of references). For example, the estimated definition “*merged definition*” included the “*execution time*” keyword, as it was cited 9 times.

The 23 modelling units, together with the 88 identified constraints and their source references and definitions are presented in Table 4.2 below. While the full references are provided in appendix B.

Table 4.2: The modelling units and their references \*References as per appendix B

Modelling Unit	ID- Constraint	Common Definition and Description Explanatory Text	Source/Reference*
Alternate Task	<b>1</b>	<b>Common Definition:</b> Represents the task to start in case the initial task fails	(1-17)
	2- Execution path	Des. 2: “It extracts the set of possible future <i>execution paths</i> ”	(19)
	3- Neighbouring paths	Des.3: “The earliness of the stimulus also determines the conduction velocity of its <i>neighboring paths</i> ”	(20)
Criticality	<b>4</b>	<b>Common Definition:</b> Is an indication of the effect a task’s failure on the whole system	(5, 12, 13, 21-39)
		<b>Merged Definition:</b> Is an indication of the importance and effect a task’s failure has on the whole system	
	5- Importance	Des.1: <i>Importance</i>	(9, 40)
	6- Weight	Des.2: “We use $W(Sk(t))$ to represent the weight of $Sk(t)$ , which is the sum of the <i>weights of non-critical</i> tasks in the schedule”	(41)
Deadline	<b>7</b>	<b>Common Definition:</b> Is to identify when a task has failed to meet its real-time constraint	(1-3, 5, 8-11, 13, 18, 21-25, 27-33, 35, 40-107)
	8- Maximum time	Des.1: “A <i>maximum time</i> is allowed between the occurrence of a stimulus and the system's response”	(108)
	9- Transduction	Des.2: “A <i>transduction</i> is a mapping from a tuple of input traces to an output trace which is causal, viz. the output value at any timer is determined by the input values prior to or at that time.”	(109)
	10- Discrete time	Des.3: “ <i>Absolute discrete time</i> domain”	(110)
Estimated Duration	<b>11</b>	<b>Common Definition:</b> Is the estimated time that a task is expected to complete within.	(4, 8, 9, 12, 13, 16, 18, 20, 25, 26, 31, 35, 36, 40-42, 47-52, 55-57, 60, 64, 70, 76, 84-86, 88, 95, 97, 98, 101, 108, 111-121)
		<b>Merged Definition:</b> Is the estimated execution time that a task is expected to complete within.	
	12- Waiting time	Des.1: “The request to go to the second floor will be served <i>waiting</i> time units”	(109)
	13- Execution time	Des.2: <i>Execution times</i>	(59, 61, 74, 106, 110, 122-125)
	14- Execution period	Des.3: “The periodic Parameters subclass has attributes like the start and end time and also the <i>execution period</i> ”	(68)
	15- Execution Model	Des.5: “The timing behaviour is often formalised by a two-phase <i>execution model</i> based on action urgency. The phases of execution are: the state of the system changes either by asynchronously executing simultaneous atomic actions, without passage of time, or by letting time pass synchronously for all the components of the system when no action can be performed.”	(66)
	16- TExec	Des.6: “Therefore, we define <i>TExec</i> : $Act \rightarrow R^+$ as a function that associates to each action a positive real number representing the time it takes to execute that action on a target platform.”	(126)

	17- Periods	Des.7: “The algorithm has a pseudo polynomial complexity and handles arbitrary relative deadlines, which can be less than, equal to, or greater than <u>periods</u> .”	(71)
Maximum Miss ratio	<b>18</b>	<b>Common Definition:</b> Is the maximum number of times a soft deadline can be missed	(74, 79)
	19- Upper bound	Des.1: “A weakly hard constraint specifies a guaranteed <u>upper bound on the maximum number of missed deadlines (late messages) during a window of time</u> ”	(25)
Maximum output jitter	<b>20</b>	<b>Common Definition:</b> Is the difference between the best execution time and the worst execution time	(25, 48, 74, 79, 119)
	21- Maximum inter-arrival time	Des.1: “The genes of the chromosome are subject to constraints as two consecutive arrival times for a particular event must have a difference of at least the <u>minimum inter-arrival time, and at most the maximum inter-arrival time</u> (if it exists). If no maximum inter arrival time is defined for an aperiodic task, it is set to $T$ .”	(124)
Minimum time	<b>22</b>	<b>Common Definition:</b> Is the minimum time for a task to complete	(2, 8, 10, 13, 20, 35, 43, 47, 51, 80, 85-89, 91, 108, 110, 120)
	23- Ready time	Des.1: “A task cannot be started before its <u>ready time</u> ”	(41)
	24- Minimum delay	Des.2 “There is a <u>minimum delay of <math>T_{Min}</math></u> between the last event executed in the RB and the first event executed in the CB”	(86)
	25- Predicate	Des.3: “A <u>predicate</u> in a conjunct represents either a delay or a deadline constraint on a pair of events”	(90)
Priority	<b>26</b>	<b>Common Definition:</b> Is the importance of the task	(2, 3, 9, 10, 13, 23-28, 30-32, 40, 44-46, 48, 51, 52, 57, 60, 63, 66, 68, 74-76, 80-82, 88-93, 95, 107, 108, 111-113, 119, 123-125, 127-134)
	27- Influence	Des.1: “The <u>influence of the transaction</u> type. A transaction reads one object and consecutively writes to 1, 3 or 5 objects”	(100)
	28- Weight	Des.2: “We consider the problem of scheduling a set of tasks without pre-emption in which each task is assigned criticality and <u>weight</u> .”	(41)
Periodic Occurrence	<b>29</b>	<b>Common Definition:</b> Is the schedule that the task happens on	(10, 12, 13, 20, 22-24, 27, 30, 31, 40, 47, 48, 51, 55-58, 60, 63, 68, 71, 72, 74-76, 83, 84, 88-90, 92, 93, 97-100, 103, 106, 112-114, 119, 121, 122, 124, 135-138)

	30- Schedulable Period deadline	Des.1: “We also proposed a heuristic to identify a <u>schedulable period-deadline</u> combination.”	(2)
	31- Schedule	Des.2: “Assuming that actions with higher satisfaction values require more execution time, this algorithm ensures that a <u>schedule</u> (if it exists) for meeting the deadline with a minimal satisfaction is made. The actions start executing according to the schedule.”	(9)
	32- Scheduling attributes	Des.3: “A task may have multiple <u>scheduling attributes</u> including <u>periods</u> , execution times, and the blocking times.”	(125)
	33- Timer	Des.4: “ <u>Timer</u> $t \times (y, P; Q)$ is a timer process that waits through channel $x$ for $t$ time units, where $t$ is a natural number. If a name $z$ is received within $t$ time units, it continues to act as $P(z/y)$ ; if nothing is received within $t$ Time units, it changes to be $Q$ ”	(1)
	34- Interval	Des.5: “By definition, event $(E1; E2)$ occurs when $E2$ occurs provided $E1$ has already occurred within some <u>interval</u> .”	(11)
	35- Transmission delay	Des.6: “ <u>Transmission delay</u> between two stations”	(111)
	36- Timing behaviour	Des.7: “Avoid language constructs that have unpredictable <u>timing behaviour (e.g. unbounded loops)</u> .”	(104)
	37- Timed state sequence	Des.8: “Sequence is called compatible with the <u>timed state sequence</u> $r$ . Instantaneous events correspond to singular intervals.”	(35)
	38- Time between two stimuli	“A maximum time is allowed between the occurrence of two stimuli”	(108)
	39- Timed state sequence	Des.9: “Each <u>timed state sequence</u> $r \in \mathcal{T}$ ” represents a system behaviour by identifying a unique system state $T(t) \in E$ with every time instant $t \in \mathbb{R}$ . Formally, a <u>timed state sequence</u> $r$ is a function from $\mathbb{R}$ to $S$ that satisfies the finite-variability condition”	(118)
Retry Attempts	<b>40</b>	<b>Common Definition:</b> Represents the number of times a task is retried/restarted	(24, 36, 50)
	41- On Request	Des.1: “Rescheduling is done when a task is activated due to a scheduled event or <u>on request</u> .”	(80)
	42- Loops	Des.2: “However, since <b>TG</b> may contain <u>loops and/or OR-subgraphs</u> , the release times and the latest completion times of modules needed in Step 3 of <b>MS</b> may not be readily determined.”	(84)
	43- Access the same object	Des.3: “Transaction $A$ attempts to <u>access the same object</u> in a conflicting mode”	(82)
Real-time order	<b>44</b>	<b>Common Definition:</b> Represents time between two tasks	(10, 12, 23, 36, 41, 42, 50, 51, 55, 63, 68, 73, 87, 88, 90, 106, 111, 116-119, 135, 139-141)
	45- Sequences	Des.1: “The set of all <u>sequences</u> of transitions that can be taken. These sequences are called timed action sequences.”	(126)
	46- Scheduler	Des.2: “ <u>Scheduler</u> : scheduler itself (scheduling algorithm). It contains the subclasses PriorityScheduler, RateMonotonicScheduler, and EDFScheduler.”	(68)
	47- Idle/Slack time	Des.3: “These techniques exploit <u>idle</u> and <u>slack time</u> of a schedule. <u>Idle time</u> can be consumed by lowering the processor frequency of selected tasks while <u>slack time</u> allows later tasks to execute at lower frequencies with reduced voltage demands”	(117)
	48- Sequenced	Des.4: “The scheduler has to compute the appropriate values for the deadlines of the <u>sequenced</u> inner Tasks”	(99)



	49- Timed Transition	Des.5: “t k is a <b><i>timed transition</i></b> : a delay used to determine the time the transition must be enabled uninterrupted before firing occurs.”	(111)
	50- Temporal tell	Des.6: “ <b><i>Temporal tell</i></b> consist of telling the start and finish time constraints of the currently executing agent to the temporal buffer followed by an ask of the start constraint from the store”.	(88)
Soft / Hard	<b>51</b>	<b>Common Definition:</b> A hard RT constraint on an operation enforces that the operation must complete within the specified timeframe or the operation is, by definition, incorrect, unacceptable, and usually has no value. On the other hand, in the case of a soft RT constraint for an operation, the value of the operation declines steadily after the deadline expires. Tasks completed after their respective soft RT deadlines are less important than those whose deadlines have not yet expired”.	(2, 3, 5, 12-14, 16, 21, 22, 24, 25, 27-31, 33, 37, 39, 43, 47, 48, 50, 53, 56, 59, 60, 62, 63, 71, 73, 74, 78, 79, 88, 91, 93, 95, 96, 98, 106, 113, 115, 117, 119, 120, 131, 134, 137-139, 142-147)
		<b>Merged Definition:</b> Tasks with hard deadline enforces that the task must complete within the specified timeframe, while a task with soft deadline value declines once their deadline expires.	
	52- HRT transaction	Des.1: “Therefore, two object categories are discerned: <b><i>HRT</i></b> -transactions and <b><i>SRT</i></b> -transactions. The deadlines of the HRT-transactions must be strictly met. The failure to meet a deadline of a HRT-transaction leads to unacceptable transactions. Their executions are periodic. The deadlines of SRT-transactions have a certain probability to be met. When deadlines are not met, only tolerable system degradation is suffered.”	(100)
	53- Criticalness	Des.2: “For example, in a system to initiate trades in a stock market, the timing constraint of a transaction is combined with its <b><i>criticalness</i></b> to take the form of the priority of the transaction. In such a system, the criticalness of a transaction represents the benefit that might be obtained in case of being committed without violating its timing constraints. In other real-time systems which are used to respond to external stimuli (e.g. in autopilot systems) reducing the deadline miss ratio is much more important than criticalness, since an out of date result is useless.”	(32)
Slack time	54	<b>Common Definition:</b> The time in which the execution duration can be increased without failing the deadline	(25, 27, 44, 59, 60, 76, 80)
	55- Timing delays	Des.1: “Timed Buchi automata (TBA). <b><i>TBA</i></b> s are Buchi automata coupled with a mechanism to express constant bounds on the <b><i>timing delays</i></b> between system events”	(116)
	56- Blocking time	Des.2: “The worst-case <b><i>blocking time</i></b> , this is the maximum time a message may need to wait due to a lower priority message on the bus;”	(25)
	57- time elapse before vertex can be triggered	Des.3: “Each edge (u; v) is labelled by an integer parameter p (u; v) denoting the <b><i>minimum amount of time that must elapse</i></b> after vertex u is triggered, before vertex v can be triggered.”	(101)
	58- Timed temporal constraints	Des.4: The <b><i>timed temporal constraints</i></b> define the permissible sequences of state transitions. The time bound, denoted (low, high), specifies that when the rule is ready to be executed, say at time t, then it must/will be executed in the time interval defined by (t + low, t + high).”	(148)

Real-time Dependency	<b>59</b>	<b>Common Definition:</b> Identifies the affected agent if the task fails	(12, 35, 113, 135, 141, 147, 149)
	60- References	Des.1: “An object relation graph that defines the <i>references</i> between the objects involved in the computation”	(106)
	61- Division	Des.2: “Once the <i>division points</i> are identified in the parents, two new children are created by inheriting fragments from parents with a 50% probability”	(124)
	62- Associated	Des.3: “Relevant states (represented by action states) to which timing requirements will be <i>associated</i> , external event sources”	(79)
	63- Mapping messages	Des.4: “By adopting different strategies for <i>mapping messages</i> (events) to threads, we can come up with as many implementation architectures.”	(125)
	64- Flexible	Des.5: “Secondly, because the schedules are built up dynamically through <i>flexible interactions</i> , they can readily be altered in the event of delays or unexpected contingencies. For example, if one of the constituent parts of a composite item is delayed en route to a synchronisation point, it can inform the remaining team members. Together they can then re-arrange the meeting time and adapt their individual behaviour accordingly.”	(6)
	65- Inconsistency problem	Des.6: “The <i>inconsistency problem</i> is highly likely to be there even if the failed program component tries to either cleanse itself of any remaining effects of the failed service execution or complete the service execution after the guaranteed completion time. There are some special cases where the inconsistency can be removed, but such case occurrences are a small fraction of all occurrences of guaranteed service time violations.”	(5)
	66- Logical threads	Des.7: “Identification of <i>logical threads</i> is a three-step process: for each logical thread, our approach identifies (1) its members, (2) priority, and (3) pre-emption threshold.”	(75)
	67- Notify	Des.8: “The application <i>notifies</i> the scheduling service after all schedulable operations have registered. The application can also use the destroy operation to <i>notify</i> the scheduling service when the program is about to exit so that it can release any resources it holds.”	(40)
	68- External Events	Des.9: “We use the event abstraction to specify pre-/post-conditions which allow for recognition of individual events. Further, since events are inter-related, each object's interface description allows for separation of <b><i>those events which are recognised internally from those ones which are external to it</i></b> ”	(150)
	69- Error propagation	Des.10: “If an error inside a component is activated and propagates outside the confines of the component that has been affected by the fault then we speak of <i>error propagation</i> ”	(102)
	70- Relate objects	Des.11: “Object identification procedure, which consists of 5 steps to identify, group and <i>relate objects</i> .”	(151)
	71- Client	Des.12: “A server that missed its deadline during method execution triggers off a timeout and notifies it to a <i>client</i> . The client then can cause a timeout.”	(94)
	72- Related	Des.13: “Objects can, be <i>related</i> in two ways: (1) syntactically: they have a common object from which they are invoked, and (2) time-wise: the actions on the objects always occur at the same moments.”	(100)
73- Origin	Des.14: “Explicitly specie the <i>origin</i> of each timing Constraint”	(104)	
74- Propagate	Des.15: “ <i>Propagate</i> the occurrence of an event on a processor <i>to others</i> .”	(90)	
75- Relate	Des.16: “The functions and <i>relate</i> inputs and outputs on the real-time axis to inputs and outputs on the logical axis.	(152)	
76- More than one	Des.1: “For one activity <i>more than one agent</i> may be approached”	(18)	

	77- VM-Shadow	Des.17: “One type of system support that may be necessary is a <i>VM-Shadow</i> type of region so that the system can retain the previous state of the shared data to be used if the transaction is aborted.”	(112)
Task Status	78	<b>Common Definition:</b> Representing the current state of the task	(17, 64, 120, 141)
	79- Current state	Des.1: “With OCL, it is already possible to check the <i>current state</i> of an object, using the pre-defined type OclState and the operation oclInState”	(19)
	80- State Transition	Des.2: “The <i>state transition</i> diagram depicts the states the control process can be in,”	(34)
Warning	81	<b>Common Definition:</b> Represents the time to sample the task performance	(18)
Composite	82	<b>Common Definition:</b> Contain is a list of simple timing requirements that are imposed at the same time	(74)
Validity Duration	83	<b>Common Definition:</b> Is the maximum time the data can be held for before expiring or being considered invalid	(45, 114)
Remaining time	84	<b>Common Definition:</b> Identifies the remaining time till the deadline is reached	(98)
Real or not	85	<b>Common Definition:</b> Identifies if the task is time dependent or not	(67)
Execution Accrued Value	86	<b>Common Definition:</b> Measures the amount of time gained to the system	(62)
Instant Value Function	87	<b>Common Definition:</b> The total accrued value of a job which is equal to the area corresponding to the instants allocated to executing the job	(62)
Check Point	88	<b>Common Definition:</b> Represents a point where task results can be saved	(5, 24, 76, 107, 133)

## 4.2 Call Centre Management Domain

Using a call centre case study, the usability and semantic adequacy of the identified modelling units are validated in the remainder of this chapter. The modelling units will be further validated in Chapters 5 and 6, during the validation of the deployment process which will be developed in Chapter 5.

For the purpose of the validation in this chapter, this section first overviews the *Call Management Centre (CMC)* domain. The description was sourced through interviewing domain experts. This was done while the researcher worked in a call centre. The analysis results are then evaluated to confirm that the domain is suitable for a MAS architecture.

### 4.2.1 Call Centre Management Background

Telephony remains an essential and efficient way of business communications. Beyond a one-to-one communication tool, it has become a tool for marketing, gathering information, purchasing, selling and recently advertising. Generally, business telephony needs are either outbound calls to customers (e.g. telemarketing products) or inbound calls (e.g. for customer support, sales handling or enquiries). Companies favour outsourcing their call management to dedicated Call Management Centres (CMC) since they tend to have the latest telephone technology and equipment, together with additional value-adding software. The CMC's specialised personnel and training saves the client's company time and money. A typical CMC may have a number of corporate clients (e.g. banks, insurance companies) and a few thousand relationship managers (RM) attending to phone calls to end-customers of its corporate clients. The operating cost of a CMC includes the relationship manager salaries and the call costs. The shorter the inbound/outbound calls, and the less outbound calls a relationship manager makes to achieve a sale, the more profitable a CMC.

CMCs can be hosted anywhere in the world with calls often transferred and routed across countries and continents. The call centre industry is one of the fastest growing industries (Golpelwar 2015), with the demand for call centre personnel expected to continually grow. Salary and training cost represents 60-80 % of an overall call centre's operations budget (Jayashankar 1998; NoahGans et al. 2003). Hence, it is imperative that the effort made by these personnel is targeted and effective. In other words, the employee (RM) with the most knowledge about a given product, with most suited communication skills, and with most appropriate availability is the one who should make or receive a

service call to/from an end-customer (EC). Matching an RM to a customer can be complicated by the dispersed geographic location of the call centre. An RM and a customer are often in different countries and across different time zones. An RM often requires additional communication skills tempered by cultural and geographic sensitivities in addition to product knowledge (Ashamalla et al. 2011, 2012, 2014, 2017). Using Multi-Agent Systems is proposed to assist in customer relationship management by routing calls and allocating calling duties to the most appropriate relationship manager (in terms of knowledge/skills and availability) to maximise effectiveness.

This thesis envisages a call management MAS consisting of distributed intelligent agents supporting the relationship managers and knowledge-based agents monitoring the call centre operation, ensuring balanced workload allocation to the agent. These agents would ensure the best match between a customer and a relationship manager, and monitor of the whole system in terms of customer satisfaction and call throughput per relationship manager. The MAS will thus help the CMC make better use of its personnel and equipment while providing a high value service to its clients and end-customers.

This thesis proposes a system to perform real-time monitoring of the CMC while relationship managers are performing their sales and to adjust the call flow rate to each relationship manager according to specific criteria to be described in this section. These specific criteria include time constraints, priority (the potential of the call), criticality (in performance level), sampling, alternative action(s), deadline. Our proposed system aims to provide dynamic call flow control for both inbound and outbound calls. It will be a distributed intelligent system: which will monitor the performance of relationship managers in real-time. Their performance is sampled every 10 minutes and if any critical or high priority issues are detected an error is logged and the relevant supervisor is notified. The system will provide assistance to relationship managers in serving their end-customers (or potential customers) and if a customer is not served within 3 minutes then calls are prioritised to resolve critical situations where too many calls are left unanswered if there are insufficient relationship managers to answer the calls. The system should result in a higher rate of sales per call made/answered. This section describes recent CMC-related research which deals simultaneously with both monitoring the performance of the relationship managers and matching them with end-customers. It is this kind of overlap that this thesis wants the call management system to achieve. The research aims to provide a dynamic matching capability of the system that changes as products and end-customers change.

A CMC operation is complicated by the varying number and nature of products offered by its corporate clients. Much work has been done on customer relationship management and appropriate matching with customers, based on relationship manager performance and product knowledge. For example, in selling travel packages on behalf of a travel agency, a CMC would do well in matching end-customers to well informed relationship managers with appropriate knowledge about the destination and its traditions. A typical relationship manager matching technique is segmenting customers into social and cultural segments according to their postcodes and surnames (Webber 2007). Supporting tools to create customer profiles exist, for example, see Larue et al. (1999). A corresponding relationship manager profile may depend on the age, sex, culture, language proficiency, experience and product knowledge. The proposed intelligent system will be used as a skill matcher between end-customers and relationship managers based on their profiles. This makes relationship managers more convincing to the customer and increases the chance to achieve a sale. In targeting potential buyers with outbound calls, the system dials numbers automatically every 3 minutes and allocates to a relationship manager, according to a customer target list previously loaded. The system allows for 2 minutes wait for potential customer to respond to non-critical calls, and 5 minutes for critical calls to high priority customers. If no answer is detected for an outbound call, the call is re-routed for a call back scheduled at a later date/time. Once a call answer is detected, the call is routed to the matched relationship manager. The relationship manager is expected to answer the call within 1 minute, otherwise the call is marked as unanswered, and the relationship manager performance is degraded reducing the amount of calls rerouted to them in the future. Marking an outbound call as unanswered and degrading a relationship manager performance is not enough, as the end-customer receiving a call from the call centre without anyone to speak to remains problematic. Hence such unanswered calls are further re-routed for a last time to another available relationship manager, whom if he/she doesn't answer within 30 seconds, then the call is dropped and an apology is played to the end-customer apologizing for the inconvenience the call might have caused. Once the relationship manager answers the call, the system retrieves the end-customer's details from the database, displays the details and provides the relationship manager with a script to use and guidelines to help in providing an adequate service to the end-customer, within 30 seconds otherwise the relationship manager will have to ask the end-customer for his/her details and search for them manually. Once the relationship manager answers a call, voice recording must also be

started and only end when the call ends. As all calls must be recorded for legal and security purposes, this is a very critical task which takes priority over any other tasks and must be completed within a hard 10 second deadline. Once the call ends and voice recording stops, the call outcome is detected within 10 seconds and logged towards the relationship manager performance matrix. At the end of every day all calls and sales are counted and voice recordings are analysed. The results are all added to the relevant relationship manager performance matrix. Each relationship manager performance matrix is then used to rank him/her in determining the amount of calls to be received in the future, where critical end-customers take priority to be routed to the best performing relationship managers and vice versa. Similarly, for outbound calls, the proposed solution will create a specific calling target list for each relationship manager and product based on his/her performance matrix skills and profile.

Many companies profile relationship managers before they are hired or during the staffing process to locate them according to their skill/profile to different areas of the call centre. Psychometric tests are carried out for new employees during the hiring process to enable matching with customers (Doe 2007). In the most basic versions of such tests, an interviewee has to tick the relevant answers on a questionnaire, a process that takes about ten minutes. Based on their answers, a profile and skill matrix is generated. For example, outbound relationship managers need to be extroverts with an ability to generate excitement and handle rejection, while inbound relationship managers need the ability to listen and solve problems.

Tools to profile employees during the initial staffing phase (e.g. Call Centre Simulation (Doe 2007)) which is commonly used to reduce the turnover rate of relationship managers. It is assumed that these provide initial relationship manager profiles for the system. The solution will dynamically adjust according to a relationship manager's performance. It will assess human interactions in real-time to dynamically adjust its criteria. It will continually evaluate the relationship manager's skills and the match with an end-customer as the sale/call progresses. It will recreate the relationship manager's calling target lists and routes calls to him/her based on the latest skill/profile evaluation.

For Inbound calls, customers dial a number reaching the CMC which has its own private automatic branch exchange (PABX or PBX). A call routing and distribution routine that minimizes inbound call costs by reducing per-call handling time is illustrated in Beydoun et al (2014). A skill score is calculated based on the relationship manager's

previous call duration and profile. A process named scoring, where each end-customer is given a score from 1-10 based on the likelihood to purchase the product is performed by classifying customers according to some preloaded criteria. The more likely the customer is to engage in a sale, the higher the score. Customers with the highest scores are served first, moving to lower scores till the end of the calling list. Skills-based routing (Thomas Robbins 2006) calls are routed to a relationship managers based on skill level or profile matching. In addition, the schedule of dialing end-customers and the estimates the call duration of each call vary according to the relationship manager's skill level and previous calling history. There is a variance between the different skill levels in one company or even within a team. For a call centre, this would make a difference when predicting calls. The work in (Thomas Fisher 2003) and (Zeynep Aksin et al. 2007) attempts to predict the calls in a multi-skill environment. In another work (Gary et al. 2000), the schedule of calls is based on a skill matrix for the relationship manager's skill-based routing, based on multiple priority skill levels. The Genesys system (the system that receives and dials the numbers for both inbound and outbound systems) (Genesys 2009) has a skill level for each relationship manager according to which the calls are routed, the higher the skills level the more calls that relationship manager receives. In the proposed system, this skill level will be automatically calculated by the agent system and matched to the skill level of the end-customer. Using a MAS this variance in skill level can be equalized to a certain degree using collaboration.

Inbound customers can be directed to an Interactive Voice Response unit (NoahGans et al. 2003) prompting them for options. The more advanced units may even ask for call reasons in a few words and then redirect the call to an Automatic Call Distributor routing the call to the first available appropriate relationship manager. Customers may hang up when they suffer from a long wait time (NoahGans et al. 2003). Call centres that use toll-free services pay out-of-pocket for the time their customers spend waiting. CMC cost can be reduced by reducing this time. This can happen by providing customers with more automated services that serves them without the need to talk to a human relationship manager thus saving the company a lot of expense and wasting the customer's time, and in some cases wasting a sales opportunity by customers hanging up and dropping their calls (NoahGans et al. 2003). Call recording and automatic analysis for various cues on effectiveness of relationship manager will be incorporated in the proposed system.



#### 4.2.2 Confirming the Suitability of a MAS Architecture for CMC Requirements

To ensure the suitability of a MAS as an architecture for the call management system, developed suitability framework was recently developed (Beydoun, Low and Bogg 2008; Beydoun, Low and Bogg 2013). The framework evaluates the applicability of a MAS solution to the particular problem. The framework has two steps. The first step identifies key features (or requirements), highlighting how appropriate a MAS solution might be in satisfying each of these features. The prominence of the features is also rated. If features rated as important (rating 4 or 5) are matched with a high level of appropriateness of a MAS solution (4 or 5), then a MAS solution is deemed highly suitable. For example, if the environment is dynamic and unpredictable, this is a strong indicator of MAS suitability, as MASs are suitable for such environments. Applying the first step of the framework to the proposed call management domain, the solution is found to be operating a dynamic, distributed, open environment with software components operating remotely (see Table 3). These are characteristics (according to the framework) that suggest the suitability of a MAS. This is especially true given that there will be a lot of negotiation between the solution components, and moreover these components need to work independently and remotely which makes autonomous agents particularly appealing.

Table 4.3: Feature ratings on the call centre domain

<b>Feature</b>	<b>Appropriate (1-5)</b>	<b>Prevalence of the requirements in CMC</b>	<b>Importance (1-5)</b>
Environment – Open	2-3	The environment is open, there is no limitation on the number of end-customers or usage profiles that can be created for both Relationship managers RMs and end-customers.	5
Environment – Uncertain	3-4	There is no guarantee that RMs will match the end-customer, some end-customers might not have a matching profile so the closest match should be provided.	5
Environment – Dynamic	5	RMs change rapidly as the company has a high turnover rate. New end-customer lists are provided by the main Customer for the call centre to call on their behalf.	4
Distributed – Data	5	Data is distributed between a database and a calling on the Dialler (e.g. Genesys system) which dials the numbers then connects to the platform providing a key to retrieve all End-customer information from the database, this is for outbound. For inbound the End-customer calls in and then the data is retrieved from the database, if present, and the call is transferred to the RM.	5
Distributed – Resources	5	Resources are distributed and include client computers with application, client Telephone, client application, client operating system.	5
Distributed – Tasks	5	Distributed tasks include sending emails, faxes and files to the main customers, Receiving and making calls from/to the end-customers. For the proposed solution,	5

Feature	Appropriate (1-5)	Prevalence of the requirements in CMC	Importance (1-5)
		there are distributed tasks like profile matching, profile analyser.	
Interactions – Negotiation	5	There should be negotiation between the components to negotiate the best matching profile to route the call to.	5
Reliability	5	In assumption, the agent profile matcher should be reliable to accurately match profiles to facilitate sales and increase the conversion rate (number of sales made to number of calls).	4
Concurrency	4	Predicting RM call ending, profile matching, profile analyser and performance monitor agents will be working concurrently for more than one RM and end-customer.	4

The second step in the framework focuses on the nature of the tasks required within the system and examines the potential suitability of agents for these tasks, Table 4.4. It examines the main tasks, performance measures, type of interaction between entities, task resources, and entities that execute the tasks. A rating is assigned according to the appropriateness of using agents (1-5) and importance of the task (1-5) based on this measure. Table 4.4 shows that all tasks with importance rating of 5 have potential agent attributes (agency measure) of 3 or more. This indicates that key system tasks can be decomposed in a way suitable for allocation to autonomous agents.

Table 4.4: Potential agent roles, tasks importance and appropriateness

Tasks	Task Inputs	Task Resources	Agency (1-5)	Importance (1-5)	Potential Agent Roles
Monitors the RMs and keeps track of their service time patterns.	Call outcome and duration	Call duration and outcomes	5	5	Performance Monitor
Estimates call duration and the number of incoming calls	Average incoming calls per hour of day, Number of RMs available	Call duration and available RMs	3	5	Load Balancer
Transfers calls to appropriate RM according to the client's preferences and RM availability.	Call start and end	Call routing to RMs	2	5	Router
Receive voice responses from end-customers and routes calls based on their selection	end-customer response	Workflow, end-customer voice, played messages.	2	3	IVR unit
Creates profiles for RM, end-Customers and products	end-customer, product and RM details.	RM details from HR, end-customer and product details from main customer	5	5	Profiler

Tasks	Task Inputs	Task Resources	Agency (1-5)	Importance (1-5)	Potential Agent Roles
This is the agent responsible for matching between product, end-customer and RM	end-customer request and profile; RM's availability and profile; available products.	RM, product and end-customer profiles.	5	5	Matcher

As indicated by the first step of the framework (Table 3), many requirements of the system point to the suitability of MAS for a call management system. This was confirmed with the second step of the framework which showed that many of the system tasks can be allocated to suitable agents requiring a degree of autonomy. In the next section, an undertaken requirements analysis is explored, using stakeholder analysis technique with *i\** which has been extensively used for MAS design (Bresciani et al. 2004).

### 4.3 Validating the Modelling units in the Requirements Analysis of the CMC MAS

The validation presented in this section constitutes a preliminary validation and a stepping stone prior to the development of the concomitant process in Chapter 5. The process will enable the analyst to decide which modelling unit(s) to be used for a given agent task and will also provide a sequence of the modelling units' execution later at runtime. In this current validation, all units will be treated equally as the process is yet to be synthesised.

#### 4.3.1 Call Management Centre Requirements Analysis

Initially, RE activities are performed using the *i\** modelling framework in (Yu 1995). This begins with stakeholder requirements analysis and rationale for the new system. In a MAS, agents depend on each other to achieve system goals and perform tasks. The stakeholder analysis represents the MAS agents and the relationships between them. This produces a high-level description of system goals and roles expressed in *i\**. The resultant model consists of two components: The Strategic Dependency (SD) model which models the different agents and the relations between them, and the Strategic Rationale (SR) model which models the different tasks each agent has and the different proposed alternatives to accomplish these tasks. Other goal-oriented languages such as KAOS (Bradshaw et al. 1997; Hiroyuki et al. 2006) and AOR (Wagner 2000) could be used instead of *i\**. However, various experiences with *i\** (Bresciani et al. 2004; Tran et al. 2008) has shown that it is a good language to express MAS requirements. The *i\** "actor"

construct lends itself to readily model the actors and agents in a CMC. For the purpose of this validation of the modelling units, the RT analysis process applied consists of the following five steps:

- Step 1: Identify agent roles (actors): This step produces an SD diagram, to enable the actors' identification.
- Step 2: Task analysis: For each identified actor, a task analysis is undertaken producing an SR diagram that identifies tasks for each role. This step produces a list of tasks for each agent role.
- Step 3: Identify and refine RT tasks: For each list of tasks, filter through which tasks are RT tasks; that is, identify any which have RT constraints. For the RT tasks, revisit whether or not they can be sub-divided. This enables the analyst to further zoom in on the nature of RT constraint that needs to be identified.
- Step 4. Revisit RT task allocation: Ensure that the allocation of RT tasks does not overload any single agent. This may require splitting some roles into two or more roles.
- Step 5: Revisit RT modelling units' allocation: For every identified RT task, for every agent role (including newly identified roles), check if and how each of the 23 modelling units is applicable.

The Alternate Task "AT" modelling unit can lead to identifying a new task which was not identified in step 2 initial task analysis. In such case, the identified new task will go through steps 3-5 once again with the 23 modelling units applied to it if applicable.

The details of each of the above five steps is presented in what follows:

#### 4.3.2 Step 1: Identifying Actors in CMC

This step aims at identifying system stakeholders, which are represented as agent roles (actors) and their goals (Desired state). An agent role is defined as an abstract characterisation of the behaviour of a social actor within some specialised context or domain of endeavour. Its characteristics are easily transferable to other social actors. Dependencies are associated with a role when these dependencies apply regardless of who plays the role (Yu 1995). Their goals are then analysed, refined and delegated to existing or new actors. This process ends when sufficient goals have been delegated in a way that all actors fulfil their assigned responsibilities and goals (Giorgini et al. 2005).

The i\* model corresponding to the proposed MAS identifies nine agent roles (actors) as shown in SD diagram in Figure 4-1.

The SD diagram (Figure 4-1) represents a starting point for all subsequent task analyses yielding SR diagrams. In all, the output of this step shows the 9 identified actors (as shown in Figure 4-1). They are as follows: inbound calling system, load balancer, voice recorder, matcher, end-customer, main customer, outbound calling system, relationship manager and performance monitor. This chapter will only focus on representing the last 3 agents (outbound calling system, relationship manager and performance monitor) while all other agents are further detailed in Appendix A.

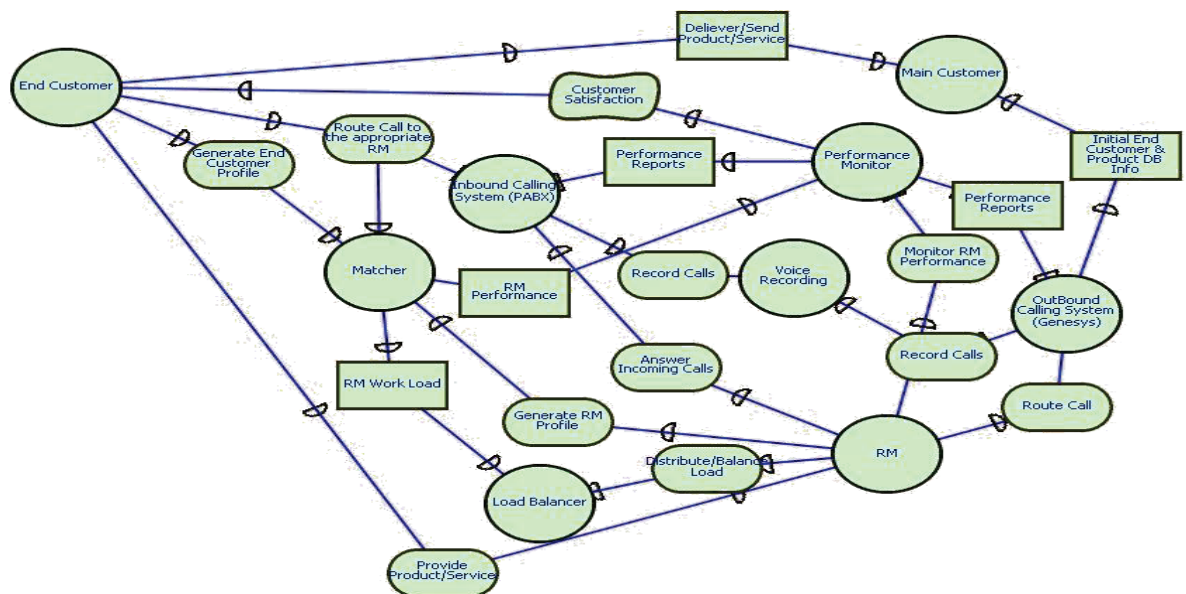


Figure 4-1 SD diagram illustrating CMC actors dependencies

#### 4.3.3 Step 2: Identifying Tasks for Each Actor in CMC

This step undertakes task analysis for each of the 9 actors identified in Step 1. Task analysis defines how a goal is accomplished in terms of undertaken activities. Goals are abstracted from the domain, including trade-offs between multiple ways to accomplishing them, where each one represents a task and/or subtask. MAS's tasks might be distributed across multiple roles, reflecting their interdependencies between these roles. Agents may also simultaneously contribute to achieving multiple or single goals (Prasad and Lesser 1999).

For each of the identified tasks, for every actor, how each task time requirements are identified will be detailed. In this section, I show the results of the task analysis for outbound calling system (OCS), relationship manager (RM) and performance monitor

(PM). Their tasks are shown in Table 4.5. The result of the tasks analysis of each of the PM, OCS and RM are respectively shown in SD diagrams in Figures 4-2, 3 and 4.

For the Performance Monitor (PM) actor the identified tasks are as follows:

1. *Count calls made.*
2. *Count sales made.*
3. *Analyse performance: This checks if there are any trends in the RM's call logging or performance, ex the RM logging all their calls as call backs or no sales.*
4. *Analysis of voice recording to analyse if the RM is saying or doing something during the call that can be enhanced to increase his/her sales, as speaking too fast or too low, speaking without passion or giving a bad impression for the product/service from his/her voice tone.*
5. *Analyse call outcomes: This is to analyse if the RM has a trend in logging his/her calls.*
6. *Generate RM performance reports.*
7. *Improve customer satisfaction: for example, using results of analysing voice recordings, the number of call backs done to each end-customer and work load on each RM.*
8. *Calculate RM load.*
9. *Determine best/worst product.*
10. *Determine Best/Worst performing RM.*
11. *Monitor Performance.*

For the *Relations Manager (RM)* actor, the identified tasks are as follows:

1. *Confirm customers' details.*
2. *Offer product/service.*
3. *Read script provided: Once the RM is on the call, he/she should be reading from the provided script.*
4. *Answer customers' questions.*
5. *Log call outcome.*
6. *Call back.*
7. *Personal call back: This is when the RM believes that they can make a sale with the end-customer; in this case, they would keep the end-customer details in a personal call back to be able to call him at the set date/time, to carry on with the sale.*
8. *Sale: This is when the RM completes a sale with the end-customer.*

9. *Do not call (DNC): This is when the End-customer chooses not to receive any more calls from the call centre/RM. The call centre has one month to block his number from being called again.*
10. *Insert call details: At the end of each call the RM has to create a call report where they put all the call details and notes on why they had to log the call as they did.*
11. *Create sales/customer reports.*

For the *Outbound calling system (OCS) actor*, the identified tasks are as follows:

1. *Dial number.*
2. *Detect call answer.*
3. *Start call recording.*
4. *Detect available RM.*
5. *Route call to matched RM.*
6. *Retrieve script.*
7. *Detect Call outcome.*
8. *Stop voice recording.*
9. *Reroute unanswered calls.*
10. *Reroute call for call back.*

Table 4.5: Relationship manager, performance monitor and outbound system tasks

<b>RM Tasks</b>	<b>PM Tasks</b>	<b>OCS Tasks</b>
Confirm customer's details	Count calls made	Dial number
Offer product/service	Count sales made	Detect call answer
Read script provided	Monitor performance	Start voice recording
Answer customer's questions	Analyse performance	Detect available RM
Log call outcome	Analyse voice recording	Route call to matched RM
Create sales/customer reports	Determine best/worst RM	Retrieve end-customer details
Answer calls	Analyse call outcomes	Retrieve script
Send sales confirmation	Generate performance reports	Detect call outcome
Add call history	Improve customer satisfaction	Stop voice recording
Call back	Calculate RM load	Reroute call for call back
Insert call details	Determine best/worst product	

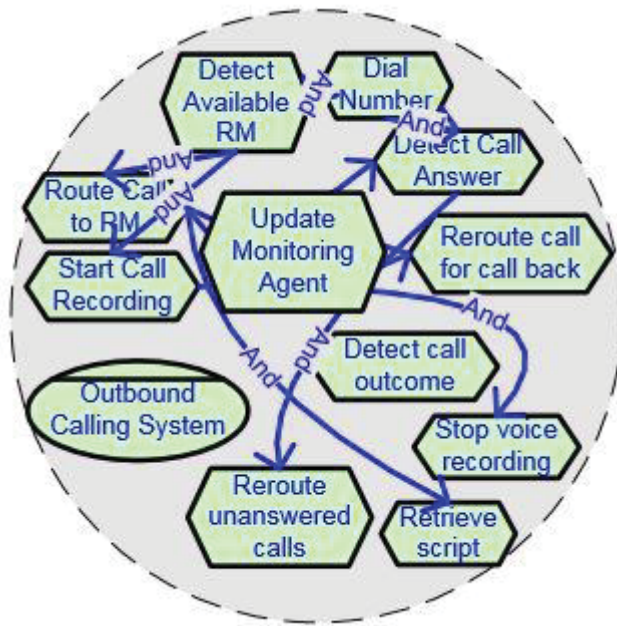


Figure 4-2 SR for the outbound calling system agent

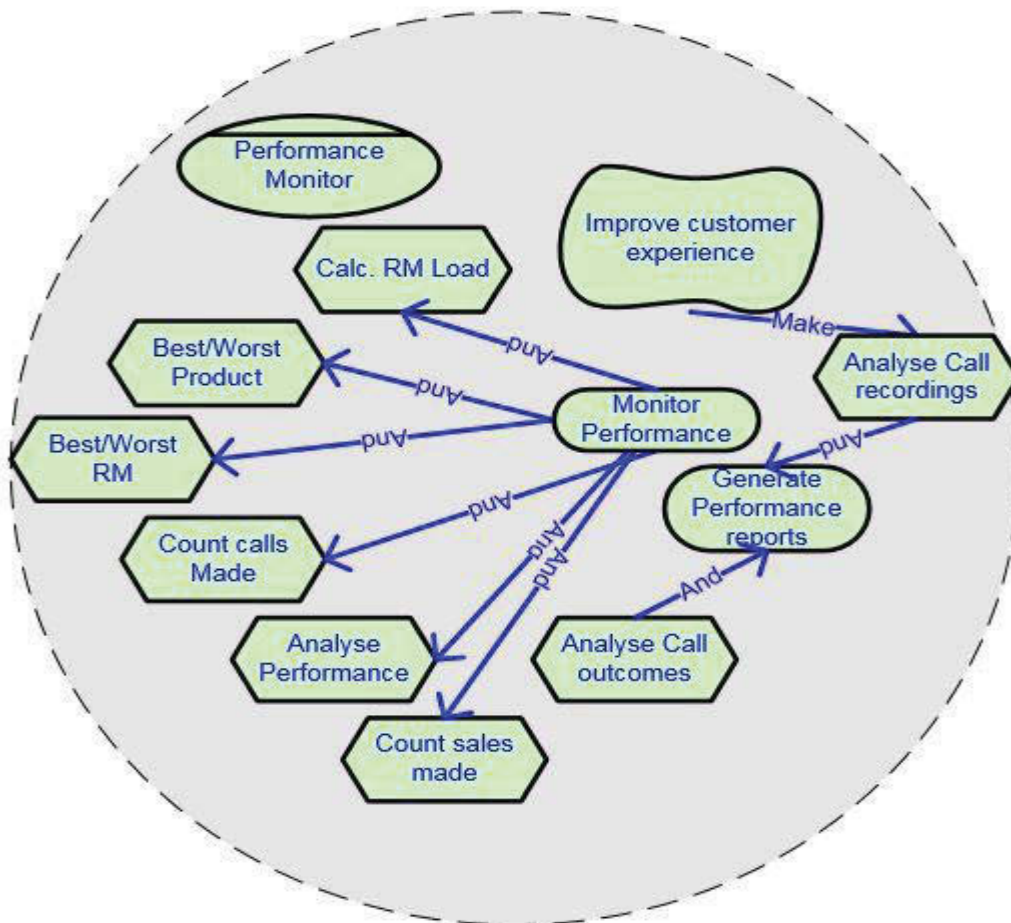


Figure 4-3 SR for the performance monitor agent



#### 4.3.4 Step 3: Identifying RT Constraints for Role Tasks in CMC

This step first involves identifying which tasks have any time requirements. This is basically asking “*is the task required to be completed within a specific time frame?*”.

Each Roles’ tasks and their RT constraint are summarised per Tables 4.6, 7 and 8.

Table 4.6: Relationship manager real time tasks before subdividing

Tasks	RT
Confirm customer’s details	X
Offer product/service	X
Read script provided	X
Answer customer’s questions	X
Log call outcome	√
Create sales/customer reports	√
Answer Calls	√
Send sales confirmation	√
Add call history	X
Call back	X
Insert call details	√

Table 4.7: Identifying tasks with time requirements for PM Role

Tasks	RT
Count calls made	√
Count sales made	√
Monitor performance	√
Analyse performance	√
Analyse voice recording	√
Determine best/worst RM	√
Analyse call outcomes	√
Generate performance reports	√
Improve customer satisfaction	√
Calculate RM load	√
Determine best/worst product	√

Table 4.8: Identifying tasks with time requirements for the OCS Role

Tasks	RT
Dial number	√
Detect call answer	√
Start call recording	√
Detect available RM	√
Route call to matched RM	√
Retrieve script	√
Detect Call outcome	√
Stop voice recording	√
Reroute unanswered calls	√
Reroute call for call back	√

The second part of this step is to do the following: For each RT task, revisit whether or not they can be sub-divided. This enables the analyst to further zoom in on the nature of RT constraint that needs to be identified. As a result, the RM tasks are subdivided into multiple subtasks as summarised in Table 4.9.

Table 4.9: Identifying tasks with time requirements for the RM Role after subdividing

<b>Tasks</b>	<b>RT</b>	<b>Tasks</b>	<b>RT</b>
Confirm customer's details	X	Update monitoring agent	X
Offer product/service	X	Generate RM Skill Matrix	X
Read script provided	X	Count calls made	√
Answer customer's questions	X	Count sales made	√
Log call outcome	√	Monitor performance	√
Answer Calls	√	Analyse performance	√
Add call history	X	Analyse voice recording	√
Call back	X	Determine best/worst RM	√
Insert call details	√	Analyse call outcomes	√
Load products /services	X	Generate performance reports	√
Retrieve end-customer details	√	Improve customer satisfaction	√
Retrieve RM sales script	X	Calculate RM load	√
Load End-customer Details	X	Determine best/worst product	√
Retrieve Matched Script for end-customer	X	Create sales/customer reports	√
Generate RM profile	√	Send sales confirmation	√
Add call history	X	Update monitoring agent	X

#### 4.3.5 Step 4: Identify Agents and Ensuring Tasks Do Not Overload Any Single Agent

This step identifies new agent roles, if any. This step ensures that the allocation of RT tasks does not overload any single agent. This may require splitting some roles into two or more roles. During this step, the RM actor was subdivided into 3 agents (RM, RM Pre-call and RM after call) as the RM was identified to have more than one role e.g. RM doing administrative work such as creating reports, while also answering calls and rerouting calls; hence, this thesis recommended distributing tasks based on the number of roles an agent can do. This is meant to not overload agents with too many RT tasks, which might lead to their failure i.e. an agent is more likely to fail when it has too many time requirements to fulfil. This identifies that the task-to-agent ratio should be reduced in order to cater for the added subtasks workload. Table 4.10 and Figure 4-4 represent the new subdivided 3 agents (RM, RM Pre-call and RM after call) and their tasks with each tasks' time requirements.

Table 4.10: Identifying tasks with time requirements for the RM agent

Agent	Tasks	RT	Agent	Tasks	RT
RM	Confirm customer’s details	X	RM pre-call	Update monitoring agent	X
RM	Offer product/service	X	RM pre-call	Generate RM skill matrix	X
RM	Read script provided	X	RM after call	Count calls made	√
RM	Answer customers’ questions	X	RM after call	Count sales made	√
RM	Log call outcome	√	RM after call	Monitor performance	√
RM	Answer Calls	√	RM after call	Analyse performance	√
RM	Add call history	X	RM after call	Analyse voice recording	√
RM	Call back	X	RM after call	Determine best/worst RM	√
RM	Insert call details	√	RM after call	Analyse call outcomes	√
RM Pre-Call	Load products /services	X	RM after call	Generate performance reports	√
RM Pre-Call	Retrieve end-customer details	√	RM after call	Improve customer satisfaction	√
RM Pre-Call	Retrieve RM sales script	X	RM after call	Calculate RM load	√
RM Pre-Call	Load end-customer details	X	RM after call	Determine best/worst product	√
RM Pre-Call	Retrieve matched script for end-customer	X	RM after Call	Create sales/customer reports	√
RM Pre-Call	Generate RM profile	√	RM after Call	Send sales confirmation	√
RM Pre-Call	Add call history	X	RM after Call	Update monitoring agent	X

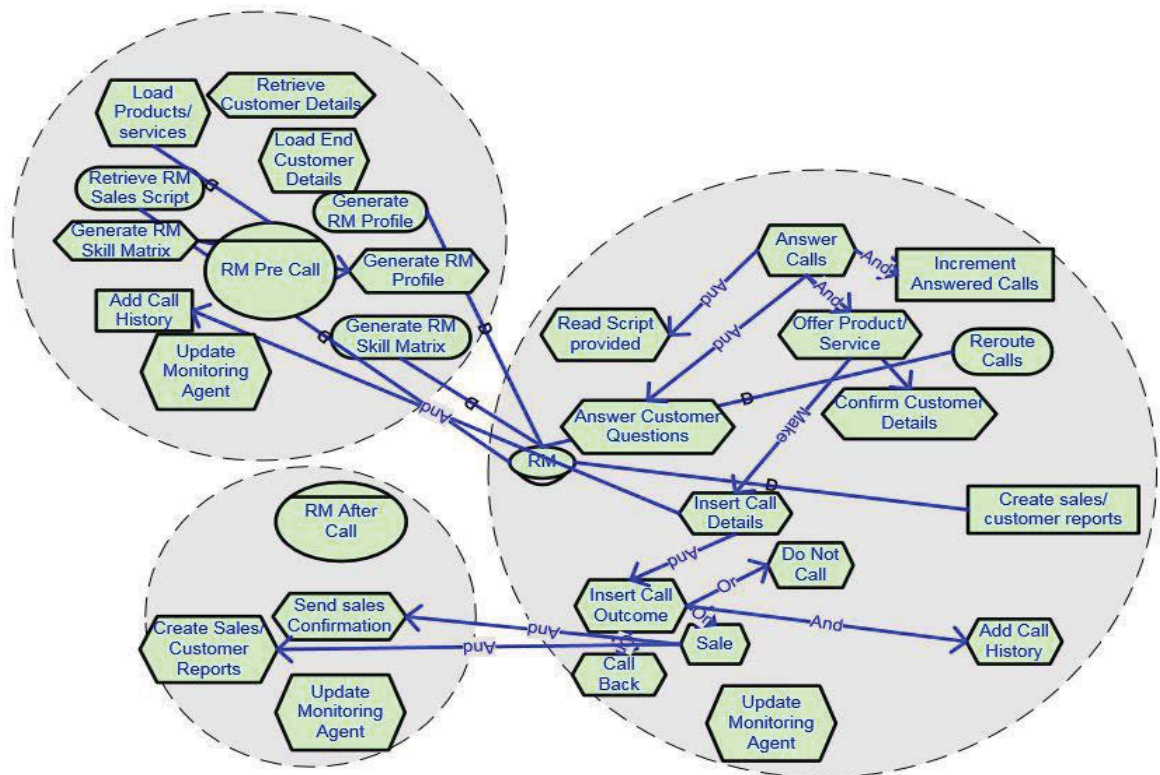


Figure 4-4 SR for the RM, RM after call and RM pre call agents

#### 4.3.6 Step 5: Applying the Modelling units

The last step is applying the 23 proposed modelling units to tasks with identified time requirements. Hence, this step extends step 3 results and checks if and how each of the 23 modelling units is applicable to every task with time requirements.

In this step, some new tasks are identified while applying the Alternate Task “AT” modelling unit. In this case, the identified new task will go through steps 3-5 once again with the 23 modelling units applied to it if applicable.

The three role task and their RT constraints are summarised in Tables 4.11, 12 and 13. Some modelling units require numeric parameters. Each table represents each agent and how the modelling units apply to its tasks, including the parameters of the modelling units where applicable. Then each agent is presented, its tasks and modelling units in a separate i\* SR diagram (Figures 4-5, 6 and 7) illustrating assigning the modelling units to each task.

The Microsoft Visio objects used to draw a diagram (Stencil) created by Horkoff (2007) for i\* MAS modelling have been extended to include the proposed modelling units. The developed 23 modelling units Visio stencil icons are summarised in Table 4.14, where each icon represents the modelling unit value on it. However, not all modelling unit values can be identified, so some values are zero “0”; for example, “Create sales/customer reports” slack time , “Load/Retrieve Voice Recording” PO (Periodic occurrence), “Receive product/service” PO (Periodic occurrence), “Receive Sale confirmation” PO (Periodic occurrence) ,“Generate product/service matrix” REM (Remaining Time) ,“Answer Calls” MMR (Maximum Miss Ration), “Dial number”, MMR (Maximum Miss Ration), “Distribute the load among different RM’s”, Com (Composite). Modelling units with no values are still represented on the i\* diagram to keep the diagram consistent and facilitate individual modelling unit recognition by their location, in a similar way to reading a clock with no numbers; that is, the position of the clock hands (arrows) allow reading the clock even if it doesn’t have numbers written on it. The three agents are each represented in a table below with their modelling units’ values. Where the agents’ tasks are presented in the top row while each modelling unit is in the first column. the values are either text e.g. Soft/Hard, Log Error, Drop Call. or numeric values representing time in seconds, except for the sample time and deadline “Dline” which represent a percentage of the expected. Task status is abbreviated as “O” for on-time, “F” for fail, “I” for idle, “S” for started and “L” for late.

Table 4.11: RM time requirements template























	Confirm EC details	Offer product/ service	Read script provided	Answer EC questions	Log call outcome	Answer Calls	Add call history	Call back	Insert call details
R/N 	X	X	X	X	√	√	X	X	√
Soft/ hard <input type="checkbox"/>					Soft	Hard			Soft
P 					10	1			10
ED 					20	5			20
Dline 					101	101			101
AT 					LogError	DropCall			LogError
PO 					CallEnd	CallStart			CallEnd
RTO 					5	1			5
MT 					10	3			10
MOJ 					5	3			5
RTD 					LocalAgent	PABX, Phone			LocalAgent
C 					6	3			6
R 					3	2			3
Sample Time 					90	60			90
TS 					F	O			1
CP 					9	4			8
VD 					2	2			9
Slack Time 					1	1			1
IVF 					5	1			5
EAV 					3	2			3
Rem 					1	1			1
MMR 					1	0			1
Com 					5	1			5

Table 4.12: Performance monitor time requirements template














































	Count calls made	Count sales made	Monitor performance	Analyse performance	Analyse voice recording	Determine best/worst RM
R/N 	√	√	√	√	√	√
Soft/ hard <input type="checkbox"/>	Soft	Soft	Soft	Soft	Soft	Soft
P 	8	8	8	3	6	1
ED  Estimated Duration	2	2	2	2	2	2
Dline 	101	101	101	101	101	101
AT 	LogError	LogError	LogError	LogError	LogError	LogError
PO  Loop Limit PER	Hourly	Hourly	10Min	10Min	10Min	Hourly
RTO 	1	1	2	1	1	4
MT 	1	1	1	1	1	1
MOJ 	1	2	4	3	1	3
RTD 	DB	DB	DB	DB	VoiceRecorder	DB
C 	8	8	8	2	3	3
R 	3	3	3	5	5	3
Sample Time 	90	90	80	40	60	60
TS 	O	L	L	L	F	F
CP 	2	2	2	2	2	2
VD 	5	5	1	2	2	3
Slack Time 	1	1	1	1	1	1
IVF 	1	4	2	3	2	4
EAV 	1	5	4	1	2	7
Rem 	4	2	1	3	4	1
MMR 	2	4	3	4	2	4
Com 	1	1	2	1	1	4



Table 4.14: The proposed 23 modelling units' icons

ID	Modelling Unit	Proposed Icon	What the icon represents
1	Real or not		A chess table symbol for other modelling units to be presented on top of it.
2	Soft or Hard		A rectangle with hard, soft, firm or weekly hard edges.
3	Priority		The priority pyramid with the priority value written within it.
4	Estimated duration		A clock showing the estimated duration of a task, within which it is to be completed.
5	Deadline		A red callout with the deadline value noted within it.
6	Alternate task		An arrow pointing/linking to the alternate task.
7	periodic occurrences		An index card with the value written on it.
8	Real-time order		A task order tag.
9	Minimum time		A sand clock.
10	Max output jitter		A graph showing different execution times.
11	Real-time dependency		A dotted arrow pointing to the RTD.
12	Criticality		A star displaying how critical the task is.
13	Retry attempts		A pentastar displaying how many times the task can be retried/restarted.
14	Warning level /Sampling time		A yellow callout with the sample time within in.
15	task status		A clock displaying the task current status.
16	Check points		The save button as the check point saves the current task values.
17	Validity duration		A parking meter.
18	Slack time		An addition (add sign) to the execution time.
19	Instant value function		A camera snapshot of the current execution times.
20	Execution accrued value		The accrual of money, as time is money.
21	Remaining time		Two clocks illustrating the time difference (remaining) between the current time and estimated duration.
22	Maximum-miss-ratio		A maximised speed odometer.
23	Composite		The composite work of four agents.



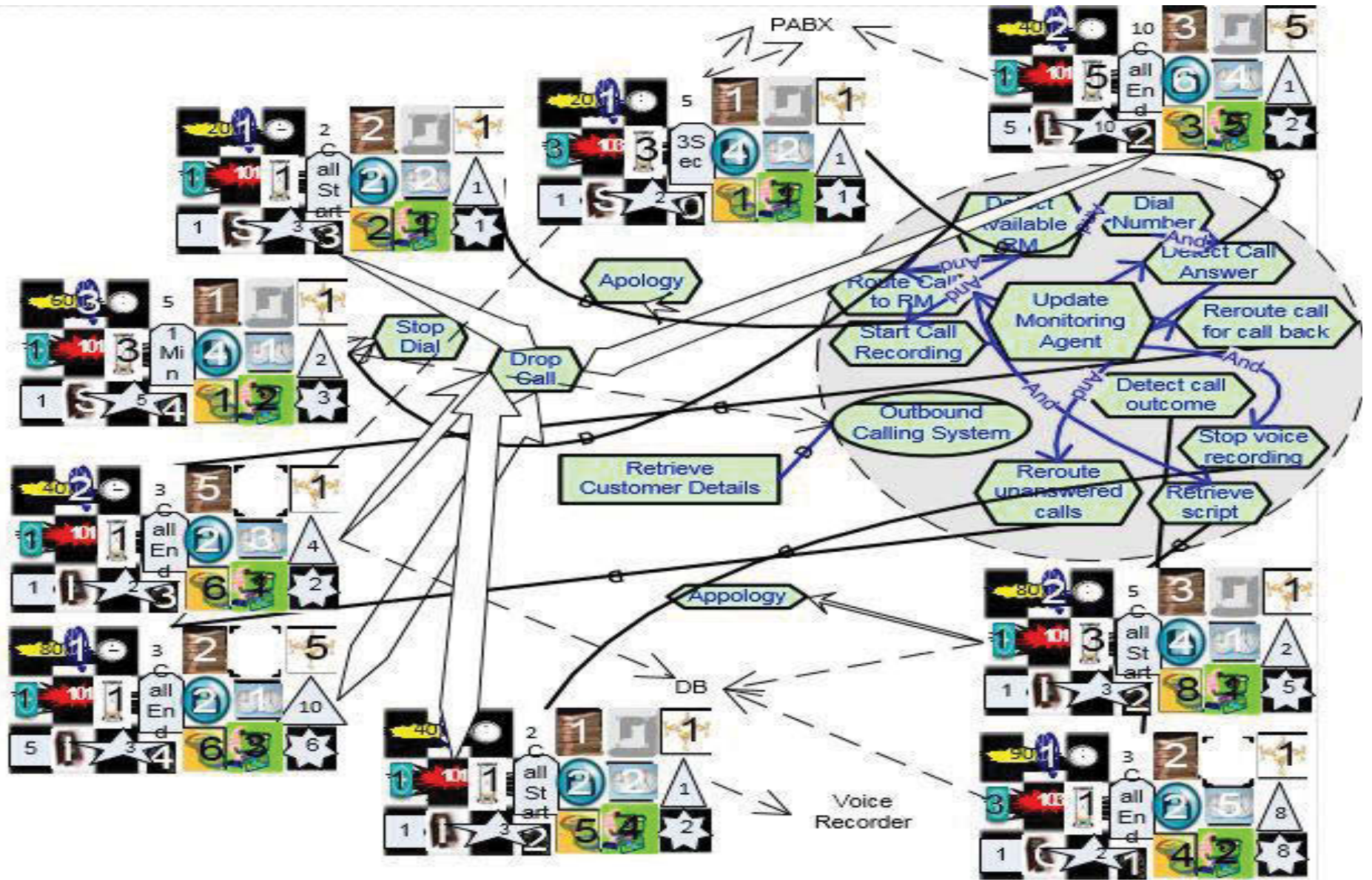


Figure 4-5 Outbound calling system agent with the modelling units

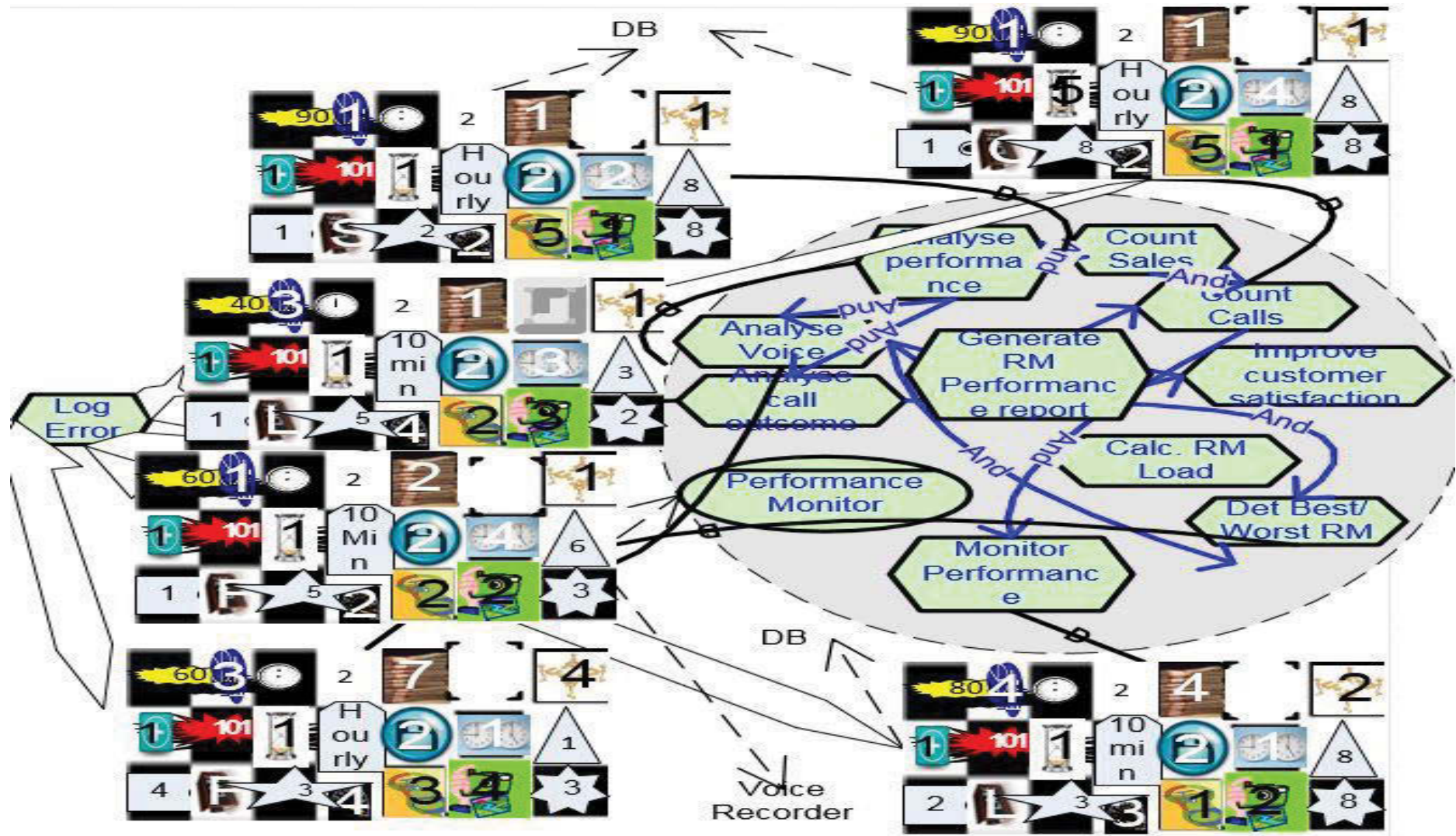


Figure 4-6 Performance monitor agent with the modelling units

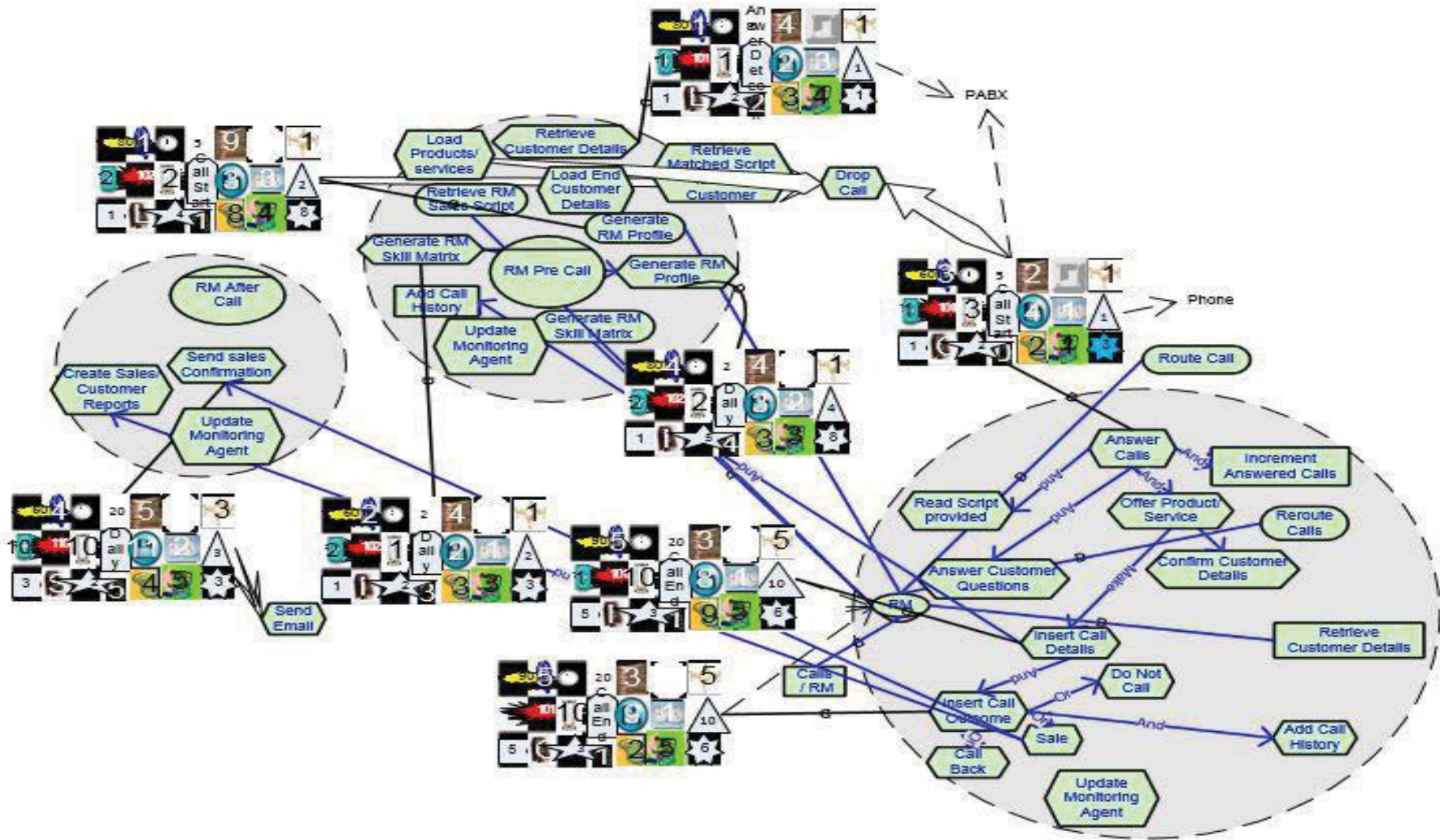


Figure 4-7 RM, RM pre-call and RM after call agents with their modelling units

#### 4.4 Conclusion

This chapter identifies 23 modelling units using the literature. Initially, an original set of 88 modelling units (23 defined + 65 with similar wording) is identified and then grouped to 23, after removing duplicate modelling units with similar definitions. The grouping process is done in two stages:

- A) Gathering similar modelling units together, which is done based on their names and definitions. The final name grouping with the definitions and references are further detailed in Table 4.2, highlighting each group by having a thick boarder square lines around each group.
- B) The 2<sup>nd</sup> stage, is choosing the most commonly-used wording and definition to represent each group of similar modelling units' names. This is done by each modelling unit name score; that is, the name that is most referenced is used as it is the most commonly-used name.

This research documented and grouped all modelling units' names and similar definitions. Each modelling unit has a count of the number of times it has been referenced, which represents the modelling unit score. That count is increased whenever a modelling unit is stated and referenced noting the new definition and the similarity between it to the existing modelling unit, as illustrated in Table 4.2.

These modelling units are then validated using a call centre case study that investigated implementing a multi-agent system (MAS) in a call management system to profile and match end-customers (EC) to relationship managers (RM), based on demographic criteria and characteristics such as age, sex, culture, language proficiency, experience, product knowledge; which makes the RM more convincing to the EC and increases the chance of achieving a sale. The MAS also assists RMs in serving ECs by adjusting call flow. For outbound calls, the MAS adjusts the RM calling list based on his performance, skills and profile. The system creates an initial profile for each RM, which is dynamically adjusted, based on the RM performance throughout the day in real-time. This case study has illustrated that identifying time requirements in the early analysis phase, leads to better load distribution among agents and ensures that agents can meet their requirements. This answers the first question of when is best to use the modelling units; it is preferable to use it in the analysis phase. This case study also proved that the modelling units are usefull as they helped redistribute the agents workload, which in turn

ensures the system success to fulfil its real-time requirements. Identifying time requirements is vital for building a MAS as two of its main characteristics are reliability and redundancy. Meeting time requirements is vital for many systems but over-engineering system redundancy and RT monitoring can have the opposite effect and lead to a very slow, unreliable system.

This chapter has initially presented the modelling units' advantage over current time requirements implementation as:

1. Encourages analysts to consider 23 modelling units versus only 1, such as deadline.
2. Encourages analysts to quantify (add values) to each of the modelling units, if possible.
3. Identifies and resolves real-time issues as they arise. As some modelling units help identify a late task, for example, sampling time, while other modelling units help rectify a late running task such as an alternative task and/or checkpoints.
4. Load balance tasks-to-agents' ratio.
5. Shifting the RT requirements to be a real-time task attribute (something attached only to RT tasks) versus repeated tasks within every agent, as illustrated in Figure 4-6; that is, when first attempts were made to use the RT requirements without modelling them (modelling units).

The next chapter, Chapter 5, details the meeting scheduler case study including a detailed simulation and an implementation process development and validation. The process was developed to address the issue stated above, and has led to implementing the modelling units in a certain order (sequence) as well as using them as task attributes. This was mainly to reduce the workload of running all the code even after resolving the delay from the first executed modelling unit(s). The process also validated the efficiency of each modelling unit to resolve task delays, as well as validating the possibility of dividing modelling units into sub-groups. This was to determine whether all modelling units were needed, or a subset of them would be sufficient. Hence, the process validated the quantity (number of modelling units to be used) and the quality (efficacy in identifying and resolving task delays) of each modelling unit, as further discussed in the next chapter.

## CHAPTER 5

### SYNTHESIS OF THE RT MODELLING PROCESS

Chapter 4 presented the second research phase of this thesis where a set of modelling units was identified and validated using a call centre analysis domain. This chapter presents the third phase of this thesis. It develops the process systematising the use of the modelling units. The process guides a software developer in generating an enhanced set of requirements models that capture real-time constraints.

To develop the process, multiple processes are created and simulated. A best-of-breed process is chosen and refined. In choosing and validating the process, the set of modelling units is further validated and refined. The threat against domain dependence is mitigated, as a different domain to the call management is used. A calendar scheduling simulation is used. The calendar here is made time aware (unlike say an *Outlook* calendar). Furthermore, this calendar scheduling domain is chosen as it can be easily mapped to other domains by using the *task* concept instead of a calendar events/meetings e.g. For instance, any project context e.g. construction, software development, supply chain, planning. all have a scheduling time component, which can be presented using their start/end times, location and dependencies. Tasks in such domains often have time constraints and need their scheduled time updated throughout their execution.

The calendar simulation used in this chapter monitors delays of arrival to a meeting for instance. It also executes rescheduling actions in case of cancellations or other unforeseen environmental changes. Users receive email alerts as required. The function of the calendar is simulated in various scenarios representing different events with various real-time constraints. Users are notified of delays, and reschedules are generated based on actual expected arrival times, when possible. In simulation runs, various processes with the modelling units are used to reschedule meetings. The chapter is organised as follows. Section 1 introduces the calendar domain. Section 2 discusses different meeting scenarios, starting by a two-person meeting then introducing meetings with multiple travel options, to get further insight into the domain. Section 3 presents the modelling units' integration; that is, how the modelling units will be used to represent the meeting scheduler attributes. Section 4 represents the modelling units' dependencies, relationships leading to various diagram representing the process, which is further discussed in Section 5. The proposed process is validated in Sections 6, as individual modelling units,

sequential modelling units and randomly-selected modelling unit sequences. Finally, this chapter is concluded in Section 7.

## 5.1 Introduction

This section provides a background and insights into the calendar domain and the simulation setup. The simulation describes a computer-based meeting scheduler that determines a meeting date and location to suit the largest number possible of potential attendees. The scheduler requests from potential attendees their availability for a date range based on their personal agendas and mediates an agreement for an acceptable meeting date/time. It is based on (Jurisica et al. 2004). The simulation begins starting with only 2 people attending a single meeting, and progresses to complex settings simulating 179 people attending 101 meetings with 754 events. Events simulated range from delays to meeting cancellations under certain conditions. The simulation is used to validate A) that the proposed modelling units in fact enhance meeting rescheduling; B) the success rate of each modelling unit; C) a preferred sequence (proposed process) for the modelling units; D) whether a subset of the modelling units is preferred to the full 23 modelling units set.

In the simulation, events are created to trigger rescheduling of meetings which might result in a cascade of meeting conflicts. A meeting conflict is when two meetings overlap for an attendee; that is, the assumption is that an attendee cannot be in two meetings, in two different locations, at the same time. The modelling units from Chapter 4 and a newly proposed process are used to resolve these conflicts. Their success rate is measured. This is reflected in how many meeting conflicts are successfully resolved (or rescheduled), allowing all required attendees to attend them. The proposed process is then further enhanced and a best-of-breed process is chosen and refined. For simplicity, only public transportation mode is considered in this simulation. The option of driving is not considered as driving routes, times, traffic conditions and parking time are less predictable and the added complexity is out of scope. Using public transport, travelling times can include riding one or multiple means of transport such as train, ferry or bus. Transport schedules are known to all attendees, enabling attendees to compare their location and times to transport schedules and estimate their travel/arrival times. All attendees are assumed to be time-aware. Time awareness is critical to identify if they are on track, late or early for their meeting. On the other hand, they must be able to communicate any changes to their meeting place or time, due to rescheduling or delays.

A person's delay proves to be an important factor, where delays are communicated to other meeting attendees; if the person is running too late, the meeting is rescheduled to another day/time.

Each meeting attendee chooses a specific transport method based to their internal state. Assumptions leading to such a choice include a preference for a mode of transport, arrival and departure times. These internal states are related to running costs, ease of use and work/personal arrangements that are not directly related to this research. There are a number of events that could affect the person's transportation choice and time of travel, such as:

1. External effects: For example, if the attendee has a leg injury then they would have to choose an option that involves less walking so as to avoid changing from buses to trains. The length of the walk from a bus stop to a train station may be more than changing trains on the same or different platforms within the same station.
2. Rescheduling effects: The trains are running on a different schedule due to delays or track work. Such schedule changes would affect the attendee's choice of time and means of travel.
3. Weather effects: This can affect both the attendee and the schedule; for example, on a rainy day an attendee would prefer to avoid walking in uncovered areas as much as possible, hence avoiding walking to train stations and/or choose the closest station/ bus stop to their destination. Rainy weather usually causes delays directly affecting bus and ferry schedules.
4. Missing a train, bus or ferry is an event that would trigger one the following three rescheduling actions: Firstly, consider another instance of the same transportation mode, if it enables the attendee to meet the deadline; for example, express train, different route. Secondly, consider an alternate transportation mode that would enable the attendee to meet the deadline. Or, simply reschedule the meeting if the attendee cannot meet the deadline.

A meeting is considered successful only when it takes place. Thus, the following criteria are used to identify successful meetings:

- 1- No attendee has cancelled, rescheduled or notified of their absence.



- 2- The meeting does not conflict with any other attendees 'schedule; that is, for each meeting attendee, no other meeting start or end times falls between this meeting's start and end times.
- 3- The time between 2 consecutive meetings allows an attendee travel time between the two meetings. That is, we cannot have meetings in 2 different locations 100 km apart with only 5 minutes between the end time of the first meeting and the start time of the 2<sup>nd</sup> meeting.

Meeting attendees in this simulation are represented by agents in a MAS, with which they are required to negotiate their availability and status. For simplicity, they are referred to as attendees, while later in this and the following chapter attendees are represented by MAS software agents. Where an agent represents an attendee, illustrating the efforts to attend a meeting e.g. negotiation meeting start time, location and duration. The next section presents meeting scenarios that will be simulated.

## 5.2 Meeting Scenarios

The calendar simulation is implemented with a database backend which stores all attendees' information, meetings, locations and transport schedules. The simulation runs will be performed using different scenarios with and without time constraints. The impact of using the modelling on the meetings success rate will be measured. The simulation will focus on how an attendee reacts when they discover that they would be late for a meeting. Since all transport methods are represented as scheduled meetings, alternatives to arrive on time in the simulation will be considered as alternative meetings that satisfy the same goal. This allows simulating alternative transport methods and meetings attendance as calendar events to be implemented as MAS, where alternative meeting times will be negotiated among different attendees (agents). Benefits from using the modelling units' constraints will be assessed by comparing the success of the scheduler with and without their use. When an attendee cannot make a meeting one of the following time aware actions are simulated:

1. Meeting delay: If an attendee's previous meeting takes longer than expected, subsequent meetings are rescheduled depending on their start times. The process of rescheduling for MAS will include negotiation and communication with other attendees, while for a single attendee it would only be time changing without any communication.

2. Meeting cancellation: If an attendee's meeting is cancelled, this is considered gained/free time allowing a MAS flexibility to negotiate other meetings that had to be delayed/reschedule or cancelled if one does exist. For non-time aware applications, this would represent only marking the meeting time as free.
3. Meeting rescheduling: If one attendee needs to reschedule a meeting, this will require negotiation between the different attendees to agree on the best time to meet. For a single attendee, this will require setting a new meeting time and marking the old meeting time slot as free.

The meetings scenarios are discussed to give some preliminary insights into the simulation variables. A simple scenario of two people travelling to meet at a specific location is first presented. This is then enriched by adding more people and meeting locations with alternative travelling methods. The resultant scenarios are based on actual times from a Sydney transit website (CityRail 2014) and Google maps (Google 2014).

*A Two Person Single Meeting Initial Scenario:* The meeting for the two people, "A" and "B", is illustrated in Figure 5.1 "A" travels from train station "X" to station "Y" where the meeting is scheduled for 11:00 am. "B" travels from station "Z" to station "Y". For "A" to arrive by 11:00 am s/he needs to take the train leaving station "X" at 10:30 am. While attendee "B" can take one of three trains. "A" train leaves every 5 minutes and the trip takes 25 minutes. The first train leaves at 10:15 am and arrives at 10:50 am. Assume that attendee "A" is running late and arrives at station "X" at 10:35 am missing the 10:30 am train leaving. The next train leaves at 10:45 am, arriving to station "Y" at 11:15 am. Thence, attendee "A" needs to notify attendee "B" of the delay and negotiate rescheduling their meeting to 11:15 am instead. Depending on "A" and "B"'s schedules, the negotiation outcome can lead to a meeting at 11:15 am or to rescheduling to another day/time. This

can also lead to delaying other meetings depending on how critical between “A” and “B” meeting is in comparison to the other meetings.

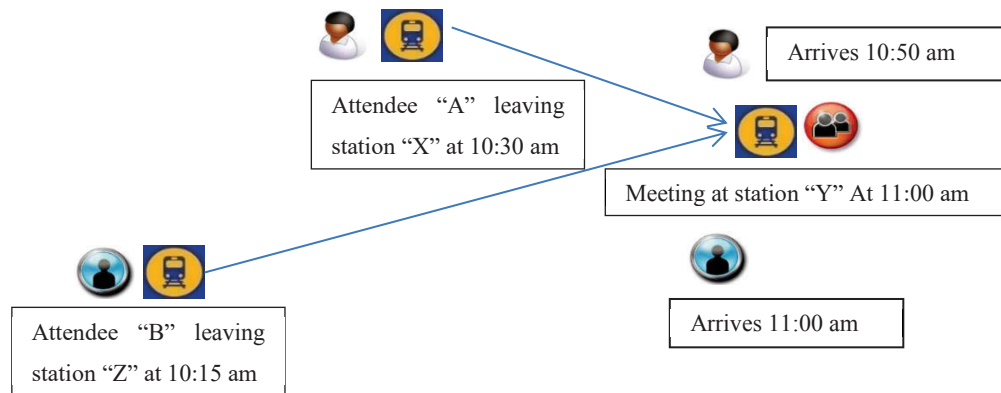


Figure 5-1 Meeting attendees travel and arrival times to a scheduled meeting

More complex scenario would include multiple attendees with a dense calendar and multiple alternative travelling methods. For example, travelling from point “X” to “Y” can include buses, trains, ferries, taxis, express trains, private cars or even plans to different countries and time zones. This leads to more complex models as illustrated in the next scenario:

*Meeting with alternative travel options and more attendees:* The next step in this simulation is to develop a more advanced scenario using Google Maps to travel between 4 Sydney locations (X=Miranda, Y=Martin Place, Z=Bankstown, K=Mossman) as shown in Figure 5.2. The scenario considers three modes of public transportation: Attendee “A” takes only trains; Attendee “B” takes trains and buses and Attendee “I” takes buses and ferries. All 3 attendees work together and are assumed to have a meeting together at 9:00 am; hence, all the planning is for them all to arrive before 9:00 am.

This scenario assumes that attendee “A” takes the train daily to work. S/he walks from the house for 10 minutes to the train station “X” where they board the train. They then change trains at station “T” to reach the closest station to their work (station “Y”). This train trip takes between 41 and 48 minutes depending on the time of day and how long they have to wait for a connecting train. An attendee then has to walk for 20 minutes to reach the office. Attendee “B” takes a bus and a train to work. S/he walks from their house for 3 minutes to the closest bus stop, where s/he takes the bus to the train station. S/he then has a choice of taking an “express train” or “limited stop” train. “Express trains” travel from station “Z” to “Y” in 43 minutes, while limited-stop trains take 50 minutes. Attendee “B” has to walk from station “Y” to the office, which takes about 20 minutes.

Attendee “I” takes a ferry, train and bus; the trip for attendee “I” starts from station “K” “Mossman” then a 3-minute walk to take a ferry then a bus to get to work.

Figure 5.2 illustrates the travel options for the three attendees and their slack time. The slack time represents a time buffer between two consecutive transport methods e.g. when changing trains, or from a bus to a train. Not having enough slack time between consecutive transports increases the chance that an attendee would miss their next ride. However, having too much slack time will increase the overall goal time, which is not always preferred. For example, the 2<sup>nd</sup> option for attendee “A” is to take one of 3 consecutive trains without much slack time, which represents a high risk of missing a train. The 1<sup>st</sup> option, to take 2 trains 9 minutes apart, reduces the risk of missing the 2<sup>nd</sup> train (e.g. if the first train is delayed) without affecting the overall goal time. Both options start at 8:05 in order to board the 8:07 am train. The first option, with less risk, finishes at 8:51, five minutes before the 2<sup>nd</sup> option completes at 8:56.

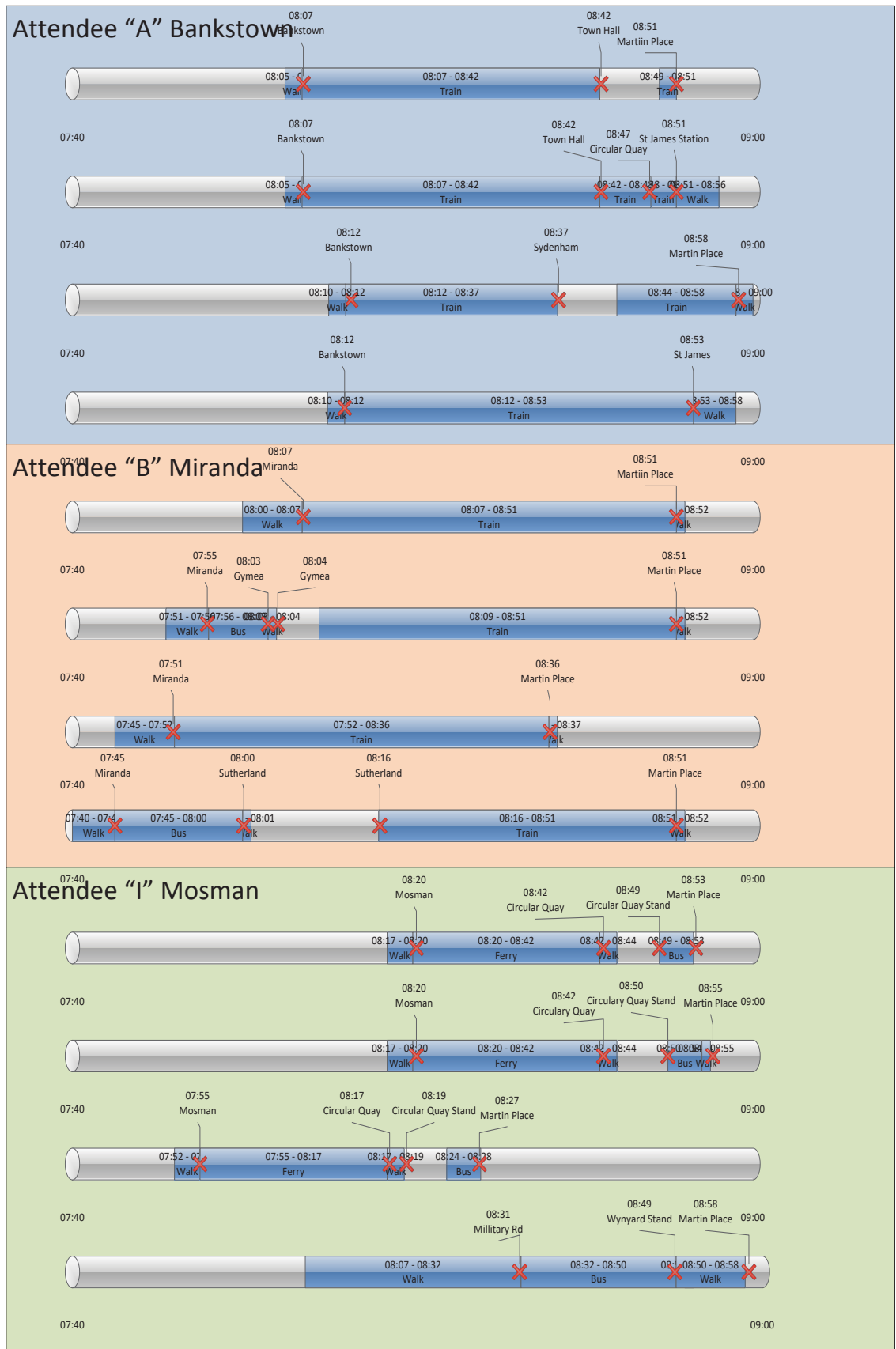


Figure 5-2 Attendee "A", "B", "I" Travel options

### 5.3 Modelling units' Integration

The RT analysis process discussed in chapter 4 is applied to the calendar domain to identify the agent roles and their tasks. As earlier described, the RT analysis process consists of 5 steps which are summarized with the analysis results below:

Step 1: Identify Agent Roles: 3 agent roles are identified: the meeting attendee, the transport method and the meeting itself.

Step 2: Task Analysis: The tasks of each of the roles from Step 1 are identified. For the meeting attendee, three tasks are identified: attending a meeting, taking notes in a meeting or rescheduling a meeting. For the transport agent, two tasks are identified: either to arrive or to leave according to given scheduled times. Finally, the meeting agent itself has these four tasks: sending invitations to attendees, sending start reminders, or sending finish reminders. The reminders contain the meeting details.

Table 5.1: Meeting attendee, transport method and meeting agents identified tasks

Meeting Attendee	Transport Method	The Meeting
Attend a meeting	Arrive on scheduled time	Send invitations to attendees
Take notes	Leave on scheduled time	Send start reminders
Reschedule a meeting		Send finish reminders

Step 3: Identify and Refine RT Tasks: Of the tasks identified in Step 2, for each agent the real-time tasks are identified. For the attendee agent, attending a meeting has RT constraints. For the transport agent, both arriving or leaving have RT constraints. For the meeting agent, all three tasks (sending invitations, start reminders or finish reminders) have also RT constraints. In the event that an attendee cannot attend a meeting, three alternate tasks are identified: cancel, reschedule, postpone meeting. These are all RT tasks as summarized and highlighted in Table 5.2. Not being able to attend a meeting can be due to any environmental or extremal factor related to the attendee (e.g. being sick), transport method (e.g. broken down or running late/ behind schedule) or the meeting itself (e.g. meeting conflict due to the arise of a higher priority meeting or location change due to weather, unviability)

Table 5.2: The agents and tasks

Meeting		Attendee		Transport	
Start reminder	RT	Attend	RT	Arrive	RT
Finish reminder	RT	Reschedule		Leave	RT
Send invitation	RT	Take notes			
		<b>Cancel meeting</b>	<b>RT</b>		
		<b>Reschedule meeting</b>	<b>RT</b>		
		<b>Postpone meeting</b>	<b>RT</b>		

Step 4: Revisit RT Task Allocation: In this domain, this step did not change the allocation of the tasks from Step 3.

Step 5: RT Modelling Units Allocation: This step is the focus of this chapter. A selection process is developed. Towards this, different subsets of the modelling units were tested, and within these tests the modelling units were sequenced differently. A best of breed of a subset and a concomitant sequence is chosen. The aim is to differentiate the importance of the modelling units wherever possible and whether applying the units in different order is important to the overall effectiveness of the subset. Hence, various subsets and sequences are evaluated in terms of their effectiveness. Therefore, all the modelling units are investigated in this domain, as follows:

1) Estimated Duration “ED”: This the estimated task duration e.g. travel time for each transport method, which is calculated as  $ED = (\text{Arrival Time}) - (\text{Departure Time})$ . ED also represents other task durations like the meeting scheduled duration.

2) Slag Time “ST”: This represents the time buffer between one task end time and the following task start time, such as the time difference between a train arrival and the meeting start time, which gives the attendee some delay time without affecting the meeting start time. Figure 5-2 illustrates the slack time for attendee “B” is 10 minutes, as the attendee’s first train arrives at 10:50 while the meeting is at 11:00.

3) Deadline “Dline”: This is the meeting start time. However, each task would have its own deadline; note that transport methods (trains, buses and ferries) departure times are considered deadlines.

4) Criticality “C”: This represents how important this meeting is to the attendee, so as to consider the possibility of rescheduling other meetings for this meeting to occur, or reschedule this meeting in favour of other meetings. For example, in Figure 5-2, the 8:49 bus option for attendee “I” can be considered a non-critical option as it does have an alternative option (take the 8:50 bus). Similarly, taking the 8:07 train for attendee “A” can be considered a non-critical option as it does have an alternate option (take the 8:12 train). For attendee “I”, the 8:20 ferry is considered a critical option as it’s the last option - all other alternate options happen before it; hence, if the attendee doesn’t catch the 8:20 ferry they are likely not to meet their 9:00 deadline.

5) Hard/Soft Deadline “S/H”: This represents whether the meeting time is considered a hard or soft deadline (i.e. if the attendee can be late for the meeting or not). For example, in Figure 5-2 the 8:49 bus option for attendee “I” can be considered a soft deadline in comparison to the 8:50 bus, as if they miss the 8:49 bus they can still catch the 8:50 bus, arriving before 9:00 (deadline). The same can be stated for the last starting times for each attendee, where they could miss the earliest starting times and still arrive before 9:00 (deadline). However, if the attendee misses the last start times (8:10, 8:00 and 8:17 for attendees “A”, “B” and “I” respectively) then they would be unable to meet their deadlines.

6) Minimum Time “MT”: This is the minimum time that a task needs to be executed. Public transport (trains, ferries and buses) MT is represented by their scheduled time; as they have to run at specific speed and meet certain schedules; that is, a train cannot arrive and leave earlier than its schedule. For cars and taxis, the MT would be the maximum speed/distance as this thesis always assumes law obedience, that is, travelling within speed limits. Simulating walking between transports methods also has a MT depending on how fast the person can walk or run if necessary; hence, the current walking time is based on an average adult walking speed.

7) Checkpoints “CP”: This represents a point where task results can be saved. Checkpoints would be on each individual arrival time, such as, attendee leaving train in “Town Hall”, “Circular Quay”, “Sydenham” and “Martin Place”. In each checkpoint the attendee could confirm that they are on schedule and their status can be saved till the next station.

8) Validity Duration “VD”: This is the maximum time the data can be held for, before expiring or being considered invalid. The saved information is valid until the next station, then must be overwritten by the new status and information.

9) Maximum-Miss-Ratio “MMR”: This is the maximum number of times an attendee can miss a soft goal; for example, in Figure 5-2, the MMR for attendee “B” is 3, so they can miss 3 trains and still arrive on time, assuming they board the 4<sup>th</sup> train, which arrives before 9:00.

10) Instant Value Function “IVF”: This is the total accrued value of a job, which is equal to the total travel time including time between different transport methods.



- 11) Execution Accrued Value “EAV”: The amount of time gained if the attendee arrives early to their destination; that is, the transport method (car, bus, trains) arriving ahead of schedule.
- 12) Task Status “TS”: This represents the current state of the task, which could be on time, delayed, ahead of schedule.
- 13) Alternate Task “AT”: This is the different option that the attendee can perform if they miss, or are close to missing, a deadline. In Figure 5-2, the 8:50 bus for attendee “I” is an alternate task for the 8:49 bus.
- 14) Periodic Occurrence “PO”: Is the rate of arrival of the public transportation; that is, the schedule of the public transport bus, trains and ferry. For 3 example, attendee “I” has a bus every 15 minutes, while attendee “A” has a train every 30 minutes.
- 15) Sample Time/Warning “W”: Used on each station to identify if an attendee is running on time or not.
- 16) The Composite “Comp” field: Used for the arrival of several events at the same time e.g. when more than one person is delayed so the maximum delay is considered.
- 17) Priority “P”: This is similar to the meeting criticality; however, Priority ranks meetings based on their importance; while criticality ranks a meeting as critical or not; that is, it provides a method to rank alternative transport methods.
- 18) Real-Time Order “RTO”: This provides the order in which meetings must occur. That is, the attendee first needs to reach his meeting location for the meeting to occur, hence the meeting becomes linked to the transport method(s) and any delay in a transport would be cascaded to dependent meetings.
- 19) Maximum Output Jitter “MOJ”: This is the difference between the best execution time and the worst execution time; it has been applied for long meetings and some transport methods, such as walking, buses. Walking does not have a strict schedule but relies more on average walking speed, while buses are highly affected by traffic and weather conditions. As for meetings, some attendees tend to run over their meeting schedules, hence when meeting with such attendees the MOJ is taken into consideration when scheduling further meetings.

20) Real-Time Dependency “RTD”: This is the meeting organiser who should be notified of any delay or meeting rescheduling.

21) Retry attempts “R”: This can be used as one of the alternate tasks; the initial alternate task is to rerun the same task if unsuccessful, then start a different alternate task. These tasks may range from rescheduling the meeting to finding alternative transport methods, as discussed above.

22) Real or not “R/N”: This indicates if the task time should be monitored, and notify of any delays or if the task/meeting is not time dependent.

23) Remaining time “REM”: This is the difference between the current time and the meeting start time.

Although the calendar simulation enables the deployment of each of the modelling units. Based on the above discussion, and further discussions when attempting to deploy the modelling units, it becomes clear that the below 5 modelling units are considered duplicates:

1. The Retry “R” modelling unit allows the task to rerun once again. This is possible when the task has enough slack time to rerun again. It suits a RT-MAS and can be identified during the analysis phase. It adds value as it provides a 2<sup>nd</sup> chance for the task to run before failing; however, it can be regarded as an extraction from the Alternate Task “AT” as restarting the same task for a number of times ensuring system redundancy; which is similar to a self-join in a database system. Hence, modelling “R” is recommended as part of an “AT” and not as a separate modelling unit.
2. The Real-Time Order “RTO” is the same as slack time “ST”; however slack time is preferred as it is better documented and referenced in the synthesis process as presented in chapter 4.
3. The remaining time “REM” is a calculated field, where its values are populated at run time. however, it was only needed in the analysis phase to develop the code to calculate it.
4. The Maximum Output Jitter “MOJ” field is also a calculated field; with its values populated at run time; however, it was only needed in the analysis phase to develop the code to calculate it.

5. The Composite “Comp” field is used for the arrival of several events at the same time e.g. when more than one person is delayed so the maximum delay is considered, which is calculated by the event delay itself.

The deployment of modelling units is required to synthesize the process sought and adds further credence for selecting this domain. The chapter will conclude in identifying a process describing the sequence of applying the modelling units. The next section discusses the modelling units’ dependencies and relationships leading to various candidate processes.

#### 5.4 Modelling units’ Dependencies

Identifying relationships between the modelling units is the first step to develop a process to implement them. The process, once elaborated and synthesized, becomes deployable by system analysts and developers. The relationships are first presented per modelling unit, then as a network of dependencies (later shown in Figure 5-3).

- 1) Criticality “C”: The degree of criticality is a function of dependant and alternate tasks; that is, a task is considered critical as the number of dependant (RT Dependency) tasks increases and the number of alternate tasks (AT) decreases. For example, a task with no alternate and 100 dependent tasks is considered critical, while a task that has 100 alternate tasks and no dependencies would be considered non-critical.

$$\uparrow C \approx AT \downarrow, RTD \uparrow$$

$$\downarrow C \approx AT \uparrow, RTD \downarrow$$

- 2) Priority “P”: Is a function of the Task Status “TS” and how critical “C” the task is “Criticality”; that is, a critical task running late should have a higher priority than a non-critical task running ahead of schedule. Generally speaking, all critical tasks should have higher priority than non-critical tasks. The priority is also a function of Periodic Occurrence “PO”, where if task occurs in very low intervals then its priority decreases as if it fails then the next instance of this task will run very soon.

$$\uparrow P \approx TS \text{ "Late" }, C \uparrow, PO \uparrow$$

$$\downarrow P \approx TS \text{ "Early" }, C \downarrow, PO \downarrow$$

- 3) Real-Time Order “RTO”: The priority “P” of the task decreases if its real-time order (RTO) has a positive value; hence, the task has some buffer time to run late.

- 4) Soft/Hard: Critical tasks normally have a hard deadline but not all tasks with a hard deadline are critical. For example, a PhD conference paper publication would have a hard deadline (paper submission date); however, it's not critical for completing a PhD thesis. While the PhD end date, first year progress review, annual progress reports. are considered critical tasks with hard deadlines.
- 5) Maximum-Miss-Ratio "MMR": All soft deadline would have a maximum miss ration; MMR, as a soft deadline cannot be missed forever.
- 6) Deadline "DLine": The deadline is a function of the Real-Time Order "RTO"; if it has a positive value then the task duration can be extended by that value before the dependant task fails, and vice versa for RTO negative values that is  
Deadline = ED  $\pm$  RTO
- 7) Estimated Duration "ED": The Estimated Duration is a function of historical runs of that task. Since first run/instance of the task will not have any history, then it would be based on the software engineer's input, lines of code or exception based on his/her experience.
- 8) Retry "R": Retry attempts are a function of Alternate Task (AT) and periodic occurrence (PO); If the PO is low, then there is no need to retry as the task will automatically re-run. However, if it keeps failing on a schedule or a retry attempt, then this task is considered un-functional and an alternate task should be considered if one exists.

$$\downarrow R \approx PO \downarrow, AT \downarrow$$

- 9) Real-Time Dependency "RTD": RT Dependency reflects the actual task dependencies, as highlighted in criticality above. The criticality degree is a function of RT Dependency.
- 10) Warning/sample time (W) "The warning/sample time (W) is a function of how critical the task is. A critical task should be sampled more often than a non-critical task, so as to early identify any potential delays and fix them; for example, assigning more resources. There is an upper limit of sample times, as sampling by itself consumes resources and could potentially delay tasks rather than help resolve task conflicts and/or delays.

$$\uparrow W \approx C \uparrow$$

- 11) Slack Time "ST": The more slack time a task has, the less priority it would have, as it will have more time to be delayed without affecting or missing its deadline.

12) Validity Duration “VD”: All checkpoints have a validity duration, after which the saved values expire.

13) Check Point “CP”: Tasks have checkpoints where their values could be saved.

The above dependencies can be graphically represented as shown in Figure 5-3, to identify their inter-relationships.

The relationships between the 23 modelling units discussed are important to identify a logical process to implement/execute the modelling units. That is the order in which the modelling units should be checked e.g. since only soft deadlines have a maximum miss ratio (MMR) then if a deadline is identified as a soft deadline then the next logical check would be to confirm its MMR. However, this research has went beyond providing a single logical process and has proposed a number of candidate processes, logical and randomly chosen. As such, the next section presents the different candidate processes. They are evaluated to identify the most effective process that enables the analysts to identify a sufficient set of modelling units for a given RT MAS application.

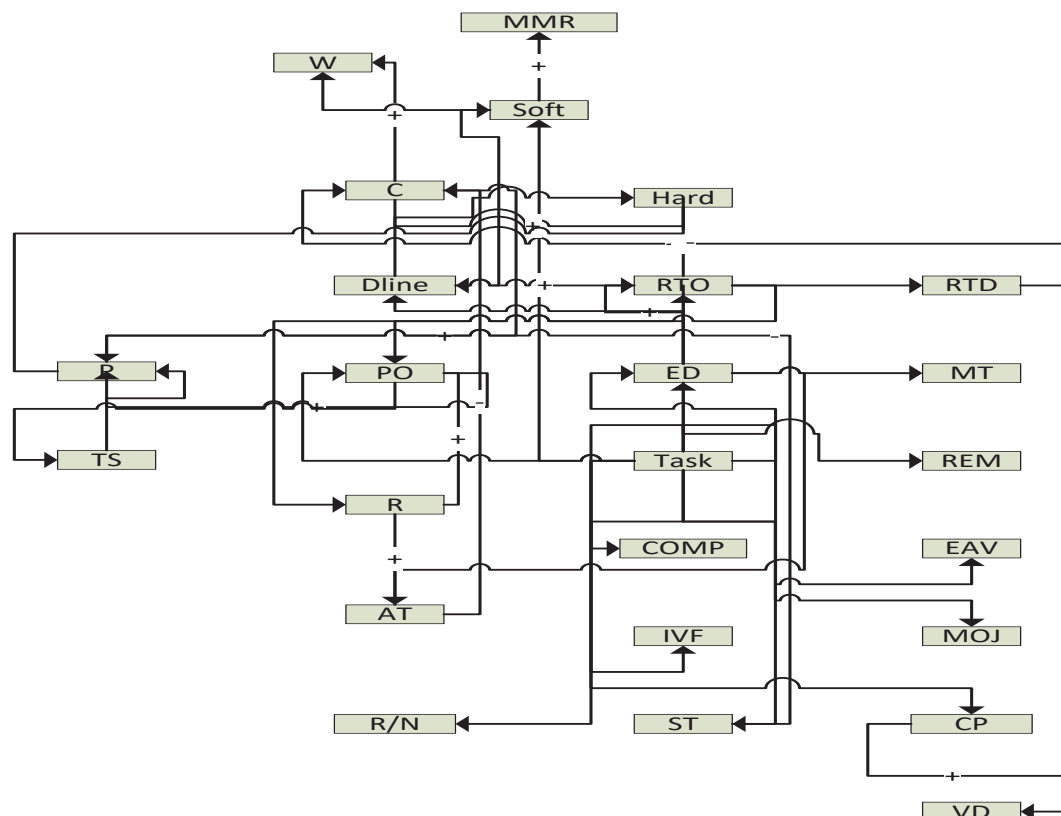


Figure 5-3 Relationships between real-time constraints

## 5.5 Proposed Process

No single modelling unit could resolve all conflicting meetings. Rather, a set of modelling units in the right order is required. An effective modelling process to guide the

identification of modelling units thus needs to describe both, (1) a sufficient set of modelling units, and (2) the order (sequence) of the targeting of the modelling units.

### 5.5.1 Identifying the Modelling units Set for the Process:

Identifying the set (i.e. a subset of the 23 modelling units) is not an easy feat. This would require different combinations of sub-sets, which meant a sub-set of 2 modelling units then 3,4,5...23 modelling units; with each set having different modelling units' order i.e. for a set of 2 modelling units there would be two tests one with modelling unit A first then B and another test with B first then A. That meant  $23^{23}$  ( $23^{23}$ ) different candidate sets to test and simulate, which is equivalent to 20,880,467,999,847,900,000,000,000,000. Unfortunately, resources were not immediately available to develop and run such simulations. The approach here rather uses a random sampling approach. Each of the candidate set had the modelling units randomly chosen as well as randomly ordered. The first candidate set was a small set of only 5 modelling units randomly chosen and ordered. This was followed by a larger set of 12 modelling units and 18 modelling units, then the whole 23 modelling units available, as per Table 5.3.

Table 5.3: The 4 modelling units' sets

<b>Modelling Unit</b>	<b>23</b>	<b>18</b>	<b>12</b>	<b>5</b>	<b>Modelling Unit</b>	<b>23</b>	<b>18</b>	<b>12</b>	<b>5</b>
1. R/N	✓	✓	✓	✗	13. R	✓	✗	✓	✗
2. S/H	✓	✓	✓	✗	14. W	✓	✓	✓	✓
3. P	✓	✓	✓	✗	15. TS	✓	✓	✗	✗
4. ED	✓	✓	✓	✗	16. CP	✓	✓	✗	✓
5. Dline	✓	✓	✓	✓	17. VD	✓	✓	✗	✗
6. AT	✓	✓	✓	✗	18. ST	✓	✓	✗	✗
7. PO	✓	✓	✓	✗	19. IVF	✓	✓	✗	✗
8. RTO	✓	✗	✓	✗	20. EAV	✓	✓	✗	✗
9. MT	✓	✓	✗	✗	21. REM	✓	✗	✗	✓
10. MOJ	✓	✗	✗	✗	22. MMR	✓	✓	✗	✗
11. RTD	✓	✓	✓	✗	23. COMP	✓	✗	✗	✗
12. C	✓	✓	✓	✓					

### 5.5.2 Identifying Candidate Sequences

Each of those four sets will then be ordered in different sequences, creating 8 candidate processes, in an attempt to evaluate the preferred sequence/order. Each set and its

sequence together represent the process which is to be evaluated in the remainder of this chapter. The eight proposed processes, illustrated in Table 5.4, can be categorized in 3 main categories:

- A. Sets 1, 2 and 3 represents only 5 randomly chosen and ordered modelling units
- B. FC1 represents randomly ordering the 18 modelling units, FC2 represents randomly ordering the 23 modelling units and FC3 represents randomly ordering the 12 modelling units
- C. Logic1 and 2 represents all 23 modelling units logically ordered based on the modelling units' dependencies, as discussed in section 5.4.

Table 5.4: The proposed modelling units' processes

	<b>Set1</b>	<b>Set2</b>	<b>Set3</b>	<b>FC1</b>	<b>FC2</b>	<b>FC3</b>	<b>Logic1</b>	<b>Logic2</b>
1	R	W	VD	RN	RN	R	RN	RN
2	EAV	C	AT	MT	MT	RTD	W	W
3	REM	DLine	R	W	W	RTO	REM	REM
4	P	CP	MMR	ED	EAV	ED	TS	C
5	W	REM	EAV	EAV	IVF	P	EAV	DLine
6				IVF	REM	RN	IVF	TS
7				CP	CP	W	CP	IVF
8				VD	VD	C	VD	EAV
9				TS	TS	DLine	SH	CP
10				ST	ED	AT	MMR	VD
11				S/H	ST	S/H	MT	SH
12				MMR	RTO	PO	C	MMR
13				C	SH		P	MT
14				P	MMR		ED	P
15				PO	C		DLine	ED
16				AT	P		RTO	RTO
17				RTD	PO		PO	PO
18				DLine	R		R	R
19					AT		AT	AT
20					RTD		RTD	RTD
21					DLine		ST	ST
22					COMP		COMP	COMP
23					MOJ		MOJ	MOJ

The 8 candidate constructed processes are shown as flowcharts (Figures 5.4 - 5.12) and each discussed in the following subsections. This chapter will present and discuss each candidate process using commonly used symbols (Llego 2016):

- 1- Oval: A terminal symbol representing the process start/end
- 2- Rectangle: A process symbol representing processing of action
- 3- Diamond: A decision representing the process logical test
- 4- Arrow Lines & Arrow heads: this shows the process decision flow from one modelling unit to the other.

#### 5.5.2.1 The first set of 5 real-time modelling units' process (Set 1):

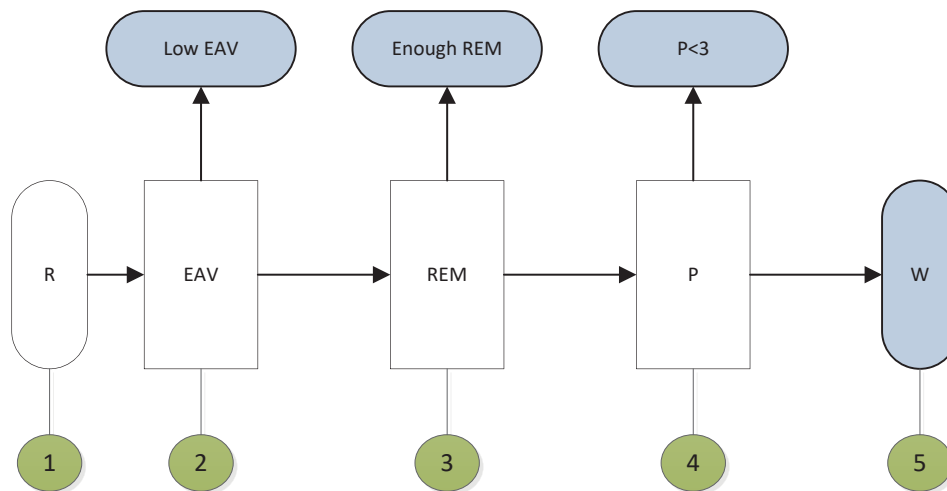


Figure 5-4 Set 1 representing the first set of 5 modelling units

The process shown in Figure 5-4 represents the first of three candidate processes with 5 modelling units. The process requires running each step (identified by bullet points below) until a meeting conflict is successfully resolved. Hence not all steps of the process must run each time; As the conflicting meetings in some cases can be resolved early in the process and the process ends at the step/ modelling unit that resolved the meeting conflict. Each modelling unit is identified by a number in a green box, to help identify the modelling units represented in each process. While each termination/exit point is highlighted as a blue oval task in the process. However, in some instances all the steps are run utilizing all the process modelling units as described in what follows:

- ❖ R “Retry attempts”: The process starts by checking if this is a recurring meeting that can be postponed to its next occurrence. If not, the process proceeds with validating the next modelling unit “Execution Accrued Value”
- ❖ EAV “Execution Accrued Value”: The process then checks the accrued value from previous meetings. If there is enough value to move/reschedule the meetings, then the process would adjust the meetings accordingly. If not, the process proceeds with validating the next modelling unit “Remaining Time”



- ❖ REM “Remaining Time”: If the Remaining time is enough to withhold the meeting, then the meeting can be completed. However, if the REM is not enough then the process will move to the next check, validating the Priority
- ❖ P “Priority”: This ensures priority meetings happen on their scheduled times. If the priority value is less than or equal to the priority threshold, that is,  $P \leq 3$ , then the meeting is considered a low priority meeting and process quits logging “successful as low priority meeting”. However, if the priority is greater than the priority threshold, that is,  $P > 3$ , then this is considered a high priority meeting and the process will move to the next step, validating the “Sample time”
- ❖ W “Warn/Sample time”: If the current duration is more than the sample time the process samples the meeting values, the meeting is marked as successful and the success rate is incremented. While if the process could not resolve a meeting conflict, then the meeting is marked as failed, decreasing the process success rate.

#### 5.5.2.2 The 2<sup>nd</sup> set of 5 real-time modelling units’ process (Set 2):

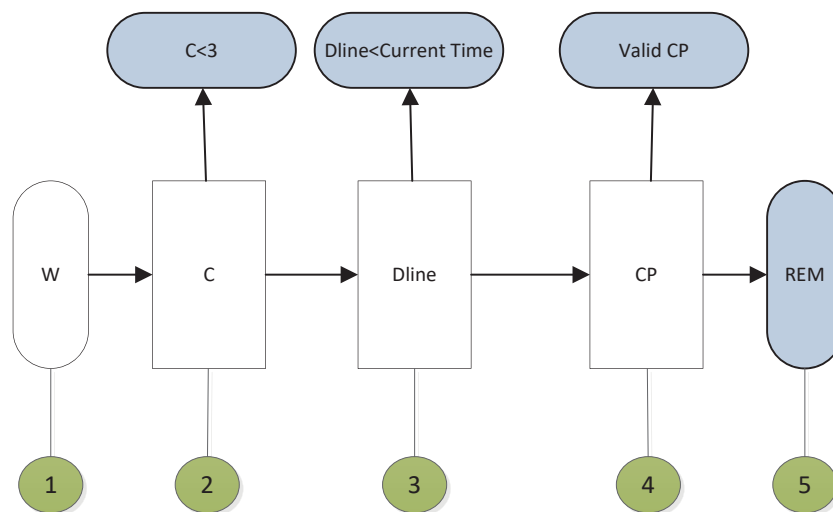


Figure 5-5 Set 2 representing the 2<sup>nd</sup> set of 5 modelling units

The process shown in Figure 5-5 represents the 2<sup>nd</sup> of three candidate processes with 5 modelling units. The process will run the steps (identified by bullet points below) until a meeting conflict is successfully resolved as described in what follows:

- ❖ W “Warn/Sample time”: If the current duration is more than the sample time the process samples the meeting values. However, if the sample time is less than or equal to the current duration the process will move to the next step, validating the Criticality
- ❖ C “Criticality”: This ensures critical meetings happen on their scheduled times and less critical meetings are the ones rescheduled not the critical ones if possible. If the

criticality value is less than or equal to the highly critical threshold, that is,  $C \leq 3$  then the meeting is considered a non-critical meeting and the process quits logging “successful as non-critical meeting”. However, if the criticality is greater than the criticality threshold, that is,  $C > 3$ , then this is considered a critical meeting and the process will move to the next step, validating the Deadline

- ❖ Dline “Deadline”: This step checks if the deadline has passed or not by comparing the current time versus the meeting start time (assuming that the deadline is the meeting start time). If the deadline has passed the process will move to the next step, validating Checkpoint
- ❖ CP “Checkpoint”: If the checkpoint is less than or equal to current duration then the process skips logging “too early for checkpoint”. However, if the checkpoint is greater than the current duration then the process will move to the next check, validating the “Remaining Time”
- ❖ REM “Remaining Time”: If the Remaining time is more than or equal to the estimated duration, then the task can be completed. The process quits logging “Successful as REM more than ED”. However, if the REM is less than the ED then the process will move to the next check, validating the warn time. then the meeting is marked as successful and the success rate is incremented. While if the process could not resolve a meeting conflict, before its REM, then the meeting is marked as failed, decreasing the process success rate.

### 5.5.2.3 The 3<sup>rd</sup> set of 5 real-time modelling units’ process (Set 3):

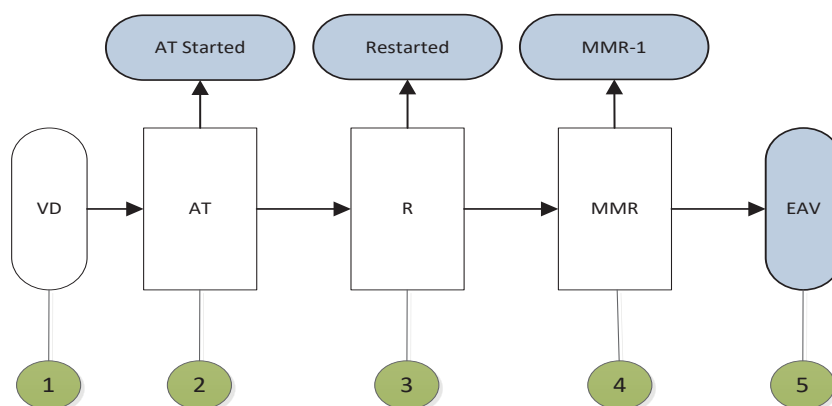


Figure 5-6 Set 3 representing the 3<sup>rd</sup> set of 5 modelling units

The process shown in Figure 5-6 represents the 3<sup>rd</sup> of three candidate processes with 5 modelling units. The process will run the steps (identified by bullet points below) until a meeting conflict is successfully resolved as described in what follows:

- ❖ VD “Validity Duration”: If the validity duration is greater than current time then this checkpoint is considered valid and has not expired. Then the process increments the validity duration by 1. That is,  $VD+ = 1$ , to give a new expiry to the new checkpoint values then quits logging “Successful After updating VD”. However, if the VD is less than or equal to the current time then process identifies that this checkpoint has expired and it will move to the next check, validating the “Alternate Task” after saving the current values as a new checkpoint
- ❖ AT “Alternate Task”: If an alternate meeting exists and can be attended, then the process quits logging “Successful after starting the AT”. However, if there is no alternate meeting (alternate task value is less than or equal to zero, that is,  $AT \leq 0$ ), then the process will move to the next step checking the next modelling unit “Retry attempts”
- ❖ R “Retry attempts”: If this is a recurring meeting then the process will postpone this meeting to the next recurrence. If not, then the process will move to the next step checking the “Maximum Miss ratio”
- ❖ MMR “Maximum Miss ratio”: If the MMR is greater than zero, then this considered a soft constraint which has not reached its MMR, and can still be violated so the process reduces the MMR by 1, that is,  $MMR- = 1$  and quits logging “successful with soft constraint MMR above Zero”. However, if the MMR is zero or less, then the process assumes that the meeting has exceeded its MMR and will move to the next step, validating the “Execution Accrued Value”
- ❖ EAV “Execution Accrued Value”: The process then checks the accrued value from previous meetings. If there is enough value to move/reschedule the meetings, then the process would adjust the meetings accordingly. If the process can reschedule the meeting, then the process is marked as successful and the success rate is incremented. While if the process could not resolve a meeting conflict, then the meeting is marked as failed, decreasing the process success rate.

#### 5.5.2.4 The 18 real-time modelling units' process (FC 1):

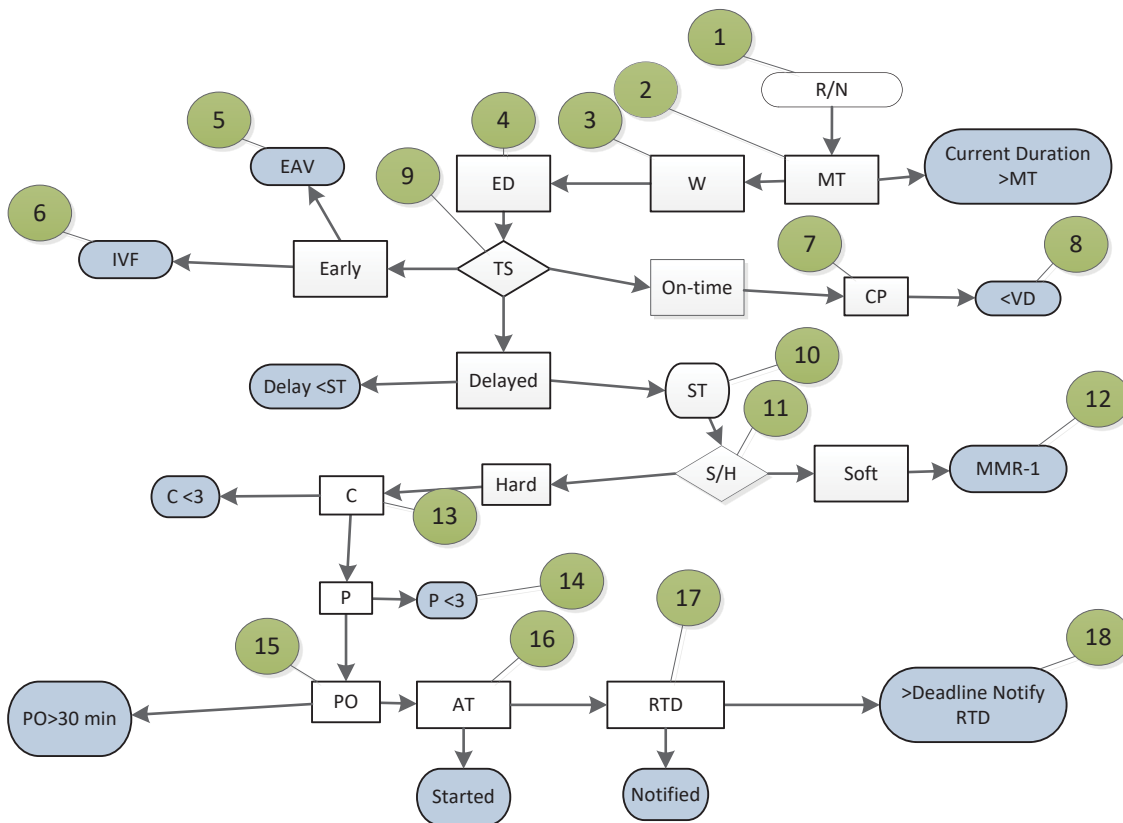


Figure 5-7 FC1 The 18 real-time modelling units' process

The process shown in Figure 5-7 represents the candidate process with 18 modelling units. The process will run the steps (identified by bullet points below) until a meeting conflict in successfully resolved as described in what follows:

- ❖ R/N “Real or Not”: If the meeting is time-dependent this value would be “1” and the process checks the rest of the modelling units, starting with the “Minimum Time” modelling unit. However, if “R/N” has a value of “0” then all other modelling units should be “0” and the process quits logging “Non real-time task”
- ❖ MT “Minimum Time”: This ensures that the meeting has taken places; hence if the current duration of a meeting is more than the MT value, the process quits logging “Successful as less than Minimum Time”. However, if the current duration of a meeting is less than or equal to MT value, then the process will move to the next step, validating the “Sample time”
- ❖ W “Warn/Sample time”: If the current duration is more than the sample time the process samples the meeting values. However, if the sample time is less than or equal to the current duration then the process will move to the next step, validating the “Estimated Duration”

- ❖ ED “Estimated Duration”: Which is the meeting scheduled duration and is compared to the current meeting duration. Based on this comparison the Task Status is updated to one of the following three values:
  - I. Early: If ED is greater than the current duration, then the “Task Status” TS is considered early and the process increments IVF and EAV by 1; that is,  $IVF+ = 1$  and  $EAV+ = 1$ . Then the process quits logging “successful early”
  - II. On-Time: If ED is equal to the current duration, then the “Task Status” TS is considered “on-Time” and the process creates a checkpoint and updates the validity duration
  - III. Delayed: If ED is less than the current duration, then the process sets the “Task Status” TS as “Delayed”. Then the process will move to the next step, validating the “Slack Time”
- ❖ ST “Slack Time”: If the meeting is delayed, however there is enough ST for it to complete before its ED i.e. there was enough buffer for the delay then the meeting would still succeed and finish before within an acceptable time. However, if the delay was more than the ST i.e. no room for that much delay then the process will move to the next step, validating next modelling unit “Soft or Hard”
- ❖ S/H “Soft or Hard”: If the meeting has a “Soft” constraint, then the next modelling unit, “maximum miss ratio” is checked. While if the meeting has a “Hard” constraint, then the process would move to the next step, validating the meeting Criticality
- ❖ MMR “Maximum Miss Ratio”: If the MMR is greater than zero, then this soft constraint has not reached its MMR, and can still be violated so the process reduces the MMR by 1, and quits logging “successful with soft constraint MMR above Zero”
- ❖ C “Criticality”: This ensures critical meetings happen on their scheduled times and less critical meetings are the ones rescheduled not the critical ones if possible. If the criticality value is less than or equal to the highly critical threshold, that is,  $C \leq 3$  then the meeting is considered a non-critical meeting and the process quits logging “successful as non-critical meeting”. However, if the criticality is greater than the criticality threshold, that is,  $C > 3$ , then this is considered a critical meeting and the process will move to the next step, validating the Priority
- ❖ P “Priority”: This ensures priority meetings happen on their scheduled times. If the priority value is less than or equal to the priority threshold, that is,  $P \leq 3$ , then the meeting is considered a low priority meeting and process quits logging “successful

as low priority meeting”. However, if the priority is greater than the priority threshold, that is,  $P > 3$ , then this is considered a high priority meeting and the process will move to the next step, validating “Periodic Occurrence”

- ❖ PO “Periodic Occurrence”: Which presents meeting re-occurrence and/or available time slots for meeting booking. The PO modelling unit has 2 checks:
  - i. If PO is between waiting threshold and 0, that is,  $0 > PO \leq 30$ , assuming that meetings run every 30 minutes, then the process will move to the next step, validating the “Alternate Task”
  - ii. If the periodic is more than the “waiting threshold”, that is,  $PO > 30$ , then the process will re-run the meeting and reduce the value by 1, that is,  $PO - 1$  and then the process quits logging “successful with PO”
- ❖ AT “Alternate Task”: If an alternate meeting exists and can be attended, then the process quits logging “Successful after starting the AT”. However, if there is no alternate meeting (alternate task value is less than or equal to zero, that is,  $AT \leq 0$ ), then the process will move to the next step checking the next modelling unit “Real-time Dependency”
- ❖ RTD “Real-time Dependency”: If the RTD is greater than zero, that is, RTD exists, then the process will notify the parent agent (RTD), whom is usually the meeting organizer, to take the necessary action, and the process quits logging “successful after notifying RTD”. However, if RTD is less than or equal to zero then the process assumes there is no dependent agent to notify and will move to the next step checking the deadline modelling unit
- ❖ DLine “Deadline”: This step checks if the deadline has passed or not by comparing the current time versus the meeting start time (assuming that the deadline is the meeting start time). If the deadline has passed yet any of the above steps has successfully resolved the meeting conflict i.e. enabling the attendees to attend the meeting, then the meeting is marked as successful and the success rate is incremented. While if the process could not resolve a meeting conflict, before its deadline, then the meeting is marked as failed, decreasing the process success rate.

### 5.5.2.5 The set of 23 real-time modelling units' process (FC 2):

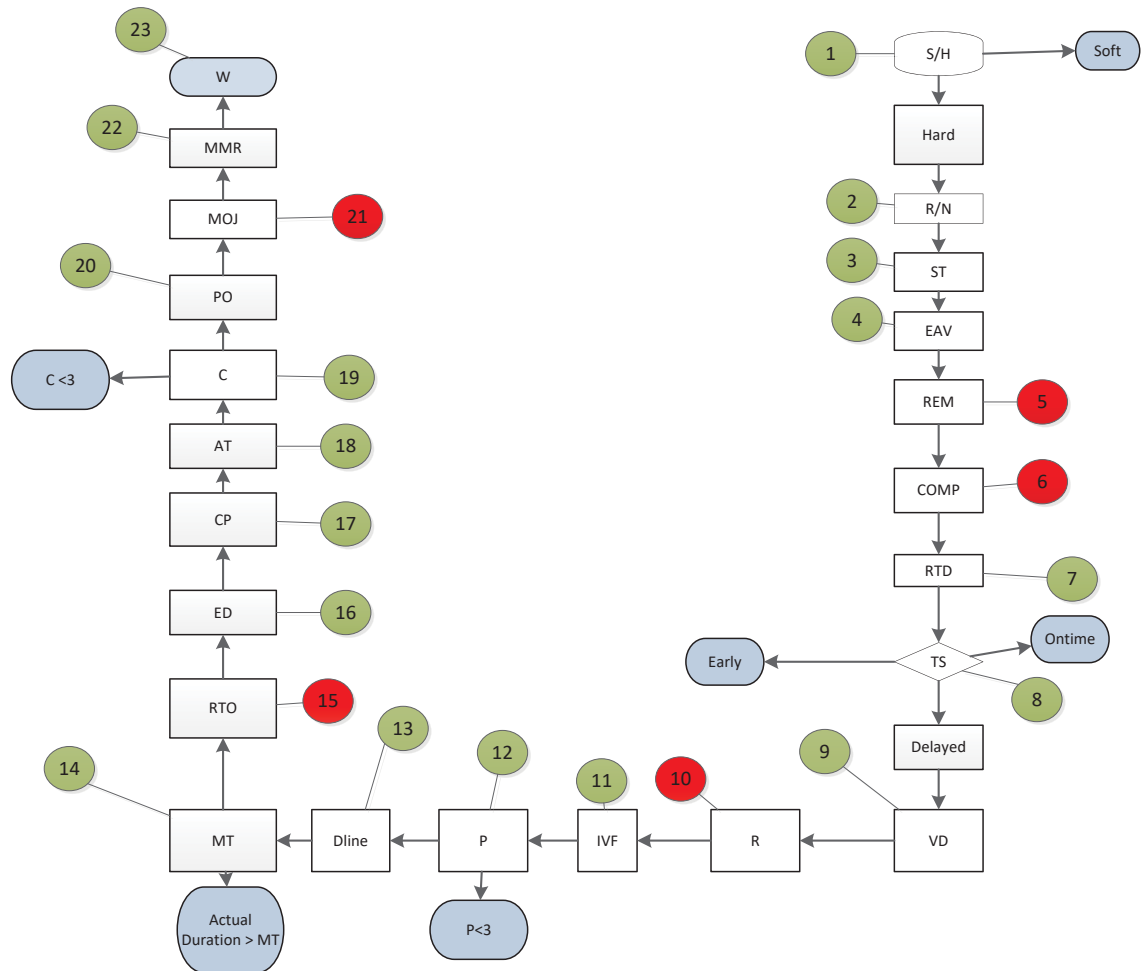


Figure 5-8 FC2 The 23 real-time modelling units' process

The process shown in Figure 5-8 represents the candidate process with 23 modelling units. Each modelling unit is identified by a number in a green box, to help identify the 23 modelling units represented in each process, while there are 5 modelling units in red boxes which were considered duplicates and removed to create the 18 modelling units' process. The process will run the steps (identified by bullet points below) until a meeting conflict is successfully resolved as described in what follows:

- ❖ If the meeting “S/H” modelling unit has a “Hard” constraint, then the process would move to the next step, validating the meeting “Real or Not”
- ❖ R/N “Real or Not”: If the meeting is time-dependent this value would be “1” and the process will check the rest of the modelling units, starting with the “Slack Time”. However, if “R/N” has a value of “0” then all other modelling units should be “0” and the process quits logging “Non real-time task”

- ❖ ST “Slack Time”: If the meeting is delayed, however there is enough ST for it to complete before its ED i.e. there was enough buffer for the delay then the meeting would still succeed and finish before within an acceptable time. However, if the delay was more than the ST i.e. no room for that much delay then the process will move to the next step, validating next modelling unit “Execution Accrued Value”
- ❖ EAV “Execution Accrued Value”: The process then checks the accrued value from previous meetings. If there is enough value to move/reschedule the meetings, then the process would adjust the meetings accordingly. If not, the process proceeds with validating the next modelling unit “Remaining Time”
- ❖ REM “Remaining Time”: If the Remaining time is enough to withhold the meeting, then the meeting can be completed. However, if the REM is not enough then the process will move to the next check, validating the “Composite”
- ❖ COMP “Composite”: If the COMP is greater than zero, that is, COMP values exists, then the process will quit logging “Comp>0”. However, if COMP is less than or equal to zero then the process will move to the next step checking the “Real-time Dependency” modelling unit
- ❖ RTD “Real-time Dependency”: If the RTD is greater than zero, that is, RTD exists, then the process will notify the parent agent (RTD), whom is usually the meeting organizer, to take the necessary action, and the process quits logging “successful after notifying RTD”. However, if RTD is less than or equal to zero then the process assumes there is no dependent agent to notify and will move to the next step checking the “Task Status” modelling unit
- ❖ TS “Task Status”: which is the meeting scheduled duration and can have one the following 3 values:
  - i. Early: If ED is greater than the current duration, then the “Task Status” TS is considered early and the process quits logging “successful early”
  - ii. On-Time: If ED is equal to the current duration, then the “Task Status” TS is considered “on-Time” and the process quits logging “On-Time”
  - iii. Delayed: If ED is less than the current duration, then the process sets the “Task Status” TS as “Delayed”. Then the process will move to the next step, checking the “Validity Duration”
- ❖ VD “Validity Duration”: If the validity duration is greater than current time then this checkpoint is considered valid and has not expired. Then the process increments the



validity duration by 1. That is,  $VD+ = 1$ , to give a new expiry to the new checkpoint values then quits logging “Successful After updating VD”. However, if the VD is less than or equal to the current time then process identifies that this checkpoint has expired and it will move to the next check, validating the “Retry attempts” modelling unit, after saving the current values as a new checkpoint

- ❖ R “Retry attempts”: If this is a recurring meeting then the process will postpone this meeting to the next recurrence. If not, then the process will move to the next step checking the “Instant Value Function”
- ❖ IVF “Instant Value Function”: If the total accrued value from the meetings is higher than the delay then the meeting can be successfully extended. If not, then the process will move to the next step checking the Priority
- ❖ P “Priority”: This ensures priority meetings happen on their scheduled times. If the priority value is less than or equal to the priority threshold, that is,  $P \leq 3$ , then the meeting is considered a low priority meeting and process quits logging “successful as low priority meeting”. However, if the priority is greater than the priority threshold, that is,  $P > 3$ , then this is considered a high priority meeting and the process will move to the next step, validating the Deadline
- ❖ Dline “Deadline”: This step checks if the deadline has passed or not by comparing the current time versus the meeting start time (assuming that the deadline is the meeting start time). If the deadline has passed the process will move to the next step, validating the “Minimum Time”
- ❖ MT “Minimum Time”: This ensures that the meeting has taken places; hence if the current duration of a meeting is more than MT value, the process quits logging “Successful as less than Minimum Time”. However, if the current duration of a meeting is less than or equal to MT value, then the process will move to the next step, validating the “Real-time Order”
- ❖ RTO “Real-time Order”: The process will check the time between each meeting and if there is enough time for the meeting to be delayed then process quits logging “successful with enough RTO”. If not then the process will move to the next step, validating the “Estimated Duration”
- ❖ ED “Estimated Duration”: Which is the meeting scheduled duration and is compared to the current meeting duration. If the actual duration is less than the ED the process quits logging “meeting still running”. However, if the duration is more than the ED

i.e. the meeting has gone for longer than expected then the process will move to the next step, validating the checkpoint

- ❖ CP “Checkpoint”: If the checkpoint is less than or equal to current duration then the process skips logging “too early for checkpoint”. However, if the checkpoint is greater than the current duration then the process will move to the next check, validating the “Alternate Task”
- ❖ AT “Alternate Task”: If an alternate meeting exists and can be attended, then the process quits logging “Successful after starting the AT”. However, if there is no alternate meeting (alternate task value is less than or equal to zero, that is,  $AT \leq 0$ ), then the process will move to the next step checking the next modelling unit Criticality
- ❖ C “Criticality”: This ensures critical meetings happen on their scheduled times and less critical meetings are the ones rescheduled not the critical ones if possible. If the criticality value is less than or equal to the highly critical threshold, that is,  $C \leq 3$  then the meeting is considered a non-critical meeting and the process quits logging “successful as non-critical meeting”. However, if the criticality is greater than the criticality threshold, that is,  $C > 3$ , then this is considered a critical meeting and the process will move to the next step, validating the “Periodic Occurrence”
- ❖ PO “Periodic Occurrence “: Which presents meeting re-occurrence and/or available time slots for meeting booking. The PO modelling unit has 2 checks:
  - iii. If PO is between waiting threshold and 0, that is,  $0 > PO \leq 30$ , assuming that meetings run every 30 minutes, then the process will move to the next step, validating the “Maximum Output Jitter”
  - iv. If the periodic is more than the “waiting threshold”, that is,  $PO > 30$ , then the process will re-run the meeting and reduce the value by 1, that is,  $PO - 1$  and then the process quits logging “successful with PO”
- ❖ MOJ “Maximum Output Jitter”: This step compares the MOJ (difference between shortest meeting time and maximum meeting time). If the MOJ is larger than this meeting delay, the process quits logging “successful MOJ”. If not, the process will move to the next step, validating the “Maximum Miss Ratio”
- ❖ MMR “Maximum Miss Ratio”: If the MMR is greater than zero, then this considered a soft constraint which has not reached its MMR, and can still be violated so the process reduces the MMR by 1, that is,  $MMR - 1$  and quits logging “successful with

soft constraint MMR above Zero”. However, if the MMR is zero or less, then the process assumes that the meeting has exceeded its MMR and will move to the next step, validating the “Sample time”

- ❖ W “Warn/Sample time”: If the current duration is more than the sample time the process samples the meeting values. If the Sample time has passed yet any of the above steps has successfully resolved the meeting conflict i.e. enabling the attendees to attend the meeting, then the meeting is marked as successful and the success rate is incremented. While if the process could not resolve a meeting conflict, then the meeting is marked as failed, decreasing the process success rate.

#### 5.5.2.6 The set of 12 real-time modelling units’ process (FC 3):

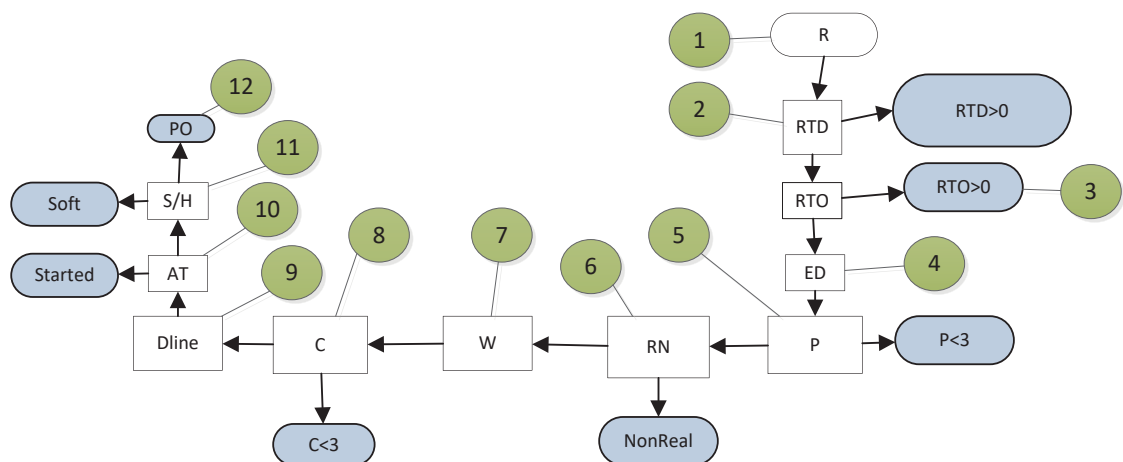


Figure 5-9 FC3 The 12 real-time modelling units’ process

The process shown in Figure 5-9 represents the candidate process with 12 modelling units. The process will run the steps (identified by bullet points below) until a meeting conflict is successfully resolved as described in what follows:

- ❖ R “Retry attempts”: If this is a recurring meeting then the process will postpone this meeting to the next recurrence. If not, then the process will move to the next step checking the “Real-time Dependency”
- ❖ RTD “Real-time Dependency”: If the RTD is greater than zero, that is, RTD exists, then the process will notify the parent agent (RTD), whom is usually the meeting organizer, to take the necessary action, and the process quits logging “successful after notifying RTD”. However, if RTD is less than or equal to zero then the process

assumes there is no dependent agent to notify and will move to the next step checking the “Real-time Order” modelling unit

- ❖ RTO “Real-time Order”: The process will check the time between each meeting and if there is enough time for the meeting to be delayed then process quits logging “successful with enough RTO”. If not then the process will move to the next step, validating the “Estimated Duration”
- ❖ ED “Estimated Duration”: Which is the meeting scheduled duration and is compared to the current meeting duration to identify if the meeting is:
  - a. Early: If ED is greater than the current duration, then the meeting is considered early. Then the process quits logging “successful early”
  - b. On-Time: If ED is equal to the current duration, then the meeting is considered “on-Time”
  - c. Delayed: If ED is less than the current duration, then the process will move to the next step, validating the Priority
- ❖ “P” Priority: This ensures priority meetings happen on their scheduled times. If the priority value is less than or equal to the priority threshold, that is,  $P \leq 3$ , then the meeting is considered a low priority meeting and process quits logging “successful as low priority meeting”. However, if the priority is greater than the priority threshold, that is,  $P > 3$ , then this is considered a high priority meeting and the process will move to the next step, validating “Real or Not”
- ❖ R/N “Real or Not”: If the meeting is time-dependent this value would be “1” and the process will check the rest of the modelling units, starting with the estimated duration “ED”. However, if “R/N” has a value of “0” then the process will move to the next step, validating the “Sample time”
- ❖ W “Warn/Sample time”: If the current duration is more than the sample time the process samples the meeting values. However, if the sample time is less than or equal to the current duration then the process will move to the next step, validating the Criticality
- ❖ C “Criticality”: This ensures critical meetings happen on their scheduled times and less critical meetings are the ones rescheduled not the critical ones if possible. If the criticality value is less than or equal to the highly critical threshold, that is,  $C \leq 3$  then the meeting is considered a non-critical meeting and the process quits logging “successful as non-critical meeting”. However, if the criticality is greater than the

criticality threshold, that is,  $C > 3$ , then this is considered a critical meeting and the process will move to the next step, validating the deadline

- ❖ Dline “Deadline”: This step checks if the deadline has passed or not by comparing the current time versus the meeting start time (assuming that the deadline is the meeting start time). If the deadline has passed the process will move to the next step, validating “Alternate Task”
- ❖ AT “Alternate Task”: If an alternate meeting exists and can be attended, then the process quits logging “Successful after starting the AT”. However, if there is no alternate meeting (alternate task value is less than or equal to zero, that is,  $AT \leq 0$ ), then the process will move to the next step checking the next modelling unit “Soft or Hard”
- ❖ S/H “Soft or Hard”: If the meeting has a “Soft” constraint, then the process exists reporting a “soft constraint”. However, if the meeting “S/H” modelling unit has a “Hard” constraint, then the process would move to the next step, validating the meeting “Periodic Occurrence”
- ❖ PO “Periodic Occurrence”: Which presents meeting re-occurrence and/or available time slots for meeting booking. The PO modelling unit has 2 checks:
  - i. If the periodic is more than the “waiting threshold”, that is,  $PO > 30$ , then the process will re-run the meeting and reduce the value by 1, that is,  $PO - 1$  and then the process quits logging “successful with PO”
  - ii. If PO is between waiting threshold and 0, that is,  $0 > PO \leq 30$ , assuming that meetings run every 30 minutes, yet any of the above steps has successfully resolved the meeting conflict i.e. enabling the attendees to attend the meeting, then the meeting is marked as successful and the success rate is incremented. While if the process could not resolve a meeting conflict then the meeting is marked as failed, decreasing the process success rate.

5.5.2.7 The first set of 23 real-time modelling units’ process logically sequenced (Logic 1):

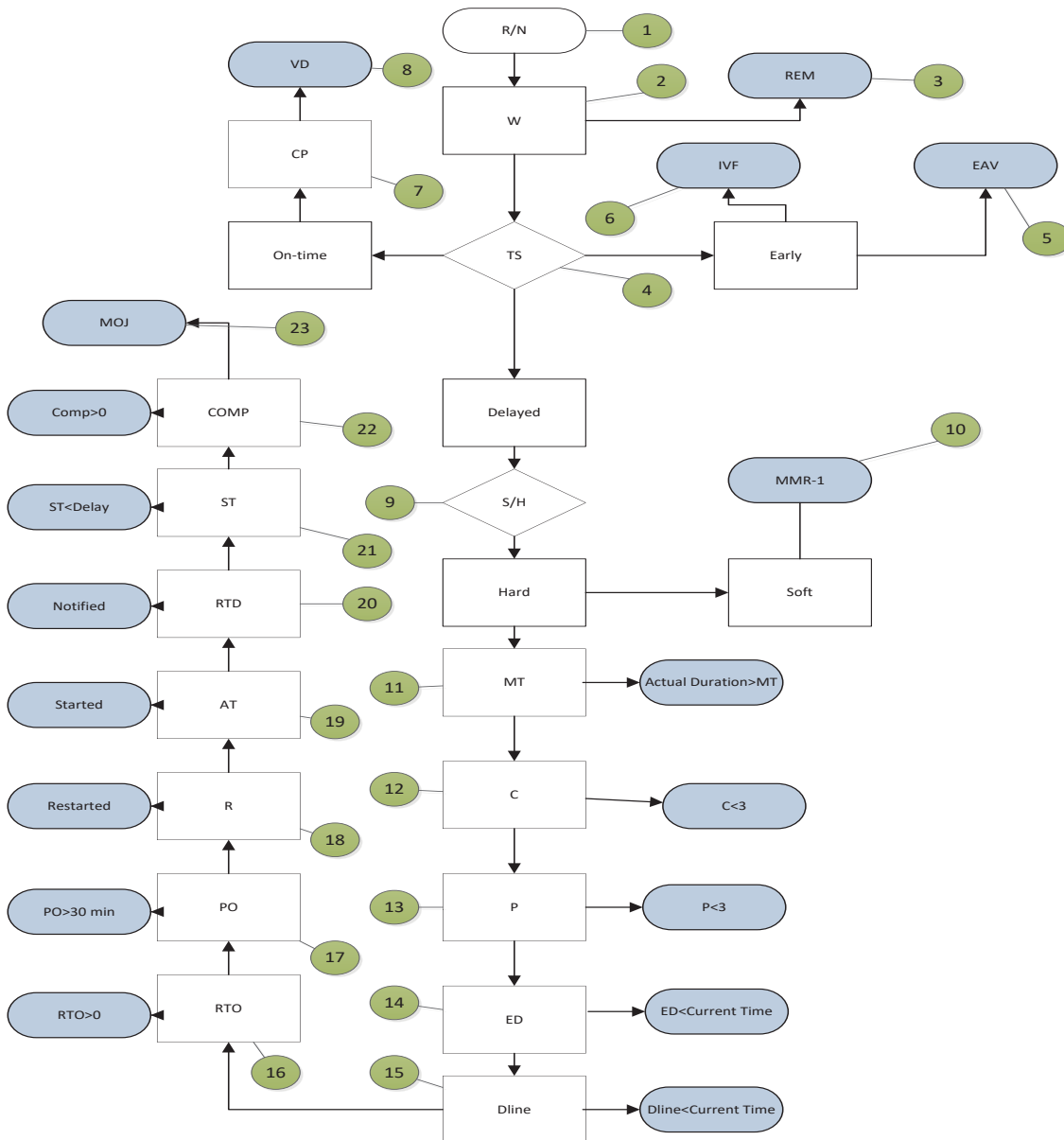


Figure 5-10 Logic1 The 23 real-time modelling units’ process

The process shown in Figure 5-10 represents the candidate process with 23 modelling units ordered in the first of two logical flow charts. The logical flow chart is based on the modelling units’ relationships and dependencies, as discussed in section 5.4. The process will run the steps (identified by bullet points below) until a meeting conflict in successfully resolved as described in what follows:

- ❖ R/N “Real or Not”: If the meeting is time-dependent this value would be “1” and the process will check the rest of the modelling units, starting with the “Sample time”

However, if “R/N” has a value of “0” then all other modelling units should be “0” and the process quits logging “Non real-time task”

- ❖ W “Warn/Sample time”: If the current duration is more than the sample time the process samples the meeting values. However, if the sample time is less than or equal to the current duration then the process will move to the next step, validating the “Task Status”
- ❖ TS “Task Status”: Which is the meeting scheduled duration and can have one the following 3 values:
  - i. Early: If ED is greater than the current duration, then the TS is considered early and the process increments IVF and EAV by 1; that is,  $IVF+ = 1$  and  $EAV+ = 1$ . Then the process quits logging “successful early”
  - ii. On-Time: If ED is equal to the current duration, then the “Task Status” TS is considered “on-Time” and the process quits logging “successful On-Time”
  - iii. Delayed: If ED is less than the current duration, then the process sets the “Task Status” TS as “Delayed”. Then the process will move to the next step, validating the “Soft or Hard” modelling unit
- ❖ S/H “Soft or Hard”: If the meeting has a “Soft” constraint, then the next modelling unit, maximum miss ratio “MMR”, is checked. While if the meeting has a “Hard” constraint, then the process would move to the next step, validating the “Minimum Time” modelling unit
- ❖ MMR “Maximum Miss Ratio”: If the MMR is greater than zero, that is,  $MMR > 0$ , then this soft constraint has not reached its MMR, and can still be violated so the process reduces the MMR by 1, that is,  $MMR- = 1$  and quits logging “successful with soft constraint MMR above Zero”
- ❖ MT “Minimum Time”: This ensures that the meeting has taken places; hence if the current duration of a meeting is more than MT value, the process quits logging “Successful as less than Minimum Time”. However, if the current duration of a meeting is less than or equal to MT value, then the process will move to the next step, validating the Criticality
- ❖ C “Criticality”: This ensures critical meetings happen on their scheduled times and less critical meetings are the ones rescheduled not the critical ones if possible. If the criticality value is less than or equal to the highly critical threshold, that is,  $C \leq 3$

then the meeting is considered a non-critical meeting and the process quits logging “successful as non-critical meeting”. However, if the criticality is greater than the criticality threshold, that is,  $C > 3$ , then this is considered a critical meeting and the process will move to the next step, validating the Priority

- ❖ “P” Priority: This ensures priority meetings happen on their scheduled times. If the priority value is less than or equal to the priority threshold, that is,  $P \leq 3$ , then the meeting is considered a low priority meeting and process quits logging “successful as low priority meeting”. However, if the priority is greater than the priority threshold, that is,  $P > 3$ , then this is considered a high priority meeting and the process will move to the next step, validating “Estimated Duration”
- ❖ ED “Estimated Duration”: Which is the meeting scheduled duration and is compared to the current meeting duration. If the actual duration is less than the ED the process quits logging “meeting still running”. However, if the duration is more than the ED i.e. the meeting has gone for longer than expected then the process will move to the next step, validating the Deadline
- ❖ DLine “Deadline”: This step checks if the meeting deadlines including the meeting start time and end time. If the deadline has passed then the process will move to the next step, validating the “Real-Time Order”
- ❖ RTO “Real-Time Order”: The process will check the time between each meeting and if there is enough time for the meeting to be delayed then process quits logging “successful with enough RTO”. If not then the process will move to the next step, validating the “Periodic Occurrence”
- ❖ PO “Periodic Occurrence”: Which presents meeting re-occurrence and/or available time slots for meeting booking. The PO modelling unit has 2 checks:
  - i. If PO is between waiting threshold and 0, that is,  $0 > PO \leq 30$ , assuming that meetings run every 30 minutes, then the process will move to the next step, validating the “Retry attempts”
  - ii. If the periodic is more than the “waiting threshold”, that is,  $PO > 30$ , then the process will re-run the meeting and reduce the value by 1, that is,  $PO - 1$  and then the process quits logging “successful with PO”.
- ❖ R “Retry attempts”: If this is a recurring meeting then the process will postpone this meeting to the next recurrence. If not, then the process will move to the next step checking the “Alternate Task”



- ❖ AT “Alternate Task”: If an alternate meeting exists and can be attended, then the process quits logging “Successful after starting the AT”. However, if there is no alternate meeting (alternate task value is less than or equal to zero, that is,  $AT \leq 0$ ), then the process will move to the next step checking the next modelling unit RT Dependency “Real-Time Dependency”
- ❖ RTD “Real-Time Dependency”: If the RTD is greater than zero, that is, RTD exists, then the process will notify the parent agent (RTD), whom is usually the meeting organizer, to take the necessary action, and the process quits logging “successful after notifying RTD”. However, if RTD is less than or equal to zero then the process assumes there is no dependent agent to notify and will move to the next step checking the “Slack Time” modelling unit
- ❖ ST “Slack Time”: If the meeting is delayed, however there is enough ST for it to complete before its ED i.e. there was enough buffer for the delay then the meeting would still successes and finish before within an acceptable time. However, if the delay was more than the ST i.e. no room for that much delay then the process will move to the next step, validating next modelling unit “Composite”
- ❖ COMP “Composite”: If the COMP is greater than zero, that is, COMP values exists, then the process will quit logging “Comp>0”. However, if COMP is less than or equal to zero then the process will move to the next step checking the “Maximum Output Jitter” modelling unit
- ❖ MOJ “Maximum Output Jitter”: This step compares the MOJ (difference between shortest meeting time and maximum meeting time). If the MOJ is larger this meeting delay, yet any of the above steps has successfully resolved the meeting conflict i.e. enabling the attendees to attend the meeting, then the meeting is marked as successful and the success rate is incremented. While if the process could not resolve a meeting conflict then the meeting is marked as failed, decreasing the process success rate.

5.5.2.8 The 2<sup>nd</sup> set of 23 real-time modelling units’ process logically sequenced (Logic 2):

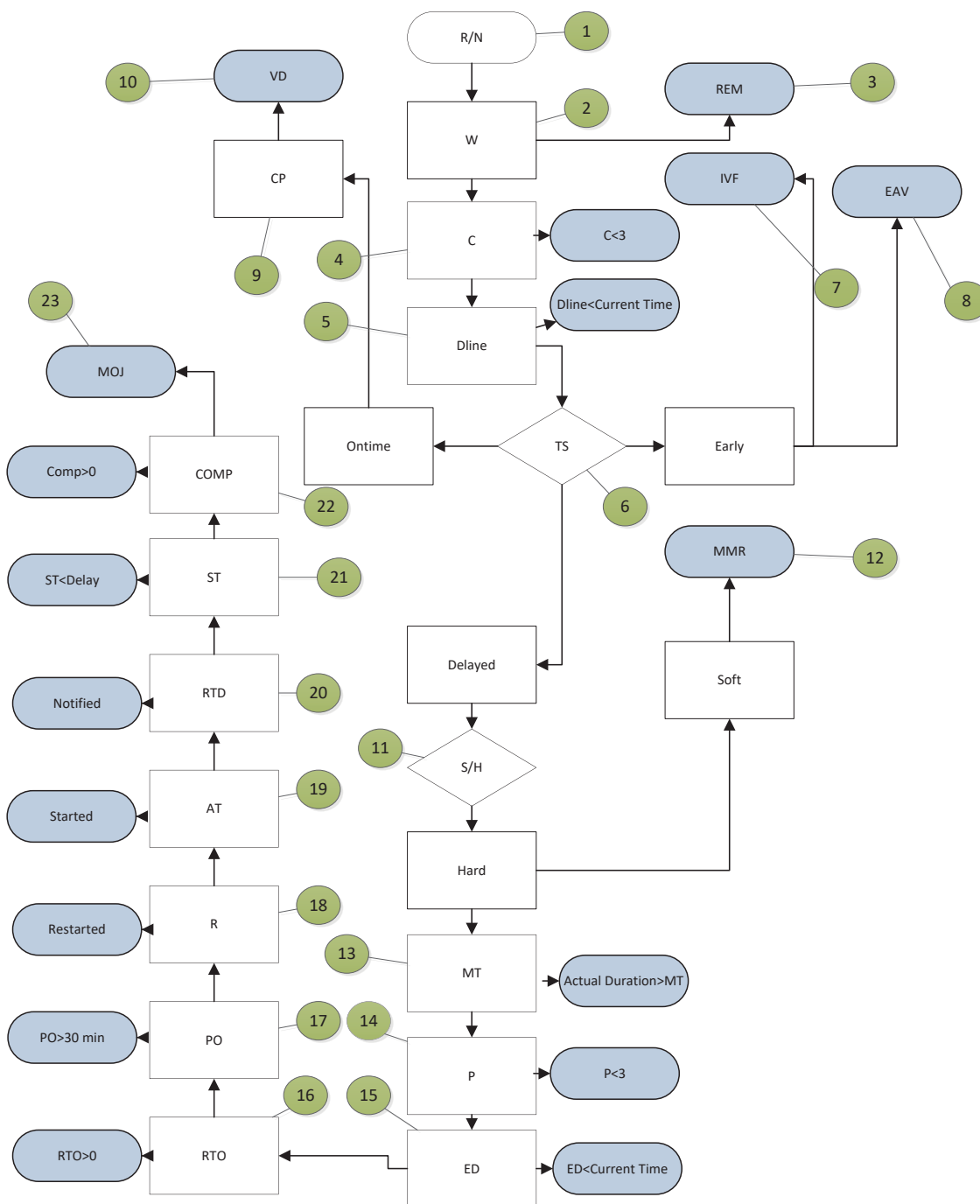


Figure 5-11 Logic2 The 23 real-time modelling units’ process

The process shown in Figure 5-11 represents the candidate process with 23 modelling units ordered in the 2<sup>nd</sup> of two logical flow charts. The logical flow chart is based on the modelling units’ relationships and dependencies, as discussed in section 5.4. The process will run the steps (identified by bullet points below) until a meeting conflict in successfully resolved as described in what follows:

- ❖ R/N “Real or Not”: If the meeting is time-dependent this value would be “1” and the process will check the rest of the modelling units, starting with the “sample time”. However, if “R/N” has a value of “0” then all other modelling units should be “0” and the process quits logging “Non real-time task”
- ❖ W “Warn/Sample time”: If the current duration is more than the sample time the process samples the meeting values. However, if the sample time is less than or equal to the current duration then the process will move to the next step, validating the Criticality
- ❖ C “Criticality”: This ensures critical meetings happen on their scheduled times and less critical meetings are the ones rescheduled not the critical ones if possible. If the criticality value is less than or equal to the highly critical threshold, that is,  $C \leq 3$  then the meeting is considered a non-critical meeting and the process quits logging “successful as non-critical meeting”. However, if the criticality is greater than the criticality threshold, that is,  $C > 3$ , then this is considered a critical meeting and the process will move to the next step, validating the Deadline
- ❖ DLine “Deadline”: This step checks if the deadline has passed or not by comparing the current time versus the meeting start time (assuming that the deadline is the meeting start time). If the deadline has passed then the process will move to the next step, validating the “Task Status”
- ❖ TS “Task Status”: Which is the meeting scheduled duration and can have one the following 3 values:
  - i. Early: If ED is greater than the current duration, then the “Task Status” TS is considered early and the process increments IVF and EAV by 1; that is,  $IVF+ = 1$  and  $EAV+ = 1$ . Then the process quits logging “successful early”
  - ii. On-Time: If ED is equal to the current duration, then the “Task Status” TS is considered “on-Time” and the process creates a checkpoint “CP” and updates the validity duration “VD”
  - iii. Delayed: If ED is less than the current duration, then the process sets the “Task Status” TS as “Delayed”. Then the process will move to the next step, validating the S/H
- ❖ S/H “Soft or Hard”: If the meeting has a “Soft” constraint, then the next modelling unit, maximum miss ratio “MMR”, is checked and accumulated ( $MMR+1$ ). While if

the meeting has a “Hard” constraint, then the process would move to the next step, validating the meeting minimum time “MT”

- ❖ MMR “Maximum Miss Ratio”: If the MMR is greater than zero, that is,  $MMR > 0$ , then this soft constraint has not reached its MMR, and can still be violated so the process reduces the MMR by 1, that is,  $MMR - = 1$  and quits logging “successful with soft constraint MMR above Zero”
- ❖ MT “Minimum Time”: This ensures that the meeting has taken places; hence if the current duration of a meeting is more than MT value, the process quits logging “Successful as less than Minimum Time”. However, if the current duration of a meeting is less than or equal to MT value, then the process will move to the next step, validating the Priority
- ❖ P “Priority”: This ensures priority meetings happen on their scheduled times. If the priority value is less than or equal to the priority threshold, that is,  $P \leq 3$ , then the meeting is considered a low priority meeting and process quits logging “successful as low priority meeting”. However, if the priority is greater than the priority threshold, that is,  $P > 3$ , then this is considered a high priority meeting and the process will move to the next step, validating “Estimated Duration”
- ❖ ED “Estimated Duration”: Which is the meeting scheduled duration and is compared to the current meeting duration. If the actual duration is less than the ED the process quits logging “meeting still running”. However, if the duration is more than the ED i.e. the meeting has gone for longer than expected then the process will move to the next step, validating “Real-Time Order”
- ❖ RTO “Real-Time Order”: The process will check the time between each meeting and if there is enough time for the meeting to be delayed then process quits logging “successful with enough RTO”. If not then the process will move to the next step, validating the “Periodic Occurrence”
- ❖ PO “Periodic Occurrence”: Which presents meeting re-occurrence and/or available time slots for meeting booking. The PO modelling unit has 2 checks:
  - i. If PO is between waiting threshold and 0, that is,  $0 > PO \leq 30$ , assuming that meetings run every 30 minutes, then the process will move to the next step, validating the “Retry attempts”

- ii. If the periodic is more than the “waiting threshold”, that is,  $PO > 30$ , then the process will re-run the meeting and reduce the value by 1, that is,  $PO = 1$  and then the process quits logging “successful with PO”
- ❖ R “Retry attempts”: If this is a recurring meeting then the process will postpone this meeting to the next recurrence. If not, then the process will move to the next step checking the Alternate task “AT”
- ❖ AT “Alternate Task”: If an alternate meeting exists and can be attended, then the process quits logging “Successful after starting the AT”. However, if there is no alternate meeting (alternate task value is less than or equal to zero, that is,  $AT \leq 0$ ), then the process will move to the next step checking the next modelling unit “Real-Time Dependency”
- ❖ RTD “Real-Time Dependency”: If the RTD is greater than zero, that is, RTD exists, then the process will notify the parent agent (RTD), whom is usually the meeting organizer, to take the necessary action, and the process quits logging “successful after notifying RTD”. However, if RTD is less than or equal to zero then the process assumes there is no dependent agent to notify and will move to the next step checking the “Slack Time” modelling unit
- ❖ ST “Slack Time”: If the meeting is delayed, however there is enough ST for it to complete before its ED i.e. there was enough buffer for the delay then the meeting would still successes and finish before within an acceptable time. However, if the delay was more than the ST i.e. no room for that much delay then the process will move to the next step, validating next modelling unit “Composite”
- ❖ COMP “Composite”: If the COMP is greater than zero, that is, COMP values exists, then the process will quit logging “Comp>0”. However, if COMP is less than or equal to zero then the process will move to the next step checking the “Maximum Output Jitter” modelling unit
- ❖ MOJ “Maximum Output Jitter”: This step compares the MOJ (difference between shortest meeting time and maximum meeting time). If the MOJ is larger this meeting delay, yet any of the above steps has successfully resolved the meeting conflict i.e. enabling the attendees to attend the meeting, then the meeting is marked as successful and the success rate is incremented. While if the process could not resolve a meeting conflict then the meeting is marked as failed, decreasing the process success rate.

This section presented the 8 proposed processes to be evaluated. In the following sections, simulations are used to test each process on two levels; first whether the collection of individual modelling units is considered a sufficient set, then if the candidate sequential set of modelling units is better in resolving meeting conflicts and whether the identified relationships between the modelling units hold.

## **5.6 Simulating the Candidate Processes**

Some characteristics of the domain of this simulation (calendar) are worth noting first. From an attendee's perspective, the action taken in not attending a meeting is similar to the one taken in cancelling a meeting. But the consequences to other attendees and the calendar rescheduling overall will depend on how many other attendees are affected. For a meeting with only 2 attendees, where one will not attend, the meeting can be considered cancelled. If the meeting has many more attendees, then the meeting might not be cancelled and the person need to only notify the meeting organiser of their absence. For example, in a lecture, the attendee can notify the lecturer, leaving the lecture unaffected.

For any transport delay, the transport times require updating. The meeting times that depend on this transport are also checked to validate any attendees' delay. The simulation validates the use of the modelling units in general, by getting a total of all successful meetings as a result of using the proposed modelling units. Even with the use of appropriate modelling units, not all meetings will succeed. However, using the modelling units will enable identifying a number of meetings as successful, e.g. either by starting a new task, or from identifying the slack time or accumulated gained time, or by notifying the meeting organiser. If a task does not have an alternate task, but the notified RTD (meeting organiser) starts an alternate task or generates an alternate plan, then the modelling unit is useful in making the initial task succeed. The modelling unit success rate is calculated by counting the successful meetings that the modelling unit has checked. That is, if the modelling unit was not checked or validated then this meeting would have succeeded/failed without the need for that modelling unit. The simulation is performed in 2 stages: First is to assess the overall positive impact of the use of modelling units. This compares the meeting success rates with and without modelling units. Second is to assess the impact of each modelling unit individually to check whether or not it should be considered in the mix of units.

The calendar simulation uses a conflict identification process as without using any modelling unit, logical errors could be identified; however, real-time errors (i.e. errors

relating to time constraints e.g. delays) were not possible to detect, as at this stage the initial simulation did not cater for real-time updates. To illustrate this (logical versus real-time errors), Table 5.5 below, lists all attendee (agent) “1” meetings. When sorted by start and end time, the simulation is able to identify the unreachable meeting “6” ending at 14:41, which is considered unreachable as the next meeting “2” starts at 14:00, that is, 41 minutes before the end of the first meeting; hence attendee/agent “1” cannot attend meeting “2” (BankstownMeeting), as his train arrives Bankstown at 2:41 PM which is 41 minutes after the meeting start time.

Table 5.5: Attendee “1” meetings

MeetingID	LocationName	Subjectname	StartTime	EndTime	AttendeeID
5	Sutherland	TrainSuthUOW	9:39	10:49	1
1	North Wollongong (UOW)	WeeklyCatchup	11:00	12:00	1
6	North Wollongong (UOW)	TrainUOWBank	12:08	14:41	1
2	Bankstown	BankstownMeeting	14:00	14:30	1
7	Bankstown	TrainBankSuth	14:32	15:14	1
3	Sutherland	SutherlandMeeting	16:00	17:00	1
8	Sutherland	TrainSuthUsyd	17:02	17:40	1
4	Redfern (USYD)	USYDMeeting	18:00	19:30	1

The above test was that the previous meeting ending time (14:41) exceeds the start time of the next meeting (14:00). This example does not consider the meeting criticality or priority, that is, which meeting is more important to attend. If the “BankstownMeeting” is more important than the “WeeklyCatchup”, then the attendee/agent would miss the “WeeklyCatchup” to catch the train from UOW to Bankstown “TrainUOWBank”. In this case, the unreachable meeting would be meeting ID “1” and not meeting ID “2”. Attendees (agents) might also take the train to Bankstown “TrainUOWBank” and arrive late for the BankstownMeeting, causing delay to all future meetings, which may or may not succeed. However, this is considered as part of the meeting rescheduling process.

Agent “1” had 8 meetings in total. S/he managed to attend 7, while only 1 was unreachable. The simulation simple testing could not identify other failed meetings like meeting “7” (in Table 5.5). As Agent “1” arrives Bankstown at 14:41, which is 9 minutes after the train leaves. This was further enhanced and was identified on all further tests. The next phase of this simulation testing is to add real-time modelling units and detect real-time errors in addition to the previous logical errors. This simulation increased the

number of meetings to 102 meetings with a single meeting for each agent except meeting “101”, which is a general lecture with 98 attendees. There was a high level of meeting success, 99 out of 102, as each agent had a single meeting on the day. There was not much conflict, interaction or unforeseen events that could affect the agent’s attendance; which was advanced further by creating dense agent calendars; that is, more meetings per day for each agent.

### **5.6.1 Individual Modelling Units’ Simulation**

The next stage of validation is to get an indication of the success rate of each modelling unit. This is the number of meetings that succeed as a direct result of using this specific modelling unit. This simulation logs the last checked modelling unit that caused the meeting to succeed. Meetings that have failed after checking the RT modelling units are then tested if an extra modelling unit is needed, as the researcher manually double-checks the modelling unit values to validate them and ensure there was no other way to make this meeting succeed. Although the number of direct successful modelling units is critical, some modelling units with low success rate should still be present in the proposed process as some are considered supplementary to other highly successful modelling units. That is, modelling units that directly contribute to the meeting success; for example, sample time (warning level) is important to be proactive in identifying a task/meeting that is unlikely to succeed. Even if it doesn’t directly ensure a meeting success, it indirectly ensures the meeting doesn’t fail by proactively identifying those that are likely to fail. The simulation starts with simple events of non-dense calendars and it then escalates to include denser calendars with complex events, as follows:

- 1- This simulation creates a new event “Event 7” to change the general lecture’s time, which affects all 98 attendees. When this simulation changed the time from 14:15-15:00 to 18:15 -19:00 this simulation had 74 successful attendees leaving 24 unable to attend. However, they were not reported as failed, as they did report their absence in advance or took other alternative actions. All 24 non-attendees were not time aware, hence they failed to reschedule or attend the lecture.



Table 5.6: General lecture non-attendees

<b>ID</b>	<b>MeetingID</b>	<b>AttendeeID</b>	<b>MU ID</b>	<b>RN</b>
104	101	5	91	0
108	101	9	95	0
112	101	13	99	0
116	101	17	103	0
120	101	21	107	0
127	101	28	115	0
131	101	32	119	0
135	101	36	123	0
139	101	40	127	0
143	101	44	131	0
147	101	48	135	0
151	101	52	139	0
155	101	56	143	0
159	101	60	147	0
163	101	64	151	0
167	101	68	155	0
168	101	69	156	0
172	101	73	160	0
176	101	77	164	0
180	101	81	168	0
185	101	86	172	0
189	101	90	176	0
193	101	94	180	0
197	101	98	184	0

This rescheduling test was simple, as each agent had only one meeting, so there were no conflicts when rescheduling. Non-RT agents were unaware of the reschedule hence they failed to attend the meeting. However, as agents get busy (have more meetings per day), the success rate is expected to decrease. However, RT agents will be notified earlier, based on the sample/warning times, so they would have enough time to reschedule and re-plan other meetings depending on the meeting criticality and priority, or have alternative solutions, so that for a lecture they can arrange for someone attending to record the lecture. They are also able to notify the instructor/tutor (RTD) explaining their situation, hence would not have failed to attend as they were not absent without reason; based on the assumption that the proper rules, regulations and procedures were followed in the notification process and the university rules allows absence with a valid reason. As the number of meetings increases the complexity of rescheduling increases, needing some form of Artificial Intelligence “AI” to reschedule the meetings based on the proposed

modelling units. Table 5.7 below illustrates each modelling units' success rate when each agent had only 1 meeting to attend.

Table 5.7: Results when each agent had 1 meeting only

Meeting Date	Outcome	Success Rate
9/27/2013	Meeting Successful After IVF	30
9/27/2013	Meeting Successful as TS early or on time	54
9/23/2013	Meeting Successful After updating VD	50
9/23/2013	Meeting Successful Early	26
9/18/2013	Meeting Successful	80
9/5/2013	Meeting Successful	99
9/5/2013	Meeting Unreachable	3
9/4/2013	Meeting Successful	7
9/4/2013	Meeting Unreachable	1
9/3/2013	Meeting Unreachable	2

1- More meetings were added to each agent making it harder to reschedule individual meetings, due to their dense calendars. Table 5.8 below illustrates the success rate as a result of an event affecting (rescheduling) meetings 1 and 101 only, when each agent had more than one meeting to attend.

Table 5.8: Adding more meetings to each agent results

Meeting Date	Outcome	Success Rate
9/27/2013	Meeting Successful After IVF	60
9/27/2013	Meeting Successful as TS early or on time	113
9/23/2013	Meeting Successful After updating VD	50
9/23/2013	Meeting Successful Early	26
9/18/2013	Meeting Successful	80
9/5/2013	Meeting Successful	99
9/5/2013	Meeting Unreachable	3
9/4/2013	Meeting Successful	7
9/4/2013	Meeting Unreachable	1
9/3/2013	Meeting Unreachable	2

The success rate was considerably good due to the IVF and TS modelling units, which successfully enabled rescheduling 30 and 59 meetings respectively as per Table 5.9 below.

Table 5.9: IVF and TS effect on the meetings success rate

Meeting Date	Outcome	Success Rate
9/27/2013	Meeting Successful After IVF	30
9/27/2013	Meeting Successful as TS early or on time	59

- 2- To examine how a large number of events going wrong simultaneously impact the contribution of individual modelling unit(s) to a meeting success, 99 simultaneous events affecting 99 meetings (2-100) were simulated. This uncovered that the more important contribution of IVF, TS and REM modelling units to the success rate as, per below Table 5.10.

Table 5.10: Results of 99 events on meetings 2-100

Meeting Date	Outcome	Success Rate
9/30/2013	Meeting Successful After IVF	2
9/30/2013	Meeting Successful as REM more than ED	3
9/30/2013	Meeting Successful as TS early or on time	9

This section has validated each single modelling unit; however, the proposed modelling units are not intended to be used individually or independently of each other. As such, the next section validates a sequential modelling units set.

### 5.6.2 Sequential Modelling Units' Simulation

This simulation creates sequential modelling units' values with a variance of 5 above and below the modelling unit threshold's. That is, if the modelling unit checked a value of  $X > 5$ , then the simulation would have  $x = 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 (\pm 5)$  from the threshold (5 in this example). This simulation then validates the success rate of each modelling unit independently. That is, validated individual modelling units illustrating each one's success rate. However, since the sample data is  $\pm 5$  then one would assume that all modelling units would have the same success rate.

This simulation then added more modelling units (RTO, RTD and Comp) then ran the events on the sequential process, sequentially checking the modelling units rather than a single modelling unit. Table 5.11 below represents the results conducted on meetings scheduled for 2 days with 223 events (141 meetings reschedules, 41 transports events and 41 agent's cancellations).

Table 5.11: Success rate for 223 events

Meeting Date	Outcome	Success Rate
10/15/2013	Meeting Successful After IVF	1
10/15/2013	Meeting Successful as less than Min Time	22
10/15/2013	Meeting Successful as REM more than ED	25
10/15/2013	Meeting Successful as TS early or on time	139
10/15/2013	Non Real-time task	11
10/14/2013	Meeting Successful After IVF	3

Meeting Date	Outcome	Success Rate
10/14/2013	Meeting Successful as less than Min Time	130
10/14/2013	Meeting Successful as REM more than ED	129
10/14/2013	Meeting Successful as TS early or on time	813
10/14/2013	Non Real-time task	57

This section has validated the sequence of modelling units set, which could be biased. As such, the next section has validated randomly chosen modelling units' sets and sequences.

### 5.6.3 Random Modelling Units' Simulation

This simulation commenced more simulations on 3 event groups where each group consists of 1 events, 10 events and 100 events respectively; hence this test consisted of 12 (4\*3) simulations. The test validated how the modelling units' sets would resolve the meeting conflicts with random events and random modelling units' values; unlike the previous test which had sequential values. The test runs for only the following three agents that were randomly chosen from a pool of 179 agents:

Table 5.12: The 3 randomly chosen agents

Person Id	Person Name	Job Title
96	Michael	Webmaster
136	Ramy	System Administrator
147	Marianne	Manager

Between the 3 randomly chosen agents, there were 24 meetings to be held between them, out of the 101 total meetings (Table 5.13). Each meeting is given a unique identification number (ID) which is for the database to use to identify each meeting. As for the attendees, they can identify the meetings they are invited to by the person Name and by the subject name, Start and End time:

Table 5.13: Meetings held by the 3 randomly chosen agents

Subject name	Start Time	End Time	Person Name	ID	Subject name	Start Time	End Time	Person Name	ID
meeting 29	16:39	16:59	Marianne	29	meeting 88	3:45	4:00	Michael	88
meeting 22	17:08	17:30	Marianne	22	meeting 21	11:30	12:00	Michael	21
TrainUOWB ank	18:20	18:50	Marianne	6	meeting 98	12:30	12:45	Michael	98
meeting 42	19:00	19:30	Marianne	42	meeting 33	14:00	14:30	Michael	33
meeting 99	19:00	19:15	Marianne	99	meeting 41	15:00	15:15	Michael	41
meeting 52	20:30	20:40	Marianne	52	WeeklyCat chup	15:00	16:00	Michael	1
meeting 91	21:15	22:00	Marianne	91	meeting 50	16:30	16:45	Michael	50
meeting 81	22:30	23:00	Marianne	81	meeting 61	17:00	17:15	Michael	61
meeting 82	23:45	0:00	Marianne	82	meeting 71	20:00	20:15	Michael	71
meeting 78	23:00	0:00	Ramy	78	meeting 85	20:00	20:20	Michael	85
meeting 36	0:00	0:15	Marianne	36	meeting 40	20:02	20:40	Michael	40

Subject name	Start Time	End Time	Person Name	ID	Subject name	Start Time	End Time	Person Name	ID
Bankstown Meeting	1:00	12:00	Marianne	2	meeting 27	20:30	21:00	Michael	27
meeting 100	10:15	10:30	Marianne	100	meeting 70	21:15	21:30	Michael	70
meeting 92	11:00	11:15	Marianne	92	meeting 51	22:00	22:15	Michael	51
meeting 13	11:39	12:49	Marianne	13	General Lecture	22:15	23:00	Michael	101
meeting 98	12:30	12:45	Marianne	98	meeting 81	22:30	23:00	Michael	81
meeting 72	13:30	14:00	Marianne	72	meeting 9	14:05	15:05	Ramy	9
USYD Meeting	13:55	14:40	Marianne	4	meeting 30	16:00	16:30	Ramy	30
meeting 65	19:30	20:00	Ramy	65	meeting 23	17:15	17:30	Ramy	23
meeting 27	20:30	21:00	Ramy	27	meeting 14	17:18	18:41	Ramy	14
meeting 68	20:45	21:00	Ramy	68	meeting 26	17:30	17:45	Ramy	26
meeting 20	21:00	21:15	Ramy	20	TrainUOW Bank	18:20	18:50	Ramy	6
meeting 83	22:45	23:00	Ramy	83	meeting 46	18:45	19:15	Ramy	46
meeting 94	19:15	19:30	Ramy	94					

Table 5.14: The randomly chosen events

Events ID's	Not Attending attendees	1 Event	1 Not attending attendees
645	521	9	701
23	580		
16	677		
170	821		
677	228		
683	522		
149	42		
655	897		
65	825		
585	20		

Table 5.15: Total successful and unreachable meetings

ID	Meeting ID	Person ID	Constraint ID
127	101	28	115

Table 5.16: The 10 random not attending agents

ID	Meeting ID	Person ID	Constraint ID
111	101	12	98
1039	40	112	1035
1045	46	106	1041
550	51	113	546
890	91	97	886
608	9	171	604
849	50	56	845



This simulation illustrated that the number of conflicts could be above the number of meetings as the number of meetings only counts the meetings for the 3 agents while the conflicts checks all conflicting meetings involving all agents and not only the 3 sampled agents. This simulation preferred checking all agent conflicts due to the interdependency between meetings and agents. For example, if sampled agent “A” has a meeting with agent “X”, which is not part of the 3 sampled agents, rescheduling this meeting (between agent “A” and “X”) can cause conflicts on the schedule of agent “X” which would only be included if all conflicting meetings were checked and not only for the sampled 3 agents.

#### 5.6.3.1 Random Simulation Tests

This simulation conducted the following 12 simulation tests, catering from simple to complex scenarios with 5, 12, 18 and 23 modelling units to validate which Process (modelling unit(s) combination) is preferred, and if there is a degree of dependency between scenario complexity and number of modelling units. This simulation attempts to reschedule 3 randomly drawn agents from a pool of 179. These three agents had 48 meetings between them out of the 101 total meetings attended by the 179 agents. The events were also randomly chosen from a pool of 754 events, simulating different delays, location changes and proposing new meeting start and end times. These tests can be categorised into the following four categories:

T1: Random choosing an event (environment change) that results in rescheduling a number of meetings

T2: Rescheduling a random meeting under the below 3 strict rules

- A. Minimum number of agents: this checks that the number of meeting agents exceeds the required minimum.
- B. Specific attendees (agents) that must attend; this represents a critical person to a meeting.
- C. Meeting dependencies: this checks that the new proposed meeting time slot is not after the end time of a dependent meeting.

T3: Rescheduling a meeting with the 3 Rules and having 1 agent not attending

T4: Rescheduling a meeting with the 3 Rules and having 1 agent not attending and randomly choosing an event that results in rescheduling a number of meetings.

These four categories were chosen to illustrate the effect of agent (attendee) versus event (environment) effect on the process outcomes. then another dimension was added to apply each of the above categories however having 1 ,10 and 100 events changed and not only 1 event. This dimension simulates the effect of meeting and events complexity and density. The combination of the four test categories and the randomly chosen 1,10,100 events resulted in the below 12 tests:

1. T1\_1 (1 Event): This tested having 1 random event to reschedule a single meeting.
2. T1\_10 (10 Events): This tested having 10 random events to reschedule a number of meetings; the number of meetings was not pre-identified as the events were randomly chosen.
3. T1\_100 (100 Events): This tested having 100 random events to reschedule a number of meetings; the number of meetings was not pre-identified as the events were randomly chosen.
4. T2\_1 (3Const 1 Event): This tested having 3 rules (constants) and then attempting to reschedule a single meeting based on test 1 (T1) random event.
5. T2\_10 (3Const 10 Events): This tested having 3 rules then rescheduling a number of meetings based on the same 10 random events that was chosen in T1.
6. T2\_100 (3Const 100 Events): This tested having 3 rules then rescheduling a number of meetings based on the same 100 random events that was chosen in T1.
7. T3\_1 (3Const 1 Event Not Attending): This tested having 3 rules then having one random agent not attending a meeting.
8. T3\_10 (3Const 10 Events Not Attending): This tested having 3 rules then having 10 random agents not attending their meetings.
9. T3\_100 (3Const 100 Events Not Attending): This tested having 3 rules then having 100 random agents not attending their meetings.
10. T4\_1 (3Const 1 Event Both Events): This tested having 3 rules then having the same random events of T1 and the same T3 random agents not attending their meetings.
11. T4\_10 (3Const 10 Events Both Events): This tested having 3 rules then having the same random events of T1 and the same 10 random T3 agents not attending their meetings.



12. T4\_100 (3Const 100 Events Both Events): This tested having 3 rules then having the same random events of T1 and the same 100 random T3 agents not attending their meetings.

### 5.6.3.2 Random Simulation Results

All test results proved the modelling units' high success rates. Most unsuccessful meetings were rescheduled and the number of conflicting meetings were reduced. The 10 events setup resulted in 10 conflicting meetings out of 48 meetings. That is 10 meetings that agents could no longer attend, however applying the modelling processes reschedules these meetings allowing agents to attend their meetings. The number of successfully rescheduled conflicting meetings illustrates the modelling framework success rate i.e. if all conflicting meetings were rescheduled and all agents could attend their meetings then the success rate is 100%. The rescheduling in some cases affected how agents had scheduled their other tasks. For example, the 100 events resulted in 25 conflicting meetings however to reschedule them, there were 27 meetings and tasks to be rescheduled due to meeting/tasks dependencies and priorities. The modelling process always had a positive success rate except in only 3 cases; where the modelling process had a negative success rate i.e. increased the conflicting meetings rather than reduced them, as highlighted in red in Table 5.18.

Table 5.18: Summarised simulation results

	T1_1	T1_10	T1_100	T2_1	T2_10	T2_100	T3_1	T3_10	T3_100	T4_1	T4_10	T4_100
<b>Conflicts</b>	15	19	75	16	22	126	15	16	13	15	20	100
<b>Logic 1</b>	3	0	28	9	11	25	14	5	6	6	11	8
<b>Logic 2</b>	0	0	23	8	10	23	11	3	9	3	12	11
<b>FC 1</b>	0	0	18	0	0	7	9	5	4	6	2	38
<b>FC 2</b>	0	5	27	11	13	22	10	13	9	8	7	12
<b>FC 3</b>	0	7	26	11	17	32	18	5	13	4	14	11
<b>Set 1</b>	0	5	25	9	24	63	11	8	7	12	16	24
<b>Set 2</b>	0	8	27	16	24	20	11	11	9	11	12	8
<b>Set 3</b>	4	14	13	8	15	28	10	12	10	10	7	10
<b>Min</b>	0	0	13	0	0	7	9	3	4	3	2	8

The success rate was mostly 100% for simple events e.g. T1\_10 and T1\_10. That is, all meetings conflicts were resolved, however, as the number of conflicts and events increased the success rate was less than 100% yet was high enough to consider the modelling process validation successful, as illustrated in Table 5.19 and Figures 5.12-16.

Table 5.19: The Meeting success rate in resolving calendar conflicts

Success Rate	T1_1	T1_10	T1_100	T2_1	T2_10	T2_100	T3_1	T3_10	T3_100	T4_1	T4_10	T4_100
Logic 1	80%	100%	62.7%	43.8%	50%	80.2%	6.7%	68.8%	53.8%	60%	45%	92%
Logic 2	100%	100%	69.3%	50%	54.5%	81.7%	26.7%	81.3%	30.8%	80%	40%	89%
FC 1	100%	100%	76%	100%	100%	94.4%	40%	68.8%	69.2%	60%	90%	62%
FC 2	100%	73.7%	64%	31.3%	40.9%	82.5%	33.3%	18.8%	30.8%	46.7%	65%	88%
FC 3	100%	63.2%	65.3%	31.3%	22.7%	74.6%	-20%	68.8%	0%	73.3%	30%	89%
Set 1	100%	73.7%	66.7%	43.8%	-9.1%	50%	26.7%	50%	46.2%	20%	20%	76%
Set 2	100%	57.9%	64%	0%	-9.1%	84.1%	26.7%	31.3%	30.8%	26.7%	40%	92%
Set 3	73.3%	26.3%	82.7%	50%	31.8%	77.8%	33.3%	25%	23.1%	33.3%	65%	90%



Figure 5.12-16 Individual success rate test results for T1-T4 and all test comparison

The higher the success rate the better the modelling process would be in resolving conflicting meetings. The 18 modelling units' process (FC1) seems to be the modelling

process with the highest success rate in all tests, followed by Logic 2. However, when it came to the hardest test where 100 events and both meeting reschedules and agents were not attending, logic 1 and “Set 2” had the best success rate with only 8 out of the 100 conflicting meetings remaining (92% success rate). There was no real “best” or “worst” process, as success rates changed between all sets with none having a straight highest or lowest success rate in all tests. There were only 2 frameworks (Set 3 and Set 1) that held the lowest success rates.

### **5.7 Conclusion**

The simulation proved that adding the real-time modelling units’ process has actually improved the robustness of the simulation and its scheduling. The user (agent) was notified when the agent was running late for meetings, giving enough time to reschedule meeting(s) or choose faster travelling method(s) to arrive on time. The 5 and 12 modelling units’ processes were not considered sufficient; For example, the slack time was compared to the delay to attending a meeting. But this comparison in certain cases didn’t accurately indicate whether an agent is late or not. A more accurate indication can be obtained by considering the EAV (Execution Accrued Value). Adding EAV to the slack time and comparing it to the delay was more accurate in identifying if the attendee (agent) will be late to the meeting or unable to attend it. Other useful modelling units like MMR (Maximum Miss Ratio), VD (Validity Duration) and MT (Minimum Time) allows the agent to better understand delays and their attendance status.

Using the 18 modelling units’ process (FC1) was preferred to using the 23 modelling units’ process as discussed in section 5.3, where the 5 modelling units (Retry, Real-time Order, Remaining Time, Maximum Output Jitter and Composite) were considered duplicate. Hence The 18 modelling units’ process (FC1) was considered sufficient and enabled the meeting attendees (agents) to accurately identify delays. In general, using the modelling process enabled the researcher and developers to create a more robust simulation, in terms of having monitoring agents and applying the real-time modelling process on communication and agents’ response times.

However, this simulation had a number of limitation. For example, it was carried on a controlled environment, which might not accurately represent the real world. Hence the next chapter validates the 18 modelling units’ process (FC1), identified by this simulation as having the highest success rate, in the real world uncontrolled environment. This is achieved by developing an iPhone calendar application utilizing the process to

assist users in identifying and rescheduling their meetings as events arise; these events can be delays, meeting cancellation.

Another limitation, was agent delays were simulated as notifications, that is, the simulation triggers a delay by sending a notification that the user is running late by X minutes with expected arrival time. This delay would then be considered by the process logic and an appropriate action is taken to, for example, reschedule a meeting if possible, while for short delays, less than 5 minutes, a notification would be sent to the meeting organiser. Hence this simulation did not consider other uncontrolled actions that real life agent would take e.g. moving the meeting to a closer location. This is only achieved by the use of mobility, which was fully utilised in the iPhone application utilising the built-in GPS system, identifying the agent location, speed, travel direction. as discussed in the next chapter.

However, using physical mobiles in the real world, as represented in the next chapter, illustrated new challenges and limitations e.g. when agents could not be contacted that did not mean the agent would not attend the meeting as mobile batteries could have run out or the mobile would be out of coverage. In MAS domains, this would have to be considered, as when remote computers running tasks fail then all tasks running on that remote computer would fail, unless it had a self-healing or redundancy mechanism. This should be considered and identified when designing the application.

## CHAPTER 6

### RT MODELLING FRAMEWORK IN AN IPHONE APPLICATION

Chapter 5 developed eight candidate processes (second component of framework). The candidates were validated using a meeting scheduler simulation. This identified a highly successful process with 18 modelling units' process (FC1). This process is now coupled with the 18 modelling units to constitute the framework. This completed phase 3 of the research plan of the thesis and readies the framework for validation as one complete artefact. This chapter represents phase 4 of this thesis which further validates and refines both the set of modelling units and the process to apply them as a single framework. In this phase, two validity threats will be further mitigated: the first will be the dependence on the application domain, and the second will be the usability, of the framework, by different software developers. Finally, the usability and enhanced effectiveness of the RT MAS developed will be illustrated.


The chapter is structured as follows: Section 6.1 introduces the application background. Section 6.2 presents the application requirements and system goals. Section 6.3 describes the integration issues faced by the developer. Section 6.4 presents the application and its interface. Section 6.5 presents the application testing and validation. Section 6.6 presents the application results. Section 6.7 presents the threats to this application. Finally, Section 6.8 concludes the chapter highlighting the future planned work.

#### 6.1 Introduction

An iPhone is a smartphone manufactured by Apple Inc. The iPhone has gained popularity, and a significant portion of the smartphone market share, since its first release in 2007. Apple sold over 500,000 iPhones on the first weekend and over 42 million in the following 30 months, making it one of the bestselling mobile phones ever launched (Laugesen and Yuan 2010). On July 27, 2016 Tim Cook, current CEO of Apple Inc., announced at an employee meeting in Cupertino that Apple recently sold the billionth iPhone (Apple 2016). Its popularity and significant market share are mainly due to its ability to browse the internet and the number of applications that have been developed to integrate with it, as its browser is based on personal computer standards and not rewritten for a mobile device (West and Mace 2010). This has made it preferable to traditional mobile phones as their browsers were limited in regards to internet and applications

usage. Over the years this has changed, and a number of competing smartphone manufacturers have gained a significant portion of the market share and have enhanced their applications and browsers, for example, Samsung, Nokia, HTC and BlackBerry, that are mainly supported by operating systems from major software houses like Microsoft Windows, Google Android.

The iPhone has evolved to replace a computer in many instances, including meetings planning, emails, watching movies, gaming and simple internet browsing. With an increasing number of applications developed specifically for iPhone, it has also gained ground in other areas like GPS navigation, internet banking, education, medical and scheduling. (Wallace et al. 2012; Ling and Sundsoy 2009; Shih et al. 2010; Faliagka et al. 2011; Mayer et al. 2010). The initial iPhone did not allow 3<sup>rd</sup> party application development; however, with the opening of the App Store in July 2008, that changed and user-developed software has been allowed since then (Kim et al 2016).

Indeed, this chapter discusses an application named Pcal, developed for the iPhone mobile with a time clock icon  (in the form of the author's initials "AA"). This application is developed to validate the framework that has been synthesized in chapters 4 and 5 of the thesis. The application described in this chapter extends the existing iPhone calendar to include the modelling units. Some modelling units are mapped to existing fields. For example, *Deadline* unit is mapped to a meeting start time; others, such as *slack time* and *REM* are calculated values, hence no need for them in the application interface. Still others, like priority, have been added to the application interface. An initial version of the application was developed for Windows mobile phones but it was abandoned in favour of the iPhone platform. There was a clear chance of acceptance once the application is developed for an iPhone device, which is more widely used within the authors' network and peer groups.

An option to develop the application for an Android-supported phone was also considered; however, learning and using Objective-C development tools for iPhone development proved to be easier and more fruitful for the author (rather than learning and using Java development tools for Android development). The main challenges faced when developing for Android (and also Windows phones) is the calendar integration capabilities and APIs. Collecting enough data usage from users' mobile devices is important for the purpose of this research and this is better facilitated with iPhone platform. Windows and Android phone calendar integration are not well-documented and

supported, in comparison to the iPhone integration documentation and support. The same can be said about Global Positioning System (GPS) and location support. GPS is also difficult to integrate with the Windows calendar to import and interact with users' meetings and schedules.

The GPS integration is crucial in the application. A mobile smartphone device is indeed chosen in this validation as it has a built-in GPS system and can integrate with the user's existing calendar. GPS is a radio-based navigation system that provides location and timing services to anyone with a GPS receiver; this service was first developed by the US Air Force, then made available to civilians in 1996 (Whipple et al. 2009). The user does not have to re-enter their meetings and schedules because the application imports all calendar meetings into its database. The current calendar application has the meeting location, and the iPhone has a built-in GPS system; however, it does not notify mobile users of their delay, based on their speed and location. The introduced Pcal application enables this functionality, notifying users of their delay and expected arrival time based on their current location, speed and meeting schedule. However, to limit the use of the iPhone battery and cellular data, which is preferred and recommended for any iPhone user and developer, the application notifies only when the location services are enabled and the application is in use. This is a part of the requirements and goals as detailed in the next section.

## **6.2 Application Requirements and System Goals**

The proposed calendar scheduler requirements are generally similar to those of the scheduler simulation discussed in Chapter 5. However, there are a number of system goals and requirements specific to the iPhone application, which shaped the application design, development and interface. They include the following:

- 1- The application should be supported on various iPhone models and devices - iPhone 4, 5, 6.
- 2- The application must integrate with the existing user's calendar, which includes the iPhone calendar and any another other application calendar imported into the iPhone calendar from Outlook, Google.
- 3- The application events must be reflected in the iPhone calendar so that if the user creates a new meeting in the application it must show in the iPhone calendar. Similarly, any changes made to a meeting using the application must be reflected in the iPhone calendar.

- 4- The application must not consume too much data, potentially incurring heavy costs for the application user.
- 5- The application must not drain the mobile battery; this limited the negotiating and agent communication.
- 6- This application is not intended for business use, as the location could not identify meeting rooms within the same building. Hence, it targets meetings in different addresses that require travel from one location to another.

An assumption is made that users always try to attend meetings in their calendar. The initial analysis suggested including a checkbox identifying if a meeting has been actually attended or not. However, this raises questions about the need to audit if someone attended a meeting, based on the assumption that if a user accepts a meeting request then they would have attended it unless it was cancelled. Meetings only appear on users' calendars if they have accepted an invitation to a meeting or they are the meeting organisers themselves. Another assumption, that had to be taken into consideration, is to differentiate between a user checking a past meeting for information versus being late for a meeting and checking it to act on it i.e. contact the meeting organizer or reschedule it. By a user checking a past meeting, this thesis means a user opening a meeting calendar event after its scheduled date/time. This is considered in the form of a time constraint, where opening a meeting 10 minutes after its start time, can be considered as a user checking a late meeting. while opening it 10 hours after its start time, can be considered as a user checking the meeting for information. This research also takes into consideration that a meeting (conference) scheduled over a number of days could be opened 10 hours after its start date/time as some attendees might not attend the first day, which is usually the opening ceremony and registration.

The Pcal application integrating with the iPhone calendar, replicates all its functionality and interface to a degree that the user is familiar with. This leads to one of the main issue faced in developing the Pcal application; that is, how to integrate with the existing iPhone calendar and extend its capabilities yet not copy the existing calendar interface; existing copyright laws prohibit copying the application interface yet the application needed an interface similar to what the user was familiar with. The main purpose of the application is not the interface or the current calendar; it is the added functionality and capabilities which enables this research to gather enough information to analyse and validate the framework.



To overcome these limitations, a SQL database that has fields for existing calendar and the proposed RT modelling units is created. Hence, for any calendar operation, the application applies the change to both the new database as well as the iPhone calendar. For example, when a user changes a meeting time, the application updates the calendar meeting and the meeting database record. To link meetings in both the iPhone calendar and the Pcal application database, the application relies on the existing iPhone calendar primary key (Unique identifier). Hence, when importing the existing meetings, the Pcal application copies them with their primary keys, while when creating a new meeting the application first creates it in the iPhone calendar then retrieves its newly-created primary key to be used in the database. The Pcal application extends the calendar by creating a new SQL database to record all events and meetings with their respective RT modelling units. This database was composed of a single table “Meetings” which had the following 29 fields (Columns) representing the modelling units and mapping existing current iPhone calendar fields when possible. However, some fields are not relevant to the research, such as All Day event, calendar, show as, private and URL which are not mapped, as discussed and illustrated in Figure 6-1 and Table 6.1:

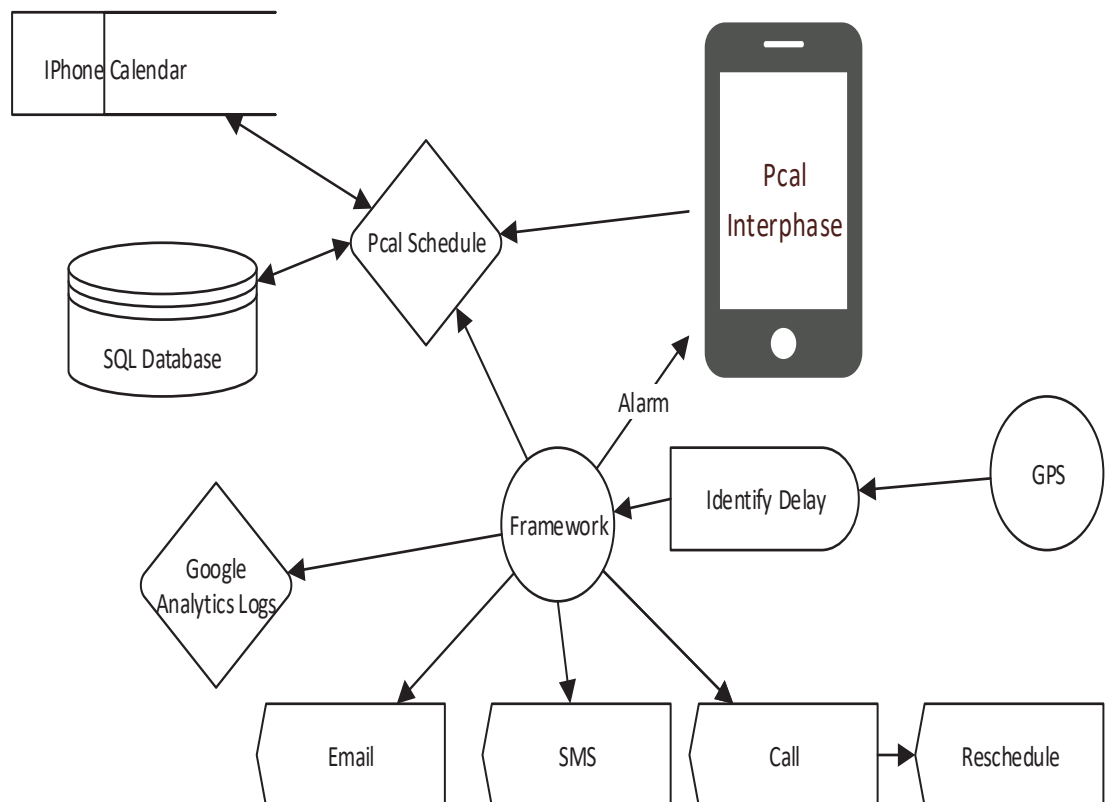


Figure 6-1 Pcal application diagram

Table 6.1: Database fields mapped to existing iPhone calendar fields

Current iPhone Calendar	New Database fields	Unmapped Fields
Title	Title	Hard
Location	Location	MMR
Starts	Dline	TS
	StartDate	REM
Ends	EndDate	CP
Repeat	R	VD
	PER	Real
	UID	P
Invitees	RTD	C
Alerts	W	RTO
Notes	Notes	Slack time
All-Day Event*		EAV
Calendar	Calendar	IVF
Show as	Show as	MOJ
Private	Private	COM
URL	URL	ED
		MT

The above fields are mapped to existing iPhone calendar fields, where the mapped fields appear next to each other. The “All Day event” field is not mapped as the application identifies it using its start and end times which are “00:00” and “23:59” respectively. Most fields depend on the users input except the following fields which are calculated:

- 1- REM: This is the remaining time to the meeting which is calculated based on the difference between the meeting start time and the current time.
- 2- TS: This represents task status and is based on the time left to the meeting versus the expected travel time. If the user had 30 minutes left to the meeting, while the expected travel time is 60 minutes, then the application identifies that they are running 30 minutes late and notifies the user.
- 3- Slack time: This is calculated as the difference between the end time of a meeting and the start time of the next meeting.

This section discussed the system goals and requirements. The next section illustrates how the framework is used to develop the Pcal application, as well as to facilitate the meeting rescheduling.

### 6.3 Integrating RT Requirements within the iPhone Calendar

The RT modelling units and process (the two components of the framework) are used in 2 parts of the Pcal application:

- 1) In the Pcal application development analysis phase the modelling units were used to identify potential time constraints and bottlenecks, and then the framework was used to work around such identifications. The framework enhances the Pcal application analysis and development by early identification of bottlenecks and exceptions. The potential exceptions identification includes time exceptions, such as when tasks take longer than expected.

Some identified time constraints that are implemented include when to send data, and how often the application should send the data so as to preserve the battery life and cellular data consumption, while having as much real-time data as possible. In this exercise, the analyst prioritises data sources where critical information needs to be sent imminently e.g. a user location to identify if the user is running late. While other low priority data sources, such as analysis and data usage are stored locally and sent in batches every 20 minutes. It also helps the developer consider alternative options and tasks to be executed in case the original code fails, for example, if speed is not identified due to the user not moving, a minimum speed of 1KM/hr is assumed; if no database file is found, or the device is unresponsive.

- 2) In the Pcal application development code the modelling units are used to identify user delay to attend meetings and to apply the framework to reschedule and re-plan such meetings, expected delays and/or meeting in attendance.

The framework leads to identification of tasks to help reschedule meetings, identify if the user is late or not and identify the appropriate actions to take. For example, being late for an unimportant meeting, an SMS or email to the organiser can be appropriate. For critical meetings, a phone call to better explain the delay and reschedule is more appropriate. Moreover, if the expected arrival is within the meeting time then the user only reports a delay, while if the user is expected to arrive after the meeting end then a reschedule is necessary.

This section has presented the use of the modelling units in the iPhone calendar application. The next section details the application and its interface.

## 6.4 The Pcal Application

The Pcal application interface is made up of 4 screens which connects to the database component of the application. The database has been discussed earlier, so this section will only focus on the application interface, features and functionality. The interface is developed using objective C classes and X-Code development framework which provide the tools required to build an iPhone application. The first screen, presented in Figure 6-2, represents the meetings list view, listing scheduled meetings for the next 30 days. The 30-day view is only to limit the list display; however, all future and past meetings are stored in the database unless deleted by the user. All meetings are also stored on the iPhone calendar to facilitate search and integration between both calendars.

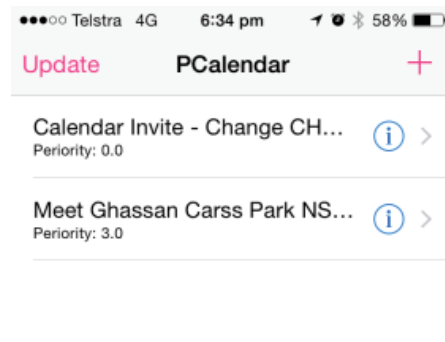


Figure 6-2 Pcal application home screen listing a month's meetings

On top of the meetings list there are 2 buttons which are an “*Update*” button and an add “+” button. The update button is used to import all changes from the iPhone calendar to the Pcal application, including any meeting updates in the iPhone calendar (deletion or addition of new meetings). Such updates are not currently automatically reflected into the application. At any time, a user can delete a meeting by swiping the to-be-deleted meeting in the meeting list to the left. This will show a delete button, as shown in Figure 6-3 below.

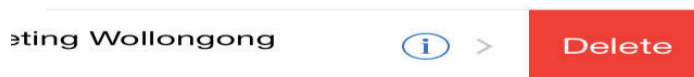


Figure 6-3 Deleting a meeting

The add “+” button is used to create a new meeting which is automatically reflected in the iPhone calendar and the Pcal application. This button opens the 2<sup>nd</sup> screen (the edit screen) presented in Figure 6-4; however, the fields do not have values in that screen. If the user edits a meeting, by pressing the “i” icon next to the meeting in the meeting list, the application will load the 2<sup>nd</sup> screen (the edit screen) with the meeting values in the

appropriate fields. A user editing the information in the 2<sup>nd</sup> screen (Figure 6-4), will see the following fields:

- 1- Meeting title: This holds the meeting title.
- 2- Priority: This is an integer from 0-100 to highlighting importance of a meeting.
- 3- Location: This marks where the meeting will take place; clicking on this button will open the 3<sup>rd</sup> screen (map search screen) of the Pcal application.
- 4- Start Date: This is the meeting start date.
- 5- End Date: This is the meeting end date.
- 6- Speed and delay information: This is a read-only label illustrating:
  - A. Speed: The users current average speed.
  - B. Remaining Distance: The distance from the user's current location to the meeting location.
  - C. Remaining time: The time left from now till the meeting start time.
  - D. You're early by or you're late by: This illustrates the users delay or early expectancy based on his current speed, remaining distance and remaining time to the meeting scheduled start date/time.
- 7- Save Button: This saves any changes the user makes to the meeting. The changes are saved to both the iPhone calendar and the Pcal application database.
- 8- Go Back Button: This takes the user back to the first screen (meetings list).

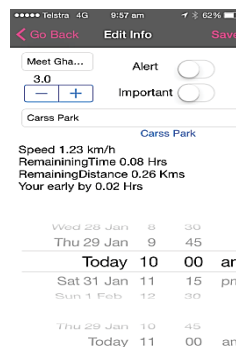


Figure 6-4 Edit meeting screen

Once the user presses on the location button a map search screen opens. The map search allows the user to enter an address, which is then validated and all search results for the entered location are listed below the search bar, as per Figure 6-5.

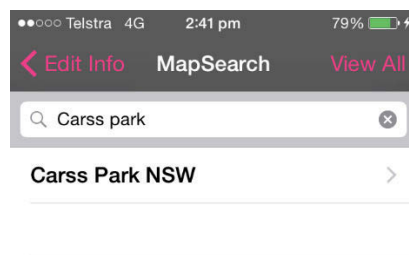


Figure 6-5 Location search screen

Once the user selects a search result, a map screen is opened displaying the users' selection on the map to confirm the location. The top left button "Map Search" takes the user back to the previous screen (Map search) if the user needs to update the location address or go back to the Pcal application main screen (meeting list) by clicking back again on the edit screen.

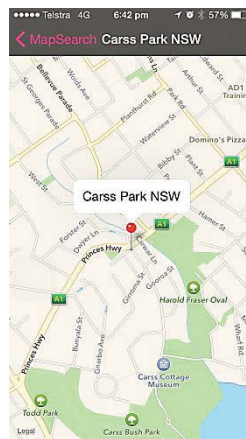


Figure 6-6 Location map display

All meetings are synchronised with other devices and calendars. Once the Pcal application updates the iPhone calendar, the relevant calendar is updated and synchronised on other devices. e.g. Microsoft Outlook calendar is updated once the iPhone calendar changes. The published application versions in the App Store are not in a sequential order, due to some versions being rejected by Apple then reviewed and enhanced for bug fixes before being accepted and published on the App Store. The current published versions are:

Table 6.2: Pcal application versions

Version	Publish Date	Notes
1.3	06 Feb 2015	Without Google Analytics
1.7	27 Feb 2015	8 Reported Event categories
2.5	07 Mar 2015	35 Reported Event categories
3.1	24 Mar 2015	8 Reported Event categories
3.11	31 Mar 2015	1 Reported Event categories
5	12 May 2015	
6	07 Oct 2015	Bug fixes

This section has presented the application interface, features and functionality. The next section discusses the tests conducted and data gathering.

### 6.5 Application Testing and Validation

Before making it available on the App Store for public download, the Pcal application was tested internally. Any iPhone user was able to download the Pcal application for free, and to validate how it was able to notify users of their delay and assist them in rescheduling the meeting and/or arriving on-time to their scheduled meetings. There were a number of conditions that led to a meeting rescheduling including, but not limited to: User delay in attending a meeting, a meeting cancelation by one of the attendees or a meeting rescheduled by one of the attendees.

The user delay was automatically identified by the user's GPS position, travel speed and remaining time to the meeting start. These variables allowed the Pcal application to identify the user status, that is, whether the user will arrive on time, early or is expected to be late for the meeting. Based on the user's expected status, the application uses the process shown in Figure 6-7 (as identified from the previous case study), to re-plan and advise the user what to do. E.g. to notify meeting attendees about the expected delay, to reschedule the meeting, or to do nothing when the expected delay is less than 5 minutes (considered an acceptable delay).

The test was mainly to validate the whole framework consisting of the 18 modelling units coupled with the process (labelled FC1 in chapter 5), in an uncontrolled real life environment. Although the results from this case study mainly focus on the framework

end results, some modelling units' results were recorded using Google Analytics as a means of validation and verification of the framework usage and validity. Although this research does not control the activities of the iPhone users who download the application from the App Store, this research obtains usage results via Google Analytics. Google Analytics is a service offered by Google that generates detailed statistics about mobile applications' usage, after including specific code within the mobile application. Google Analytics reports the total events and hits an application has had, as well as how many unique users an application had over a specific period of time, which country they came from, and more. By tracking events in the application, like pressing specific buttons or performing different actions, this setting provides insights on calendar event, users' delays and delay causes without violating any privacy laws.

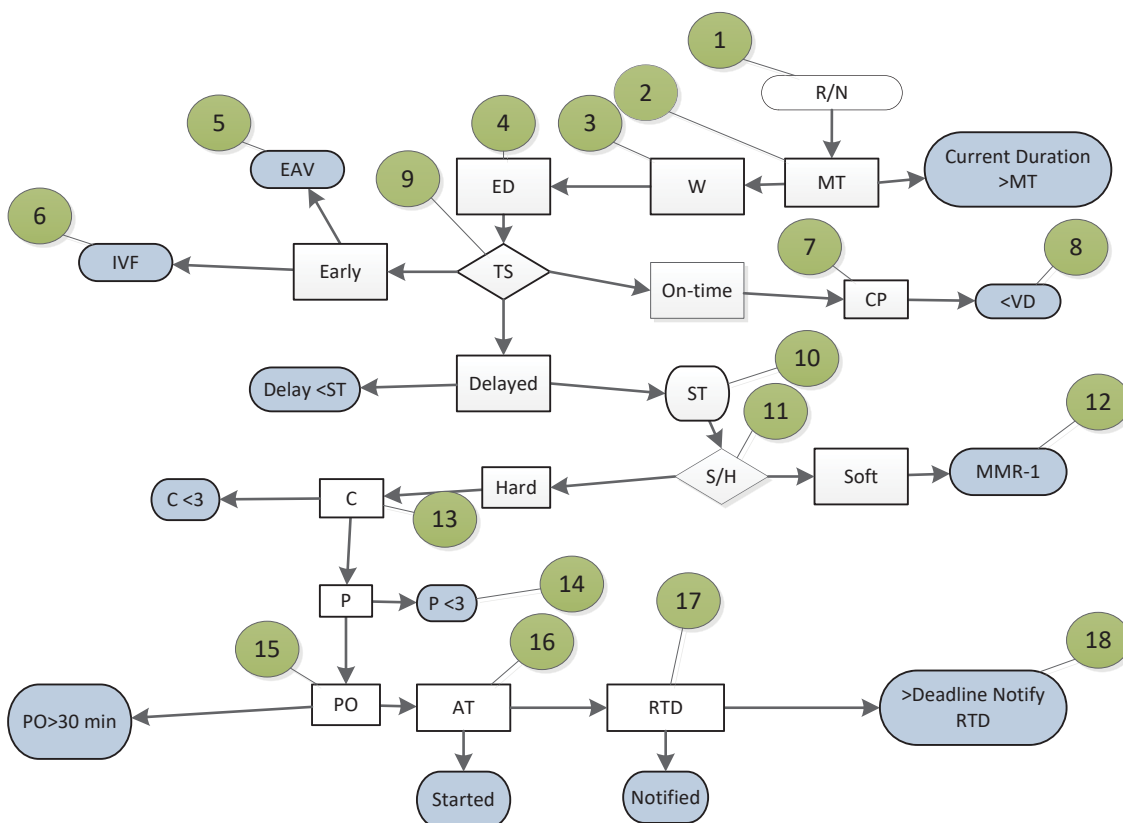


Figure 6-7 The framework showing the process and the used 18 modelling units

Each of version of the Pcal application reported user events. These were analysed and used to enhance further versions. The main events were whether the meeting was successful or not. Other supplementary events were which modelling and part of the framework enabled the meeting success. The application version 2.5 was the best performing version with 1,717 events in 35 categories within 3 weeks. This was mainly



due to the advertisement and high user engagement, as it was this thesis' first published application with the aim of gathering as much input and data as possible. The next version, 3.1, had only 399 events in 8 categories in one week, and the longest version (5) lasted 5 months, yet only recorded 388 events in 6 categories. These event results illustrate the effective role advertising and publicity play in marketing an application and user engagement. Although the Pcal application was highly successful, users were not engaged for long periods and would only check it when running late to meetings or events out of their offices, due to lack of location services within the same building; the application could not identify distance and user location when the meeting was held in the same address but in different meeting rooms, floors or offices. The Pcal application interface is also said to be “not that attractive”, but this is not the aim of the research author, who developed the application only to collect user data for analysis.

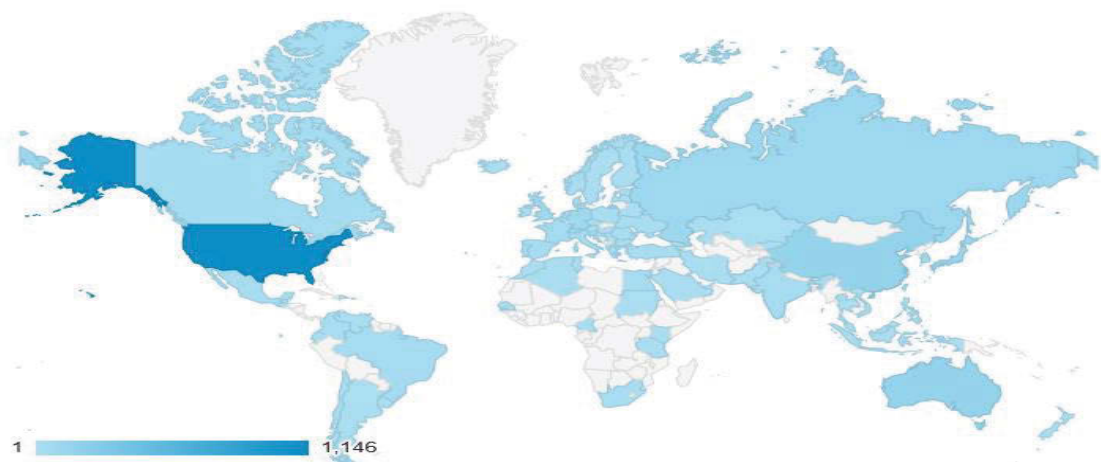


Figure 6-8 Pcal application users density per country, as per Google Analytics e.g. 1000 users from USA, i.e. the darker the blue colour the more users per country

## 6.6 Results

This research is not intended to develop a profitable commercial level application, as the application is developed only as a data source to validate the framework. The application has been approved and published on the App Store as “Pcal”, short for priority calendar, on 14/03/2015. Since then it has been downloaded by 2,890 users from 86 different countries in 6 continents as per Figure 6-8, and is still being downloaded on a daily basis. All reviews and results were in favour of the application.

An interesting item of consumer feedback is that flight calendar events deadline should be 45 min ***before*** the scheduled flight departure time, as that is when the gates close for boarding passengers. While if the event is to meet an arriving passenger, then the deadline should be 45 minutes ***after*** the schedule flight arrival time, to cater for the

time taken by the traveller to exit the airport through customs, border security. This is actually part of the core of this thesis, but is not documented properly in the application manual to keep it simple to access.

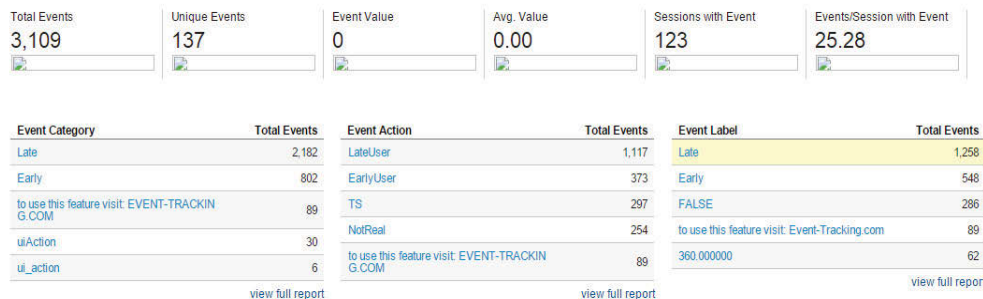


Figure 6-9 The total number of events and their categories, actions and labels

Google Analytics results show, as per Table 6.3, the majority of the 1,117 events were events relating to users being late for meetings; this is due to the fact that users tend to check late meetings more than meetings they are actually running early for, or are on time, while the application only communicates results after a user checks and opens it. Although there were reported 1,117 late events, the SendMessage, SendEmail, reschedule and PhoneCall functionality (events) were only used 10 (0%), 12(0%), 20 (1%) and 48 (2%) times respectively. These results illustrate that users prefer using the built-in iPhone functionality for making phone calls, sending emails and messages, to using the Pcal application functionality when they are late. These figures illustrate the limitation of the developed Pcal application and its inability to lock the user to only use it and not use other built-in iPhone features like making phone calls, sending emails, sending messages, rescheduling directly using the built-in calendar. Not using the logged features in the Pcal application results in some of the reported usage figures seeming to be missing, incomplete or even incorrect and misleading. Some modelling units were not illustrated due to a bug in the Pcal application logic flow. The bug has been identified and fixed. The Google Analytics new data is time-based, while this thesis compared versions based only on their publish date. However, this analysis might not be accurate, as some users might not have upgraded their Pcal application and continued to use the older version which did not report on some of the modelling units.

Table 6.3: Google events modelling units' events hits and percentage

<b>Event Action</b>	<b>Total Events</b>	<b>Total Events %</b>
LateUser	1,117	36%
EarlyUser	373	12%
TS	297	10%
NotReal	254	8%
EVENT-TRACKING.COM	89	3%
High MT	74	2%
High CP	52	2%
High VD	52	2%
High EAV	51	2%
High ED	51	2%
High MOJ	51	2%
High IVF	50	2%
PhoneCall	48	2%
REM	34	1%
CP	32	1%
High REM	32	1%
RN	32	1%
VD	31	1%
ButtonPress	25	1%
ED	29	1%
EAV	27	1%
MOJ	27	1%
MT	27	1%
IVF	26	1%
REAL	24	1%
Low W	23	1%
Reschedule	20	1%
Low CP	19	1%
Low ED	19	1%
Low MT	19	1%
Low VD	19	1%
Low EAV	18	1%
Low MOJ	18	1%
Low IVF	16	1%
SendEmail	12	0%
Low REM	11	0%
SendMessage	10	0%
<b>Total</b>	<b>3,109</b>	<b>100%</b>

Table 6.4: Detailed Google Analytics per Pcal application events

	V1.7	V2.5	V3.1	V3.11	V5
LateUser	629	297	124		167
TS		120			
NotReal		77	120		134
EarlyUser	296	74	33		
High MT		52		4	
High CP		52			
High VD		51			
High EAV		51			
High ED		51			
High MOJ		50			
High IVF		34			
REM		32			
CP		32			
High REM		32			
RN		31		32	
VD		29			
ED		28			
PhoneCall	19	27		18	1
EAV		27			
MOJ		27			
MT		26			
IVF		24			
REAL		23		4	
Low W		19		4	
Low CP		19			
Low ED		19			
Low MT		19			
Low VD		18			
Low EAV		18			
Low MOJ		16			
Low IVF		11			
Low REM		4			
SendEmail	7	4			1
SendMessage	4	2			1
Reschedule	20	2			
EVENT-TRACKING				7	83
ButtonPress	25				

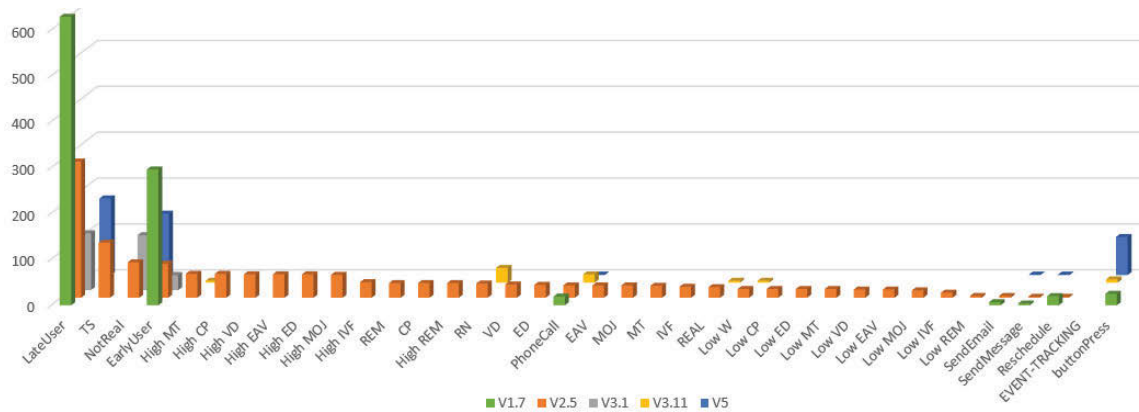


Figure 6-10 Detailed Google Analytics per Pcal application event

Table 6.5: Summarized Google Analytics per Pcal application event

	V1.7	V2.5	V3.1	V3.11	V5
CP		103			
EAV		96			
ED		98			
EVENT-TRACKING		0		7	83
IVF		45			
MOJ		93			
REM		68			
VD		98			
W		19		4	
buttonPress	25				
EarlyUser	296	74	33		
IVF		24			
LateUser	629	297	124		167
MT		97		4	
RN		131	120	36	134
PhoneCall	19	27		18	1
Reschedule	20	2			
SendEmail	7	4			1
SendMessage	4	2			1
TS		120			

In general, users seem to be using the Pcal application to identify if they are running late and check their estimated arrival time, yet they prefer taking matters into their own hands or using other means to reschedule, re-plan or contact the relative parties and negotiate their meeting changes; for example, version 5 of the Pcal application had 167 late users yet only 2 sent a message, 1 made a phone call and one sent an email. Similar results were noted for version 3.1 which had 124 late users with only 18 making phone calls. However, 32 were identified as non-real meetings as they were not time-dependent and could be done at any convenient time; mostly these are actual calendar scheduled tasks with no attendees or specific start time, such as all-day events or tasks with titles like “write a shopping list”, “start reading a new book”, “watch a movie”, “finish writing documentation”. Four meeting delays exceeded the minimum time, and 4 were sampled early enough to speed up and arrive on time.

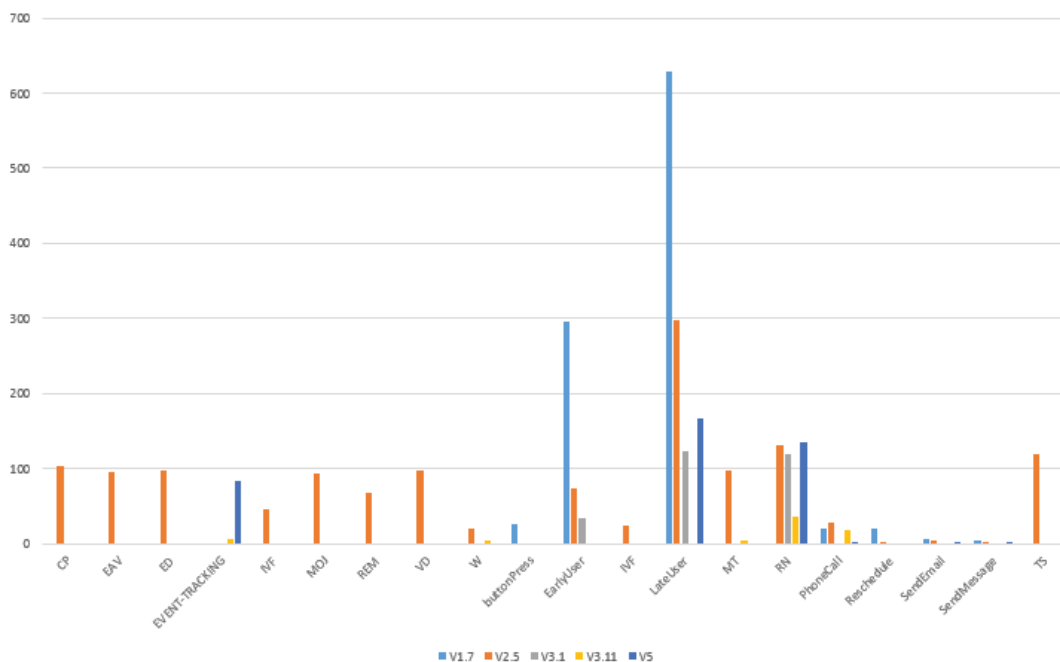


Figure 6-11 Summarised Google Analytics per Pcal application event

Version 2.5, which is the most-used Pcal application version with most of the functionalities and buttons clicked, provided more insight details on user activity and delay. In total, there were 1,717 reported events, 321 of them were late users and 77 were actually early for their meetings. While 297 were users only checked their status and 120 were non real-time meetings. Although the Pcal application did not log all events due to a reported bug, the logged events are consistent with previous research findings in Chapters 5, which validate the importance and effectiveness of the framework to ensure the meeting success. The use of the framework enabled the system to resolve meetings conflicts. The preferred modelling units were: task status (identifying if the user is late or early for the meeting); alternate action (rescheduling meeting, alternate transport method.); and to notify the meeting organizer (RTD). There were always “make call”, “send email” or “send message” events in all the Pcal application events, even those with very limited reported results (e.g. V5 and V3.11). High priority meetings didn’t seem to be checked that regularly, as it seems people were more aware of such meetings.

## 6.7 Threats

Apple’s IOS 8.1 introduced a new feature to the iPhone calendar named “Travel Time”, which allows users to choose estimated travel duration from a pre-set list. However, this option is used to alert by a specific time before the travel time or before the actual meeting

time as per Figure 6-12 and 13. This is different from this thesis’ framework as the “travel time” feature is based on the user’s alarm choice; that is, it’s like setting the iPhone alarm to go off at a specific time before the scheduled travel start time. The framework here identifies delay before and during the task or “travel time”, and also proposes alternative travel and rescheduling options enabling the user to attend the meeting on time.

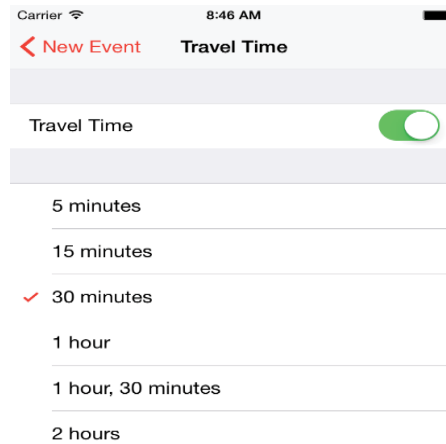


Figure 6-12 Apple’s IOS 8.1 Travel Time Alert settings option

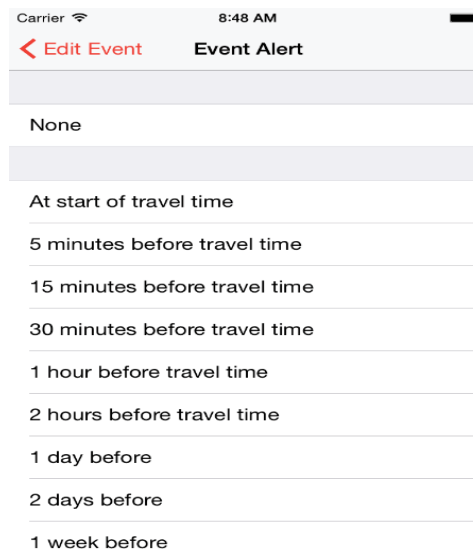


Figure 6-13 Apple’s IOS 8.1 Travel Time option

## 6.8 Summary

The framework developed in chapters 4 and 5 was used in the Pcal application. Starting by designing and analysing the user requirements, on to implementing the Pcal application and utilising this research findings and framework to extend the existing iPhone calendar functionality. This is the initial version of the Pcal application and further enhancements can be made, such as alerting the user without the need to open the Pcal application while still maintaining battery life and cellular data usage. Other required enhancements include automatically importing all updates from the iPhone calendar to

the Pcal application without the need to press the “Update” button, and enhancing the user interface and performance. These have been marked for future work but currently they have little bearing on the validation of the framework as an analysis tool.

The use of the framework to identify real-time requirements ensured that nothing is missed or overlooked. Hence, the development of this Pcal application validates the framework as an analysis tool and shows how it enhances the system analysis process by early identifying and avoiding a number of exceptions what would have otherwise been considered as bugs or errors in the application. However, the application is not guaranteed to be bug free as there will always be a number of bugs that are not identified (as in any other software development exercise). However, there is a clear case that the framework actually enhances the product quality. The quality of analysis is also affected by the guidelines and process, so by enriching the guidelines and process, the software analysis in general is being enriched.



## CHAPTER 7

### CONCLUSION

This chapter concludes the thesis by summarising its main achievements and contributions. The chapter is structured as follows: Section 1 summarizes the research; Section 2 highlights the thesis' main contributions from both a general software development perspective and a specific multi agent systems perspective; Section 3 outlines the limitations of the research done and the possibilities for future and Section 4 concludes the thesis with final remarks.

#### 7.1 Thesis Summary

This thesis facilitates the use of multi-agent systems for real-time applications by developing a framework, composed of a set of modelling units and a process to guide their use. The framework enables the software developer to define concepts from which elements from the set of modelling units be instantiated from which a model (a design) can be constructed. The design can then be hand-coded or used as the input to a model based (or model-driven) information systems development. The framework ensures that failed tasks can be distinguished from those that are likely to succeed even if they are a bit late, e.g. ensuring that the latter ones are provided with more resources, or delaying dependant tasks to prevent a cascade of failed tasks.

The first component of the framework, the set of modelling units consists of two subsets: The first subset provides the knowledge to identify the success or failure of the task to meet its real-time constraints. The second subset provides the possible set of available behavioural actions. Both subsets of the modelling units were identified based on a rigorous literature review as advocated in Kitchenham et al (2009) and also used by other agent modelling researchers e.g. Kardas (2013). The output of the review was analysed into the operational set of modelling units to be used during the requirements analysis phase. To facilitate their use, the modelling units were represented using symbolic icons which were first used in the i\* SR model and applied to a call centre case

study requirements analysis. This first case study investigated the requirements analysis of a real-time multi-agent system in a call system to match end-customers to agents representing relationship managers according to specific criteria and characteristics e.g. skills, age, sex, culture, language proficiency, experience and product knowledge. This case study illustrated that identifying real-time requirements in the early analysis phase leads to a better load distribution among agents and ensures that agents can meet their real-time requirements and helps design a more efficient, reliable, robust and redundant system.

To enable the systematic use of the modelling units in the analysis development phase, the framework has an analysis process. The process essentially interleaves the use of the modelling units into typical agent oriented requirements analysis. To develop the process, multiple processes were created and simulated. Candidate processes were formulated based on an analysis of the relationships between the various modelling units. A best-of-breed process was then chosen and refined. The threat against domain dependence of the framework was mitigated by using a different domain in the second case study, a calendar scheduling. The calendar was made real time aware. Choosing the calendar scheduling domain ensures the generalisability of the framework as this domain can be easily mapped to other domains. For instance, any project context e.g. construction, software development, supply chain or planning and other, all has a scheduling time component, which can be presented using their start/end times, location and dependencies. Tasks in such domains often have time constraints and need their scheduled time updated throughout their execution. The function of the calendar was simulated in various scenarios representing different events with various real-time constraints. The calendar simulation was made time aware by monitoring delays of arrival to a meeting. It also executed rescheduling actions in case of cancellations or other unforeseen environmental changes. Users were notified of delays, and new schedules were generated based on actual expected arrival times, when possible. In simulation runs, various processes with the modelling units were developed to reschedule meetings. In total eight processes with different modelling unit combinations were developed and validated using a simulation case study that confirmed that a process with 18 modelling units (framework) had the highest success rate i.e. succeeded in rescheduling unreachable and/or conflicting meetings. In choosing and validating the process, the set of modelling units were further validated and refined.

The overall framework, the modelling units and the process combined, was further validated using an iPhone application publicly available in the Apple Store. This was done to further validate the framework in an uncontrolled environment. The research presented in the thesis followed a design science methodology. Using the design science research approach, the research was organized into 4 phases: problem identification, solution proposal, synthesis literature review, and case study validation. The modelling units and process (the framework) have been identified, developed, refined, modelled, formalized and validated using both a synthesis literature review and the case studies from the two different domains (*call management* and *a calendar meeting scheduling*).

## 7.2 Thesis Contributions

Software modelling processes typically involve a number of phases including analysis, specification, design, implementation, and testing. Each phase would create its own model (system representation) and bring the software system closer to realisation. Each phase represents the software system from a different abstraction point of view and collectively they represent the system. This research takes the view that the earlier we model real-time requirements in the software development life cycle, the more reliable and robust the resultant system will be. Furthermore, the more likely it is that an appropriate balance between competing time requirements will be achieved.

The requirements analysis phase in developing multi-agent systems captures system goals and refines these into agent goals and respective roles descriptions. Later in the design phase, goals and roles are further analysed to identify agent tasks and agent classes that are closer to the system implementation (DeLoach 2001). The main contribution of the thesis is a framework consisting of a set of identified real-time modelling units and their deployment process in the requirements analysis phase of developing a MAS. This framework enhances the analysis tasks of requirements engineering for MAS. The modelling framework can be viewed as a real-time metamodel to support requirements analysis of a MAS, tightly coupled with a process to identify the real-time requirements of a multi-agent system during the analysis phase. With this view in mind, the thesis contributes to bridging the gap between modelling the MAS requirements and the realisation of the real-time software components required to operationalise real-time constraints of MAS tasks. The research presented facilitates

identifying a sufficient set of activities that can be used by software modellers to identify when a task has failed to meet its real-time constraints.

Current agent methods do not easily enable analysts to make these distinctions i.e. identify problematic tasks and set “an alternative course of action”, thus creating a more robust system overall, let alone identify real-time tasks in the midst of requirements analysis and elicitation. This research fills this gap and in essence it promotes further context awareness of agents as advocated in (Barbosa et al 2012). To represent the salient features of the environment and the required agent interactions that are relevant to identify real-time constraints on agent’s actions, this research focussed on providing a reliable and precise analysis process. It ensures that the system modeller captures the real-time constraints and the concomitant required agent’s behaviour. The work relied on using modelling criteria to identify the set of alternative actions to be taken once a task has been determined as having failed to meet its real-time constraints. This set of behaviour actions can range from logging an error to starting an alternate task. The need to include further support for modelling languages to support RT requirements was addressed. This thesis provides a list of constructs to assist analysts in identifying real-time tasks and specifying their relevant and critical attributes. This thesis also provides a process that interleaves the use of the constructs in a typical agent oriented system analysis phase.

### **7.3 Thesis Limitations and Future Work**

Using the framework incurs additional analysis effort on the part of the system’s developers. This added effort is clearly justified in critical applications. However, the rigour of the process identifying the RT constraints could conceivably be reduced in less critical applications. A line for possible future extension of this research is making the process part of the framework more adaptive. A more adaptive process could incorporate a cost-benefit analysis in applying the modelling units. In critical applications, for instance, the added cost can be easily justified. In less critical applications, a less thorough process could be applied.

The framework enables the software developer to define concepts from which modelling elements can be instantiated and a model (a design) can be constructed. How the design is converted to a system was not within the scope of the thesis. The design can either be hand-coded or used as the input to a model based (or model-driven) information systems development, as in MDE (model-driven engineering) or a specific flavour of

MDE like OMG's Model-Driven Architecture (MDA) (Pavón et al 2006, Rodrigues 2015 and OMG 2008). This framework differentiates between RT requirements that are modelled by the software developer during the design stage and RT related actions that are performed by the multi-agent system during run-time in order to satisfy the RT requirements. This allows for the development of a RT-aware platform-independent design, providing part of a Platform Independent Model for MDA. In this sense, in this thesis MDE support is restricted to providing RT language elements to facilitate requirements analysis. Further support for a MDE development approach is to realize a working RT-aware multi-agent system for a specific platform requires additional modelling framework for the remaining phases of the development and bridging this framework to the requirements models in this thesis work. The work presented in Hahn et al (2009) and Wautelet et al (2016) can then be used to support the remainder of MDE or RT MAS. Facilitating the linkage between the output of the framework and the input for such MDE approaches is a strong future possibility for extending this research.

Many efforts that attempt to bridge the gap between modelling requirements and generating a working system, i.e. facilitating MDE of MAS, also provide a graphical editor to enable the easy capture and mapping of models e.g. (Fuentes-Fernandez et al. 2010; Kardas et al. 2009; Gómez-Sanz et al. 2010). An extension of the effort in this thesis will also consider developing a graphical editor to ease the access and the deployment of the approach.

#### **7.4 Concluding Remarks**

The research successfully achieved its original three main goals:

1. Providing a modelling framework to facilitate identifying when a task has failed to meet its real-time requirements.
2. Synthesizing a reliable and precise analysis process to ensure that the system modeller captures the real-time requirements and the concomitant required agent's behaviour.
3. Using modelling criteria to identify the set of alternative actions to be taken once a task has been determined that it failed to meet its real-time requirements.

This thesis provided an effective framework to create more effective multi-agent systems in a real-time setting. This is very significant as many modern applications of such systems do entail real-time constraints. The framework complements the

requirements analysis phase with a model driven approach to better identify real-time tasks. Following a rigorous validation of its two components (the modelling units and the concomitant process), the full framework was effectively used to develop an iPhone application, which validated the effectiveness of the framework. The overall resultant system had improved robustness. Users were notified when they were running late for appointments, giving them enough time to reschedule meetings or choose other alternate faster traveling methods to arrive on time.

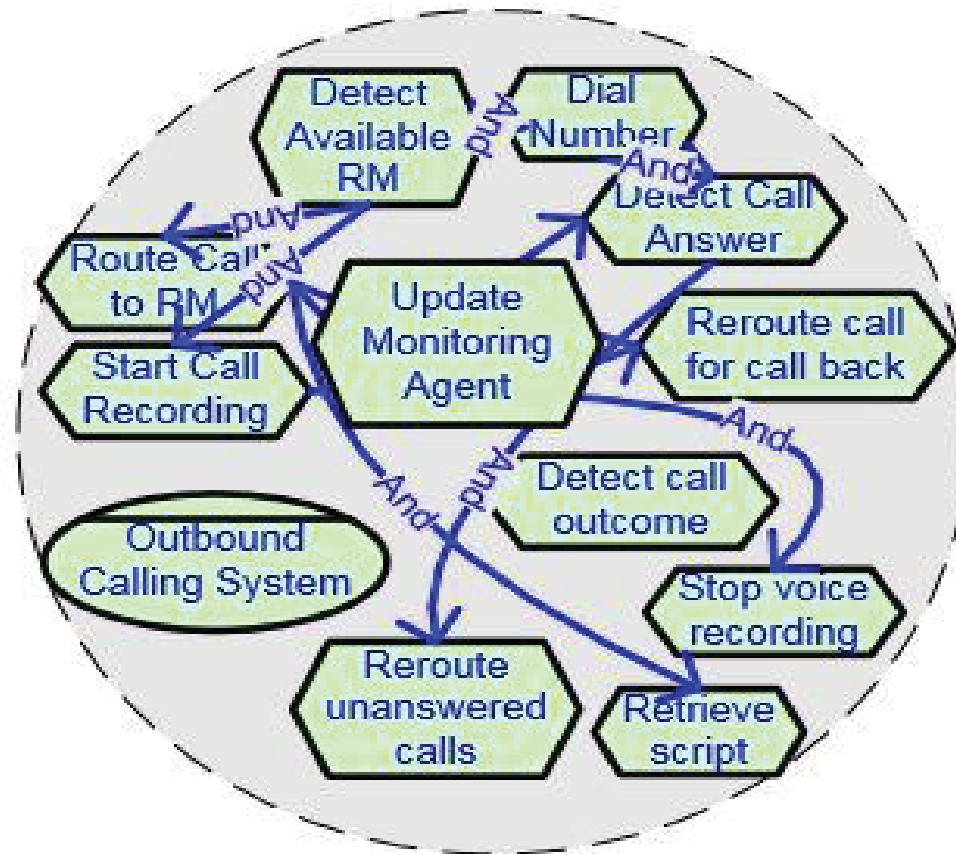
The modelling framework developed will help developers better understand the problem requirements and give them more insights as to the different real-time aspects of the problem. This will avoid future problems that might arise as a result of not meeting real-time constraints.

This thesis proposed framework can be applied in critical RT applications including applications where any response delays or faults can cost lives e.g. medical, flight autopilot and self-driving cars applications. For example, if a self-driving car detects a fault it cannot just stop the car in the middle of the street, however it should safely park the car and notify the driver whom can override such process and takeover driving at any time. A fault detection can be from an unresponsive device e.g. a lost GPS signal that does not allow the car to self-drive safely.

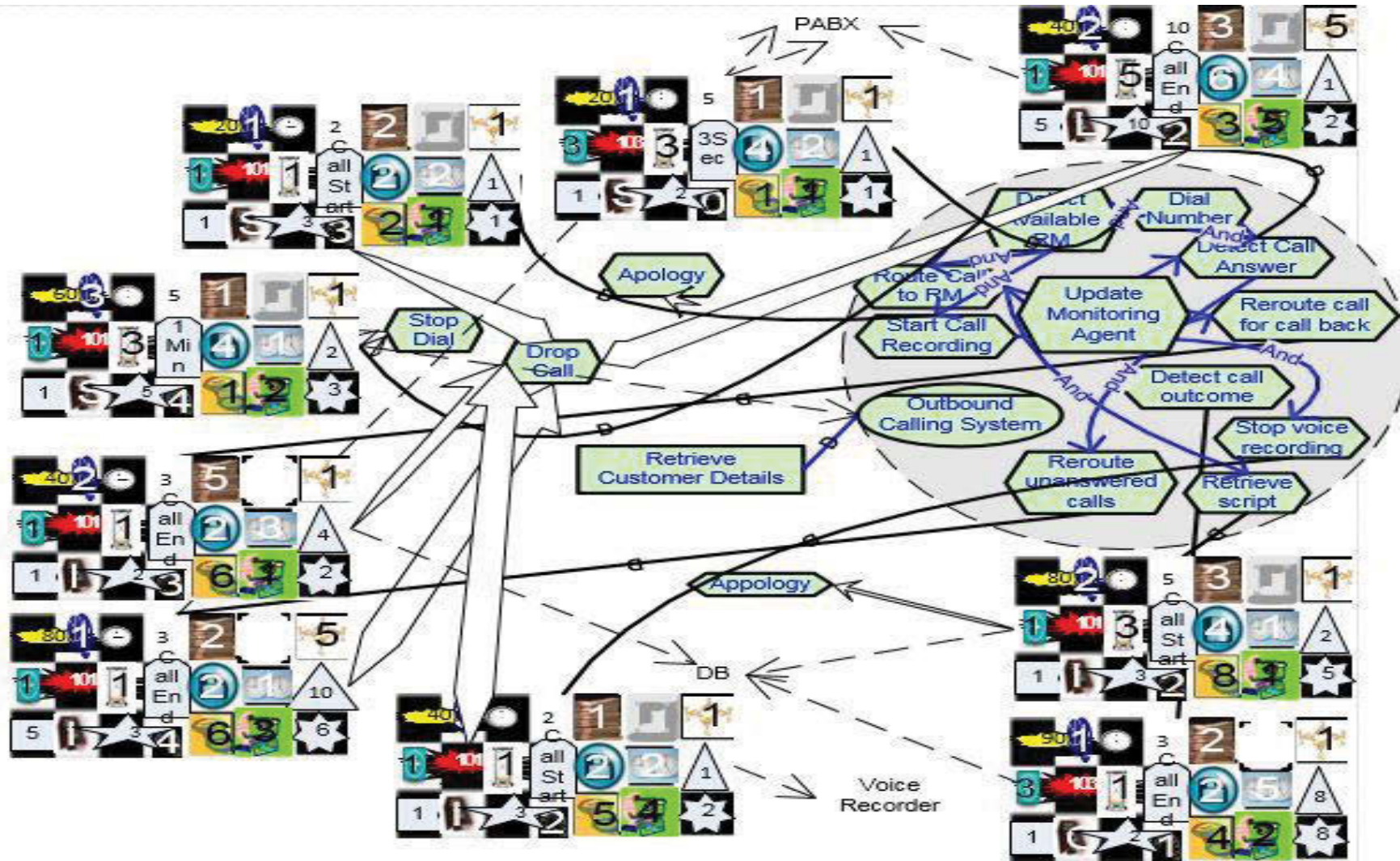
Many efforts that attempt to bridge the gap between modelling requirements and generating a working system, i.e. facilitating MDE of MAS, provide a graphical editor to enable the easy capture and mapping of models e.g. Fuentes-Fernandez et al, 2010; Kardas et al 2009; Gómez-Sanz et al 2010. An extension of the effort in this paper will also consider developing a graphical editor to ease access and deployment of the approach.

## APPENDIX A

## CALL MANAGEMENT SYSTEM SR DIAGRAMS

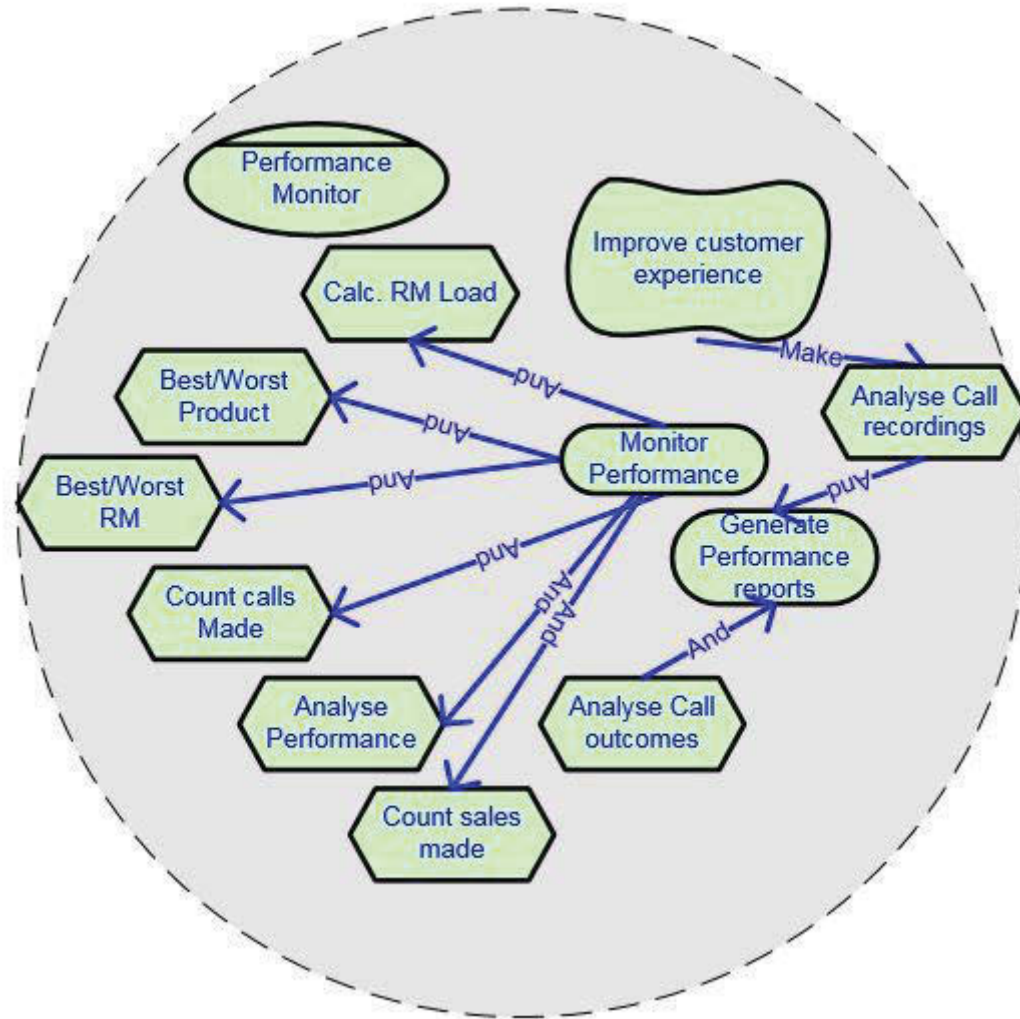


A-1 Outbound calling system SR diagram

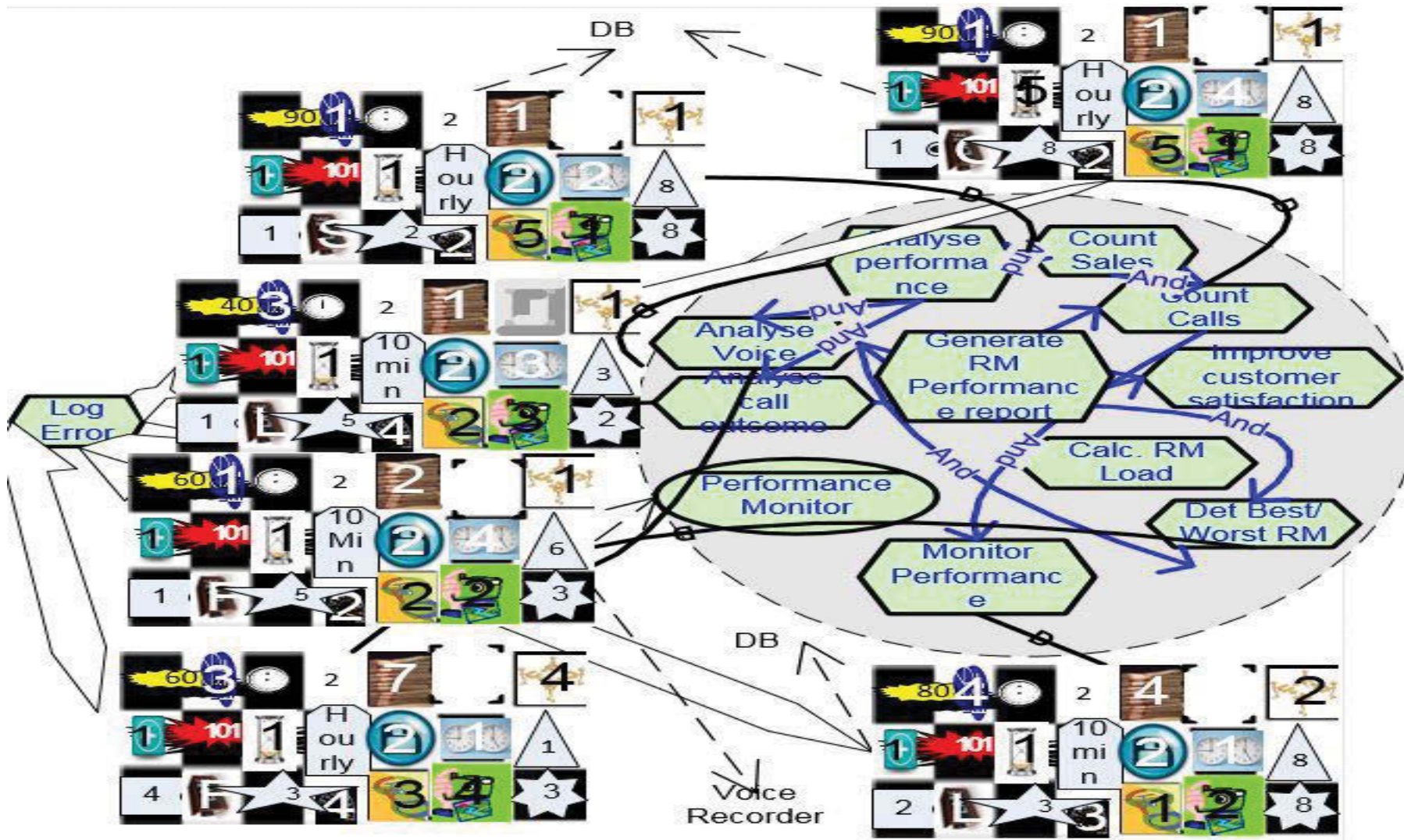


A-2 SR with modelling units

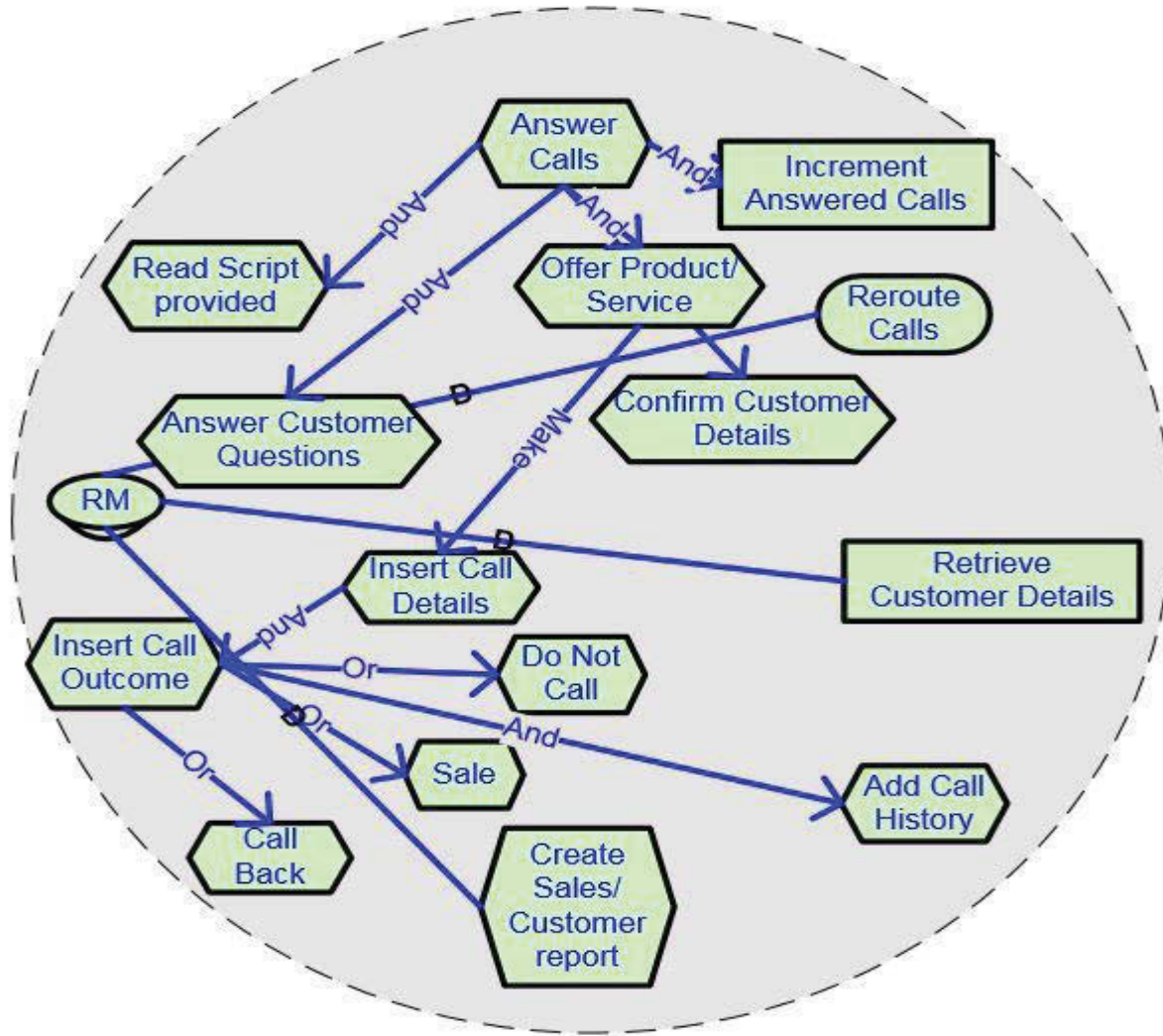




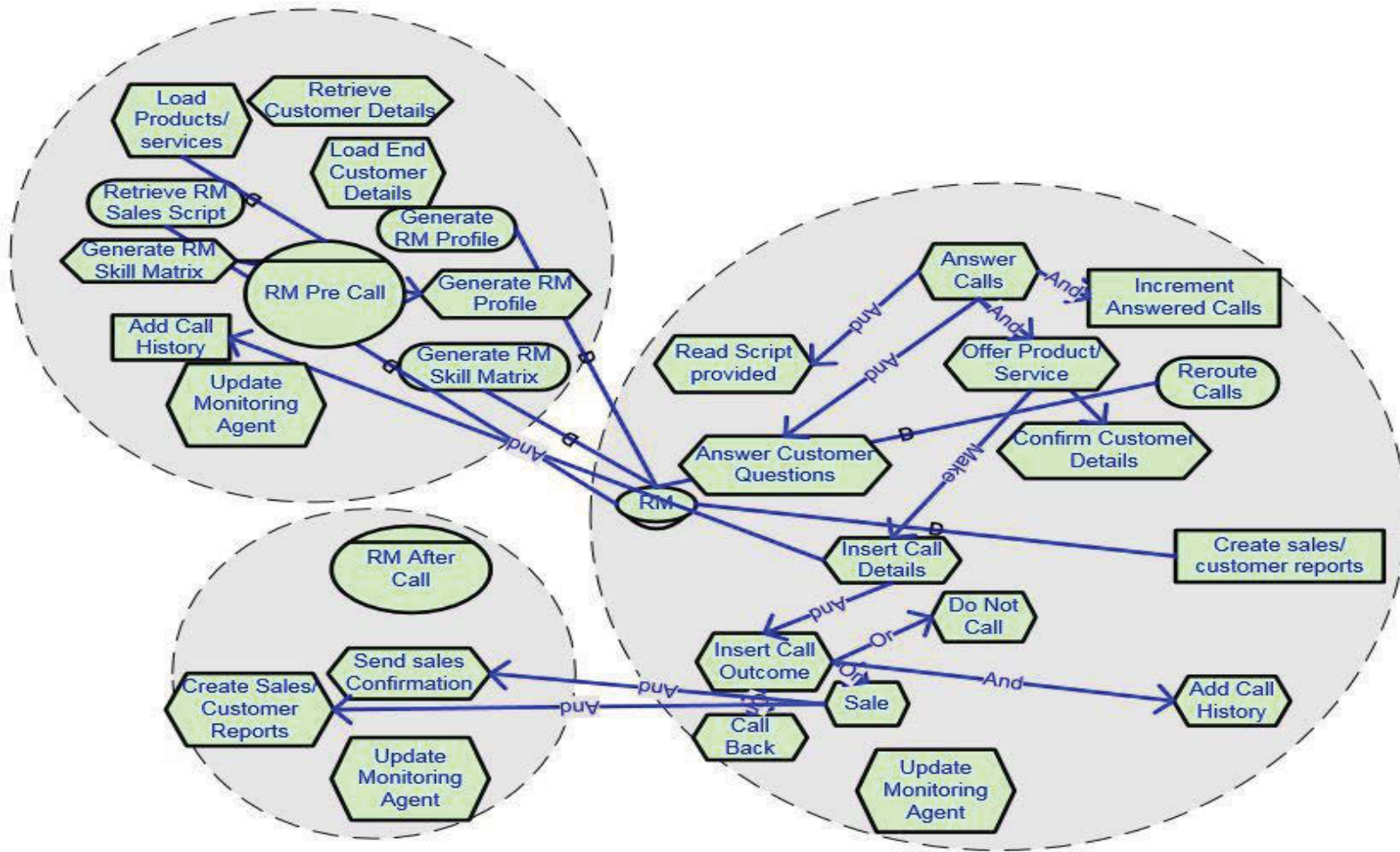
A-3 Performance monitor SR diagram



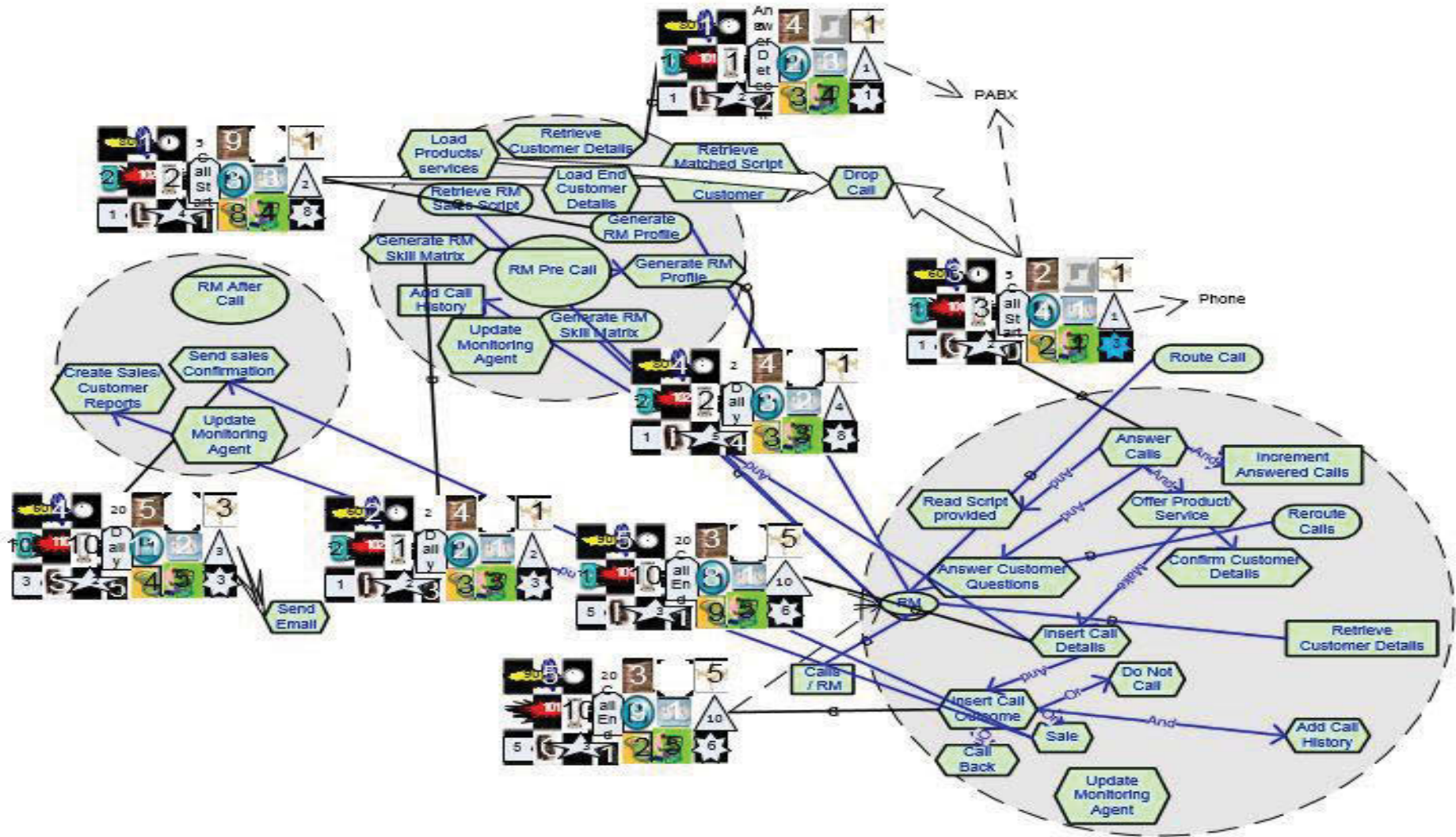
A-4 Performance monitor SR modelling units diagram



A-5 Relationship manager SR diagram



A-6 RM after split SR diagram



A-7 Relationship manager after split SR modelling units diagram

## APPENDIX B

### TABLE 4.2 SOURCES OF MODELLING UNITS

1. Wei, J., W. Hanpin and Z. Meixia (2011). Modeling MARTE Sequence Diagram with Timing Pi-Calculus. Object/Component/Service-Oriented Real-Time Distributed Computing (ISORC), 2011 14th IEEE International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing, 61-66.
2. Chantem, T., W. Xiaofeng, M. D. Lemmon and X. S. Hu (2008). Period and Deadline Selection for Schedulability in Real-Time Systems. Real-Time Systems, 2008. ECRTS '08. Euromicro Conference on Real-Time Systems.
3. Cavalieri, S. (2005). "Meeting real-time constraints in CAN." Industrial Informatics, IEEE Transactions on 1(2): 124-135.
4. Jyhjong, L. (2000). Real-time systems development: from structured analysis to object-oriented design. Real-Time Computing Systems and Applications, 2000. Proceedings. Seventh International Conference on Real-Time Computing Systems and Applications. IEEE: 486-490.
5. Kim, K. H., L. Juqiang and K. Moon-Hae (2000). Deadline handling in real-time distributed objects. Object-Oriented Real-Time Distributed Computing, 2000. (ISORC 2000) Proceedings. Third IEEE International Symposium on Object-Oriented Real-Time Distributed Computing:7-15.
6. Jennings, N. R. (2000). "On agent-based software engineering." Artificial Intelligence.117(2): 277-296.
7. Neto, A., F. Sartori, F. Piccolo, R. Vitelli, G. De Tommasi, L. Zabeo, A. Barbalace, H. Fernandes, D. F. Valcárcel and A. J. N. Batista (2009). MARTE: a Multi-Platform Real-Time Framework. Proc. of the 16th IEEE NPSS Real-Time Conference, Beijing, China.
8. Dasarathy, B. (1985). "Timing constraints of real-time systems: Constructs for expressing them, methods of validating them." Software Engineering, IEEE Transactions on (1): 80-86.
9. Occello, M., Y. Demazeau and C. Baeijs (1998). "Designing organized agents for cooperation with real time constraints." In Collective Robotics, First International Workshop, volume 1456, pages 25-37. LNCS, Springer.

10. Kirk, B., Nigro, L., & Pupo, F. (1997). Using real time constraints for modularisation. *Lecture Notes in Computer Science* (Vol. 1204, 236-251). Berlin: Springer.
11. Liu, G., A.K. Mok, and P. Konana (1998). A unified approach for specifying timing constraints and composite events in active real-time database systems. In *RTAS*. IEEE.
12. Soh, L.-K. and C. Tsatsoulis (2005), A real-time negotiation model and a multi-agent sensor network implementation. *Autonomous Agents and Multi-Agent Systems*. 11(3): p. 215-271.
13. Atkins, E. M., T. F. Abdelzaher, K. G. Shin and E. H. Durfee (2001). "Planning and resource allocation for hard real-time, fault-tolerant plan execution." *Autonomous Agents and Multi-Agent Systems* 4(1): 57-78.
14. Giunchiglia, F., J. Mylopoulos, and A. Perini (2003), The tropos software development methodology: processes, models and diagrams. *Agent-Oriented Software Engineering III*: p. 162-173.
15. Červenka, R., I. Trenčanský, M. Calisti and D. Greenwood (2005). AML: Agent modeling language toward industry-grade agent-based modeling. *Agent-Oriented Software Engineering V*, Springer: 31-46.
16. Miller, T., S. Pedell, L. Sterling and B. Lu (2011). Engaging Stakeholders with Agent-Oriented Requirements Modelling. *Agent-Oriented Software Engineering XI*. D. Weyns and M.-P. Gleizes, Springer Berlin Heidelberg. 6788: 62-78.
17. Weyns, D. (2007), A. Omicini, and J. Odell, Environment as a first class abstraction in multiagent systems. *Autonomous Agents and Multi-Agent Systems*. 14(1): 5-30.
18. Brazier, F. M. T., F. Cornelissen, C. M. Jonker and J. Treur (2000). "Compositional Specification and Reuse of a Generic Cooperative Agent Model." *International Journal of Cooperative Information Systems* 9(3): 171.
19. Flake, S. (2002) Real-time constraints with the OCL. in *Object-Oriented Real-Time Distributed Computing, 2002. (ISORC 2002)*. Proceedings. Fifth IEEE International Symposium on Object-Oriented Real-Time Distributed Computing.

20. Zhihao, J., M. Pajic, A. Connolly, S. Dixit and R. Mangharam Real-Time Heart Model for Implantable Cardiac Device Validation and Verification. Real-Time Systems (ECRTS), 2010 22nd Euromicro Conference on Real-Time Systems.
21. Legout, V., M. Jan, and L. Pautet (2015), Scheduling algorithms to reduce the static energy consumption of real-time systems. Real-Time Systems. 51(2): 153-191.
22. Liu, C. and J.H. Anderson (2012). Supporting soft real-time parallel applications on multicore processors. in Embedded and Real-Time Computing Systems and Applications (RTCSA), IEEE 18th International Conference on. IEEE.
23. Buttazzo, G.C. (2011), Hard real-time computing systems: predictable scheduling algorithms and applications. Vol. 24: Springer.
24. Axer, P., M. Sebastian, and R. Ernst. (2011) Reliability analysis for MPSoCs with mixed-critical, hard real-time constraints. in Hardware/Software Codesign and System Synthesis (CODES+ ISSS), Proceedings of the 9th International Conference on. IEEE.
25. Broster, I., G. Bernat, and A. Burns. (2002) Weakly hard real-time constraints on controller area network. IEEE.
26. DiPippo, L. C., V. Fay-Wolfe, L. Nair, E. Hodys and O. Uvarov (2001). A real-time multi-agent system architecture for e-commerce applications. Fifth International Symposium on Autonomous Decentralized Systems, Dallas, Texas, IEEE Computer Society Washington, DC, USA.
27. Silly-Chetto, A.M.a.M. (2006), Dynamic Real-time Scheduling of Firm Periodic Tasks with Hard and Soft Aperiodic Tasks Real-Time Systems. 32(1-2): 21-47.
28. Bergmans, L. and M. Aksit, (1996) Composing synchronization and real-time constraints. Journal of parallel and distributed computing. 36(1): p. 32-52.
29. Kihwal, L. and S. Lui. (2005) A dependable online testing and upgrade architecture for real-time embedded systems. 11th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications.
30. Palopoli, L., G. Buttazzo, and P. Ancilotti. (1999) A C language extension for programming real-time applications. in Real-Time Computing Systems and



- Applications, 1999. RTCSA '99. Sixth International Conference on Real-Time Computing Systems and Applications.
31. Moron, C.E. (1996) Designing a real-time recoverable action. in Real-Time Computing Systems and Applications, Proceedings of the Third International Workshop on Real-Time Computing Systems and Applications.
  32. Cha, S. K., B. D. Park, S. J. Lee, S. H. Song, J. H. Park, J. S. Lee, S. Y. Park, D. Y. Hur and G. B. Kim (1995). Object-oriented design of main-memory DBMS for real-time applications. Real-Time Computing Systems and Applications, Proceedings of the Second International Workshop on Real Time Computing Systems and Applications.
  33. Izosimov, V., P. Eles, and Z. Peng. (2010) Value-based scheduling of distributed fault-tolerant real-time systems with soft and hard timing constraints. in Embedded Systems for Real-Time Multimedia (ESTIMedia), 2010 8th IEEE Workshop on Embedded Systems for Real-Time Multimedia.
  34. Krasovec, G., N. Shankar, and P. Ward. (1996) Integration of formal verification with real-time design. in Object-Oriented Real-Time Dependable Systems, Proceedings of WORDS '96., Second Workshop on Object-Oriented Real-Time Dependable Systems.
  35. Kuster, J. and J. Stroop. (2001) Consistent design of embedded real-time systems with UML-RT. in Object-Oriented Real-Time Distributed Computing, ISORC - 2001. Proceedings. Fourth IEEE International Symposium on Object-Oriented Real-Time Distributed Computing.
  36. Laouadi, M.A., F. Mokhati, and H. Seridi-Bouchelaghem, (2010) A novel formal specification approach for real-time multi-agent system functional requirements, in Multiagent System Technologies, Springer. p. 15-27.
  37. Pěchouček, M. and V. Mařík, (2008) Industrial deployment of multi-agent technologies: review and selected case studies. *Autonomous Agents and Multi-Agent Systems*. 17(3): 397-431.
  38. Furfaro, A., L. Nigro, and F. Pupo, (2006) Modular Design of Real-Time Systems Using Hierarchical Communicating Real-time State Machines. *Real-Time Systems*. 32(1): 105-123.
  39. Bruno, O. M., R. M. Cesar, L. A. Consularo and L. D. F. Costa (2001). "Σynergos—Synergetic Vision Research." *Real-Time Systems* 21(1): 7-41.

40. Schmidt, D.C., D.L. Levine, and S. Mungee, (1998) The design of the TAO real-time object request broker. *Computer Communications*. 21(4): 294-324.
41. Hwang, S.I., C.M. Chen, and A.K. Agrawala. (1996) Scheduling an overloaded real-time system. *IEEE*.
42. Gerards, M. E., J. L. Hurink and P. K. Hölzenspies (2016). "A survey of offline algorithms for energy minimization under deadline constraints." *Journal of Scheduling*: 1-17.
43. Mahabadi, A., A. Khonsari, B. Khodabandeloo, H. Noori and A. Majidi (2014). "Critical Path-Aware Voltage Island Partitioning and Floorplanning for Hard Real-Time Embedded Systems." *Integration, the VLSI Journal*.
44. Liu, C. and J.H. Anderson. (2013) Suspension-Aware Analysis for Hard Real-Time Multiprocessor Scheduling. in *Real-Time Systems (ECRTS), 25th Euromicro Conference on*. *IEEE*.
45. Ellouze, Z., N. Louati, and R. Bouaziz. (2013) A real-time object-oriented data model and prototype implementation. in *Object/Component/Service-Oriented Real-Time Distributed Computing (ISORC), 2013 IEEE 16th International Symposium on*. *IEEE*.
46. Abdeddaïm, Y., Y. Chandarli, and D. Masson. (2013) The Optimality of PFPasap Algorithm for Fixed-Priority Energy-Harvesting Real-Time Systems. in *Real-Time Systems (ECRTS), 2013 25th Euromicro Conference on*. *IEEE*.
47. Toma, A. and J.-J. Chen (2013). Computation offloading for frame-based real-time tasks with resource reservation servers. in *Real-Time Systems (ECRTS), 2013 25th Euromicro Conference on*. *IEEE*.
48. Axer, P., S. Quinton, M. Neukirchner, R. Ernst, B. Dobel and H. Hartig (2013). Response-time analysis of parallel fork-join workloads with real-time constraints. *Real-Time Systems (ECRTS), 25th Euromicro Conference on*, *IEEE*.
49. Lemerre, M. and E. Ohayon. (2012) A Model of Parallel Deterministic Real-Time Computation. in *Real-Time Systems Symposium (RTSS), IEEE 33rd*. *IEEE*.
50. Stigge, M., P. Ekberg, N. Guan and W. Yi (2011). On the tractability of digraph-based task models. *Real-Time Systems (ECRTS), 23rd Euromicro Conference on*, *IEEE*.

51. Chen, J.-J. and S. Chakraborty. (2012) Partitioned packing and scheduling for sporadic real-time tasks in identical multiprocessor systems. in Real-Time Systems (ECRTS), 24th Euromicro Conference on. IEEE.
52. Short, M. and J. Proenza. (2013) Towards efficient probabilistic scheduling guarantees for real-time systems subject to random errors and random bursts of errors. in Real-Time Systems (ECRTS), 25th Euromicro Conference on. IEEE.
53. Hettiarachchi, P.M., N. Fisher, and L.Y. Wang. (2013) Achieving Thermal-Resiliency for Multicore Hard-Real-Time Systems. in Real-Time Systems (ECRTS), 25th Euromicro Conference on. IEEE.
54. Toma, A. and J.-J. Chen. (2013) Server resource reservations for computation offloading in real-time embedded systems. in Embedded Systems for Real-time Multimedia (ESTIMedia), IEEE 11th Symposium on. IEEE.
55. Li, J., M. Qiu, J.-W. Niu, L. T. Yang, Y. Zhu and Z. Ming (2013). "Thermal-aware task scheduling in 3D chip multiprocessor with real-time constrained workloads." ACM Transactions on Embedded Computing Systems (TECS) 12(2): 24.
56. Zhang, S., Z. Wang, M. Qiu and M. Liu (2013). "Energy-Efficient Soft Real-Time Scheduling for Parameter Estimation in WSNs." International Journal of Distributed Sensor Networks.
57. Abdeddaïm, Y. and D. Masson. (2012) Real-time scheduling of energy harvesting embedded systems with timed automata. in Embedded and Real-Time Computing Systems and Applications (RTCSA), IEEE 18th International Conference on. IEEE.
58. Kapitanova, K., S. H. Son, K. Wochul and K. Won-Tae (2011). Modeling and Analyzing Real-Time Data Streams. Object/Component/Service-Oriented Real-Time Distributed Computing (ISORC), 14th IEEE International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing.
59. Zhang, F., A. Burns, and S. Baruah. (2010) Sensitivity analysis for EDF scheduled arbitrary deadline real-time systems. IEEE.
60. Petters, S. M., M. Lawitzky, R. Heffernan and K. Elphinstone (2009). Towards real multi-criticality scheduling, IEEE.

61. Vikhorev, K., N. Alechina, and B. Logan. (2009) The ARTS Real-Time Agent Architecture. in Proceedings of Second Workshop on Languages, Methodologies and Development Tools for Multi-agent Systems.
62. Farzinvash, L. and M. Kargahi. (2009) A scheduling algorithm for execution-instant sensitive real-time systems. IEEE.
63. Yang, Z., L. Jie, and E.A. Lee. (2007) A Programming Model for Time-Synchronized Distributed Real-Time Systems. in Real-time and Embedded Technology and Applications Symposium, RTAS '07. 13th IEEE.
64. Lu, R.G.a.S., (2006) Modeling distributed real-time applications with specification PEARL Real-Time Systems. 35(3): p. 181-208.
65. Mitra, T. and P. Yu. (2005) Satisfying real-time constraints with custom instructions. IEEE.
66. Pao-Ann, H. and L. Shang-Wei. (2005) Model checking timed systems with priorities. in Embedded and Real-Time Computing Systems and Applications, Proceedings of the 11th IEEE International Conference on.
67. Hsin-Wen, W., H. Pei-Chi, C. Hsung-Pin and S. Wei-Kuan (2005). Scheduling real time information in a broadcast system with non real time information. in Proceedings of the 11th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications, 286-292.
68. Wehrmeister, M.A. and L.B. Becker, (2005) An object-oriented platform-based design process for embedded real-time systems. Proc. of the 8th IEEE Int. Symp. on Object-Oriented Real-Time Distributed Computing, IEEE Computer Society, 125-128
69. Dang Van, H. and A. Bui Vu. (2005) Model checking real-time component based systems with blackbox testing. in Proceedings of 11th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications, Washington, DC, USA, 76–79.
70. Broster, A.B.a.G.R.-N., (2005) Timing Analysis of Real-Time Communication Under Electromagnetic Interference. Real-Time Systems. 30(1-2): p. 55-81.
71. Hoang, H., et al. (2006) Computing the Minimum EDF Feasible Deadline in Periodic Systems. Proceedings of the 12th IEEE Int'l Conf. Embedded and Real-Time Computing Systems and Applications, 125-134.

72. Moon Hae, K., L. Sun-Hwa, and K. Jung-Guk. (2003) Modeling of a real-time distributed network management based on TMN and the TMO model. Proceedings of the Eighth International Workshop on Object-Oriented Real-Time Dependable Systems (WORDS 2003).
73. Isovich, D., G. Fohler, and L. Steffens. (2003) Timing constraints of MPEG-2 decoding for high quality video: misconceptions and realistic assumptions. in Real-Time Systems, Proceedings of the 15th Euromicro Conference on Real-Time Systems.
74. Gonzalez Harbour, M., J. J. Gutierrez Garcia, J. C. Palencia Gutierrez and J. M. Drake Moyano (2001). MAST: Modeling and analysis suite for real time applications. Proceedings of the 13th Euromicro Conference on Real-Time Systems, p.125.
75. Saehwa, K., C. Sukjae, and H. Seongsoo. (2000) Schedulability-aware mapping of real-time object-oriented models to multi-threaded implementations. Proceedings of the Seventh International Conference on Real-Time Computing Systems and Applications.
76. Seong Woo, K., C. Byung Jae, and K. Byung Kook. (2000) Checkpointing strategy for multiple real-time tasks. Proceedings of the Seventh International Conference on Real-Time Computing Systems and Applications.
77. Jigorea, R., S. Manolache, P. Eles and Z. Peng (2000). Modelling of real-time embedded systems in an object-oriented design environment with UML. Proceedings of the Third IEEE International Symposium on Object-Oriented Real-Time Distributed Computing (ISORC 2000).
78. Bondavalli, A. and F. Di Giandomenico (2000). A position on design, methods, and tools for object-oriented real-time computing. Proceedings of the Third IEEE International Symposium on Object-Oriented Real-Time Distributed Computing (ISORC 2000).
79. Pasaje, J.L.M., M.G. Harbour, and J.M. Drake. (2001) MAST real-time view: A graphic UML tool for modeling object-oriented real-time systems. Proceedings of the 22nd IEEE real-time systems symposium, London, UK, pp 245–256

80. Gumzej, R. and M. Colnaric. (2001) An approach to modeling and verification of real-time systems. Proceedings of the Fourth IEEE International Symposium on Object-Oriented Real-Time Distributed Computing ISORC - 2001.
81. Attoui, A., (2000) Real-Time and Multi-Agent Systems, ed. 1st: Springer.
82. Haritsa, J.R., M.J. Carey, and M. Livny. (1990) On being optimistic about real-time constraints. in PODS '90 Proceedings of the ninth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems. ACM.
83. Jahanian, F., R. Rajkumar, and S.C.V. Raju, (1994) Runtime monitoring of timing constraints in distributed real-time systems. Real-Time Systems. 7(3): p. 247-273.
84. Hou, C.J. and K.G. Shin. (1992) Allocation of periodic task modules with precedence and deadline constraints in distributed real-time systems. IEEE Transactions on Computers 46(12): 1338–1356.
85. Ramamritham, K., (1996) Where Do Time Constraints Come From? Where Do They Go? Journal of Database Management Spring. 7(2): p. 4-4.
86. Gerber, R. and S. Hong, (1995) Compiling real-time programs with timing constraint refinement and structural code motion. Software Engineering, IEEE Transactions on. 21(5): p. 389-404.
87. Han, C.C. and K.J. Lin, (1992) Scheduling real-time computations with separation constraints. Information Processing Letters. 42(2): p. 61-66.
88. Savor, T. and P. Dasiewicz (1993). A real-time extension to logic programming based on the concurrent constraint logic programming paradigm. Proceedings of First Workshop on Principles and Practice of Constraint Programming. Ottawa National Library of Canada: 279-287.
89. Campos, S., E. Clarke, W. Marrero and M. Minea (1995). Timing analysis of industrial real-time systems. Workshop on Industrial-Strength Formal Specification Techniques, IEEE.
90. Raju, S.C.V., R. Rajkumar, and F. Jahanian. (1992) Monitoring timing constraints in distributed real-time systems. Proc. Real-Time Systems Symp., 57-67, IEEE.
91. Pang, H., M. Livny, and M.J. Carey. (1992) Transition scheduling in multiclass real-time database systems. Proceedings of the Real-Time Systems Symp., 23-34, IEEE.

92. Tei-Wei, K., N. Shie-Kai, and H. Giun-Haur. (1998) Load adjustment and filtering based on process criticality. Proceedings of the Fifth International Conference on Real-Time Computing Systems and Applications.
93. Baba, M. D., H. Ekiz, A. Kutlu and E. T. Powner (1996). Toward adaptable distributed real-time computer systems. Proceedings of the Third International Workshop on Real-Time Computing Systems and Applications (RTCSA'96), Seoul, Korea (170–175).
94. Takashio, K., H. Shitomi, and M. Tokoro. (1995) Constructing distributed real-time systems with DROL real-time objects. Proceedings of the Second International Workshop on Real-Time Computing Systems and Applications.
95. Hooman, J. and O. van Roosmalen. (1997) Timed-event abstraction and timing constraints in distributed real-time programming. Proceedings of the Third International Workshop on Object-Oriented Real-Time Dependable Systems 153-170.
96. Bosch, J. and P. Molin. (1997) A model for a flexible and predictable object-oriented real-time system. Proceedings of the Third International Workshop on Object-Oriented Real-Time Dependable Systems.
97. Fraga, J., J. Farines, and O. Furtado. (1997) RTR model: an approach for dealing with real-time programming in open distributed systems. Proceedings of the Third International Workshop on Object-Oriented Real-Time Dependable Systems.
98. Chen, Y.J., D. Mosse, and S.K. Chang. (1996) An object-based model for dependable real-time distributed systems. Proceedings of WORDS '96., Second Workshop on Object-Oriented Real-Time Dependable Systems.
99. Streich, H. and M. Gergeleit. (1997) On the design of a dynamic distributed real-time environment. Proceedings of the Joint Workshop on Parallel and Distributed Real-Time Systems, 251-256.
100. Van der Stok, P.D.V. and P.T.A. Thijssen (1994). Simulation of distributed real-time transactions. Proceedings of the Second Workshop on Parallel and Distributed Real-Time Systems, 82–87.
101. Baruah, S.K. (1998) A general model for recurring real-time tasks. In Proceedings of the Real-Time Systems Symposium. Madrid, Spain: IEEE Computer Society Press, 114-122.

102. Selic, B. (1996) Modeling real-time distributed software systems. Proceedings of the 4th International Workshop on Parallel and Distributed Real-Time Systems, 11-18.
103. Gumzej, R. and S. Lu, (2007) Modeling distributed real-time applications with specification PEARL. *Real-Time Systems*. 35(3):181-208.
104. Bihari, T.E. (1993) Real-time software product development. Proceedings of the IEEE Workshop on Real-Time Applications.
105. Roop, P.S. and A. Sowmya. (1998) Hidden time model for specification and verification of embedded systems. Proceedings of the 10th Euromicro Workshop on Real-Time Systems, pages 98-105. IEEE Computer Society Press.
106. Eriksson, C., J. Gustafsson, J. Brorson and M. Gustafsson (1993). An Object-Oriented Framework for Designing Hard Real-Time Systems. Proceedings of the Fifth Euromicro Workshop on Real-Time Systems, pages 90-97. IEEE Computer Society Press.
107. Huang, J. and L. Gruenwald. (1996) Impact of timing constraints on real-time database recovery. Proceedings of the workshop on Databases: active and real-time (Concepts Meet Practice), DART'96, 54-58, Rockville, Md.
108. Lofrumento, G. and V. Fazio. (1993) An Object-Oriented Representation of Real-Time Application Domains. Proceedings of the Fifth Euromicro Workshop on Real-Time Systems, Oulu, Finland.
109. Jin Song, D. and Z. Lin. (1997) A framework for adding time into formal object models. Proceedings of the Third IEEE International Workshop on Object-Oriented Real-Time Dependable Systems (WORDS'97).
110. Kelling, C. and G. Hommel. (1994) Modeling priority schemes with timed Petri nets. Proceedings of the Second Workshop on Parallel and Distributed Real-Time Systems, Cancun, Mexico.
111. Mercer, C.W. and H. Tokuda. (1990) The ARTS real-time object model. Proceedings. Of the 11th IEEE Real-Time Systems Symposium, 2-10.
112. Alvarez, J. M., M. Diaz, L. Llopis, E. Pimentel and J. M. Troya (2003). "An object-oriented methodology for embedded real-time systems." *The Computer Journal* 46(2): 123.



113. Kim, J. H., H. S. Shim, H. S. Kim, M. J. Jung, I. H. Choi and J. O. Kim (1997). A cooperative multi-agent system and its real time application to robot soccer, Proceedings of the IEEE International Conference on Robotics and Automation, Minneapolis, MN, 638–643
114. Garousi, V. (2010) Experience and challenges with UML-driven performance engineering of a Distributed Real-Time System. *Information and Software Technology*. 52(6): 625-640.
115. Sorel, Y. (1994) Massively parallel computing systems with real-time constraints: the "Algorithm Architecture Adequation" methodology. Proceedings of the Massively parallel Computing Systems Ischia: IEEE.
116. Dudani, A., F. Mueller, and Y. Zhu. (2002) Energy-conserving feedback EDF scheduling for embedded systems with real-time constraints. *ACM SIGPLAN Notices*. 37(7): 213-222.
117. Alur, R. and T. Henzinger, (1992) Logics and models of real-time: A survey in "Real-Time: Theory in Practice", Lecture notes in computer science, vol 600, 74-106, Springer-verlag, New York/Berlin.
118. Lele, A. (2014) Analyzing preemptive fixed priority scheduling of data flow graphs, 2014 IEEE 12th Symposium on Embedded Systems for Real-time Multimedia (ESTIMedia), IEEE, 50–59.
119. Imes, C., D. H. Kim, M. Maggio and H. Hoffmann (2015). POET: a portable approach to minimizing energy under soft real-time constraints. *Real-Time and Embedded Technology and Applications Symposium (RTAS)*, IEEE.
120. Stigge, M. and W. Yi. (2012) Hardness results for static priority real-time scheduling. in 24th Euromicro Conference on Real-Time Systems (ECRTS), IEEE.
121. Bohlin, M., Y. Lu, J. Kraft, P. Kreuger and T. Nolte (2009). Simulation-based timing analysis of complex real-time systems, in 15th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA'09), 321-328.
122. Saksena, M., P. Freedman, and P. Rodziewicz. (1997) Guidelines for automated implementation of executable object oriented models for real-time embedded control systems. Proceedings of the IEEE Real-Time Systems Symposium, 240-251.

123. Briand, L.C., Y. Labiche, and M. Shousha. (2005) Stress testing real-time systems with genetic algorithms. Proceedings of the 7th annual conference on Genetic and evolutionary computation GECCO '05. ACM.
124. Saehwa, K., C. Sukjae, and H. Seongsoo. (2001) Automatic implementation of real-time object-oriented models and schedulability issues. Proceedings of the Sixth International Workshop on Object-Oriented Real-Time Dependable Systems, 149-153.
125. Florescu, O., J. Huang, J. Voeten and H. Corporaal (2006). Strengthening Property Preservation in Concurrent Real-Time Systems. Proceedings of the 12th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA), 106-109.
126. Chokshi, D.B. and P. Bhaduri. (2008) Modeling fixed priority non-preemptive scheduling with real-time calculus, Proceedings of the 2008 14th IEEE into conf. on Embedded and Real-Time Computing Systems and Applications (RTCSA'08).
127. Bollella, G. and J. Gosling. (2000) The real-time specification for Java. Addison-Wesley. 33(6): 47-54.
128. Lisa Cingiser DiPippo, V.F.-W., Lekshmi Nair, (2001) Ethan Hodys, Oleg Uvarov. A Real-Time Multi-Agent System Architecture for E-Commerce Applications. Fifth International Symposium on Autonomous Decentralized Systems: 357-364.
129. Sifakis, J. (2004) Modeling Real-Time Systems. In IEEE Real-Time Symposium (RTSS04), pages 5–6.
130. Puente, J.A.d.l. (2000) Real-Time Object-Oriented Design and Formal Methods. Real-Time Systems. 18(1): 79-83.
131. Ulusoy, O. (1993) Lock-based concurrency control in distributed real-time database systems. Journal of Database Management. 4(2): 3-3.
132. Sasikumar Punnekkat, A.B.a.R.D. (2004) Analysis of Checkpointing for Real-Time Systems. Real-Time Systems. 20(1): 83-102.
133. Brandenburg, B.B. (2013) A fully preemptive multiprocessor semaphore protocol for latency-sensitive real-time applications. In 25th Euromicro Conference on Real-Time Systems (ECRTS). IEEE.

134. Konrad, S. and B. H. C. Cheng (2005). Real-time specification patterns. Proceedings of the 27th international conference on Software engineering. St. Louis, MO, USA, ACM: 372-381.
135. Karangelen, N.E. and N.T. Hoang. (1994) Representing system behavior in design and analysis of large complex real-time systems. Proceedings of the Second Workshop on Parallel and Distributed Real-Time Systems.
136. Panait, L. and S. Luke. (2005) Cooperative multi-agent learning: The state of the art. *Autonomous Agents and Multi-Agent Systems*. 11(3): 387-434.
137. Fisher, N., J. Goossens, and S. Baruah. (2010) Optimal online multiprocessor scheduling of sporadic real-time tasks is impossible. *Real-Time Systems*. 45(1): 26-71.
138. Alur, R. and D. Dill. (1990) Automata for modeling real-time systems. *Automata, languages and programming*: 322-335.
139. Howe, A.E., D.M. Hart, and P.R. Cohen. (1990) Addressing real-time constraints in the design of autonomous agents. *Real-Time Systems*. 2(1):81-97.
140. Wooldridge, M., N.R. Jennings, and D. Kinny. (2000) The Gaia methodology for agent-oriented analysis and design, *Autonomous Agents and Multi-Agent Systems*. 3(3): 285-312.
141. Zeilinger, M. N., D. M. Raimondo, A. Domahidi, M. Morari and C. N. Jones (2014), "On real-time robust model predictive control, " *Automatica* 50(3): 683-694.
142. Liu, K., V. C. Lee, J. K.-Y. Ng and S. H. Son (2013), Scheduling temporal data for real-time requests in roadside-to-vehicle communication, RTCSA, Citeseer.
143. Gang, T., L. Jun-lin, Y. Fu-min and L. Wei (2007). Relationships between Window-based Real-time Constraint, In 13th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications, RTCSA.
144. Wolfgang. Halang, R.G., Matjaz Colnaric and Marjan Druzovec (2004), Measuring the Performance of Real-Time Systems *Real-Time Systems*. 18(1): 59-68.

145. Mok, A.K. (1983) Fundamental design problems of distributed systems for the hard-real-time environment, Ph.D. Dissertation, M.I.T., Cambridge, MA 02139.
146. Bresciani, P., A. Perini, P. Giorgini, F. Giunchiglia and J. Mylopoulos (2004), "Tropos: An agent-oriented software development methodology." *Autonomous Agents and Multi-Agent Systems* 8(3): 203-236.
147. Kung, D. C., J. Lin, P. Hsia and B. Carroll (1997). Object-oriented real time systems modeling and verification. *Proceedings of the Third International Workshop on Object-Oriented Real-Time Dependable Systems*.
148. Selic, B. (1996) Tutorial: real-time object-oriented modeling (ROOM), in *Real-Time Technology and Applications Symposium*, IEEE, 214-217.
149. Hanish, A.A. and T.S. Dillon. (1997) Object-oriented behaviour modelling for real-time design, *Proceedings of the Third International Workshop on Object-Oriented Real-Time Dependable Systems*, Newport Beach, IEEE, Los Alamitos/USA, S. 74–82.
150. Seung-Min, Y., Y. Tae-Myung, K. Moon Hae, M. Byoung-Joon, K. Jung-Guk and H. Shin (1996), System development based on a real-time object model, *Proceedings of WORDS '96, Second Workshop on Object-Oriented Real-Time Dependable Systems*.
151. Bruno, G. and A. Castella (1992). *Software Models For Real-time Systems*, *Proceedings of the Fourth Euromicro workshop on Real-Time Systems*.

## BIBLIOGRAPHY

1. Adams, L. A. and J. F. Courtney (2004). Achieving relevance in IS research via the DAGS framework, Proceedings of the 37th Annual Hawaii International Conference on System Sciences. Los Alamitos, CA: IEEE Computer Society Press.
2. Agudo-Peregrina, Á. F., S. Iglesias-Pradas, M. Á. Conde-González and Á. Hernández-García (2014). "Can we predict success from log data in VLEs? Classification of interactions for learning analytics and their relation with performance in VLE-supported F2F and online learning." *Computers in human behavior* 31: 542-550.
3. Alhir, S. S. (2003). *Learning Uml*, O'Reilly Online Books.
4. Ambriola, V. and Gervasi, V. (1999). Experiences with domain-based parsing of natural language requirements. In Proceedings of the Fourth International Conference on Applications of Natural Language to Information Systems, G. Flieidl and H. C. Mayr, Eds. OCG Schriftenreihe (Lecture Notes), no. 129. OGC, Klagenfurt, Austria, 145-148.
5. Apple (2016). "Apple celebrates one billion iPhones ". Retrieved 01/08/2016, from <http://www.apple.com/newsroom/2016/07/apple-celebrates-one-billion-iphones.html>.
6. Argente, E., G. Beydoun, R. Fuentes-Fernández, B. Henderson-Sellers and G. Low (2011), *Modelling with agents*, Agent-Oriented Software Engineering X, Springer: 157-168.
7. Ashamalla, A., G. Beydoun and G. Low (2011). Towards agent-oriented approach to a call management system, *Information Systems Development*, Springer New York: 345-356.
8. Ashamalla, A., G. Beydoun and G. Low (2017). "Model Driven Approach for Real-time Requirements Analysis of Multi-Agent Systems." *Computer Languages, Systems and Structures journal's (COMLAN)*, Elsevier.
9. Ashamalla, A., G. Beydoun, G. Low and J. Yan (2012). "Towards Modelling Real time Constraints." *ICSOFT 2012 7th International Conference on Software Paradigm Trends*. SciTePress Digital Library: 158-164.

10. Ashamalla, A., G. Beydoun and N. Paramesh (2014). "Real-time task attributes and temporal constraints.". AMCIS 2014 American conference on information systems. Savannah, Georgia, August 2014.
11. Attoui, A. (2000). *Real-time and Multi-agent Systems*, Springer Verlag.
12. Badano, B. I. (2008). "A multi-agent architecture with distributed coordination for an autonomous robot." University of Girona, PhD theses.
13. Barbosa, J., F. Dillenburger, G. Lermen, A. Garzão, C. Costa and J. Rosa (2012). "Towards a programming model for context-aware applications." *Computer Languages, Systems & Structures* 38(3): 199-213.
14. Basra, R., K. Lu and P. Skobelev (2007). "Resolving scheduling issues of the London Underground using a multi-agent system." *International Journal of Intelligent Systems Technologies and Applications* 2(1): 3-19.
15. Bellifemine, F., A. Poggi and G. Rimassa (2001). *Developing multi-agent systems with JADE. Intelligent Agents VII Agent Theories Architectures and Languages*, Springer: 89-103.
16. Beydoun, G., C. Gonzalez-Perez, B. Henderson-Sellers and G. Low (2006). *Developing and Evaluating a Generic Metamodel for MAS Work Products. Software Engineering for Multi-Agent Systems IV*. A. Garcia, R. Choren, C. Lucena et al., Springer Berlin Heidelberg. 3914: 126-142.
17. Beydoun, G., G. Low and P. Bogg (2013). "Suitability assessment framework of agent-based software architectures." *Information and Software Technology* 55(4): 673-689.
18. Beydoun, G., G. Low, F. García-Sánchez, R. Valencia-García and R. Martínez-Béjar (2014). "Identification of ontologies to support information systems development." *Information Systems* 46(0): 45-60.
19. Beydoun, G., G. Low, B. Henderson-Sellers, H. Mouratidis, J. J. Gomez-Sanz, J. Pavon and C. Gonzalez-Perez (2009). "FAML: a generic metamodel for MAS development.", *IEEE Transactions on Software Engineering* 35(6): 841-863.
20. Beydoun, G., G. Low, H. Mouratidis and B. Henderson-Sellers (2009). "A security-aware metamodel for multi-agent systems (MAS)." *Information and Software Technology* 51(5): 832-845.

21. Beydoun, G., G. Low, N. Tran and P. Bogg (2011). "Development of a peer-to-peer information sharing system using ontologies." *Expert Systems with Applications* 38(8): 9352-9364.
22. Beydoun, G., N. Tran, G. Low and B. Henderson-Sellers (2006). *Foundations of Ontology-Based MAS Methodologies. Agent-Oriented Information Systems III: 7th International Bi-Conference Workshop, AOIS 2005, Utrecht, Netherlands, July 26, 2005, and Klagenfurt, Austria, October 27, 2005, Revised Selected Papers.* M. Kolp, P. Bresciani, B. Henderson-Sellers and M. Winikoff. Berlin, Heidelberg, Springer Berlin Heidelberg: 111-123.
23. Bicchi, A. and L. Pallottino (2000). "On optimal cooperative conflict resolution for air traffic management systems." *Intelligent Transportation Systems, IEEE Transactions on* 1(4): 221-231.
24. Bogg, P., G. Beydoun and G. Low (2008). *When to Use a Multi-Agent System?* 11th Pacific Rim International Conference on Multi-Agents, Prima 2008. Hanoi, Vietnam, Springer Berlin / Heidelberg. Volume 5357/2008: 98-108.
25. Bohlin, M., Y. Lu, J. Kraft, P. Kreuger and T. Nolte (2009). *Simulation-based timing analysis of complex real-time systems*, in 15th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA'09), August 2009, 321-328.
26. Botti, V., C. Carrascosa, V. Julián and J. Soler (1999). "Modelling agents in hard real-time environments." *Multi-Agent System Engineering*: 63-76.
27. Botti, V. and V. Julian (2004). "Developing real-time multi-agent systems." *Integrated Computer-Aided Engineering* 11(2): 135-149.
28. Botti, V., Navarro, M. and V. Julian (2008). *Commitment Management in Real Time Multi Agent Systems.* International Symposium on Distributed Computing and Artificial Intelligence (DCAI 2008), Springer.
29. Bradshaw, J. M., S. Dutfield, P. Benoit and J. D. Woolley (1997). "KAoS: Toward an industrial-strength open agent architecture." *Proceedings of the CIKM '95 Workshop on Intelligent Information Agents*: 375-418.
30. Brazier, F. M. T., F. Cornelissen, C. M. Jonker and J. Treur (2000). "Compositional Specification and Reuse of a Generic Cooperative Agent Model." *International Journal of Cooperative Information Systems* 9(3): 171-207.

31. Bresciani, P., A. Perini, P. Giorgini, F. Giunchiglia and J. Mylopoulos (2004). "Tropos: An agent-oriented software development methodology." *Autonomous Agents and Multi-Agent Systems* 8(3): 203-236.
32. Broster, A. B. a. G. R.-N. (2005). "Timing Analysis of Real-Time Communication Under Electromagnetic Interference." *Real-Time Systems* 30(1-2): 55-81.
33. Budgen, D. and P. Brereton (2006). Performing systematic literature reviews in software engineering, in: *Proceedings of the 28th International Conference on Software Engineering (ICSE'06)*, Shanghai, China, 1051-1052.
34. Cabri, G., L. Leonardi and F. Zambonelli (2003). *BRAIN: A Framework for Flexible Role-Based Interactions in Multiagent Systems. On The Move to Meaningful Internet Systems: CoopIS, DOA, and ODBASE*, Springer Berlin / Heidelberg. 2888: 145-161.
35. Carrera, Á., C. A. Iglesias and M. Garijo (2014). "Beast methodology: An agile testing methodology for multi-agent systems based on behaviour driven development." *Information Systems Frontiers* 16(2): 169-182.
36. Červenka, R., I. Trenčanský, M. Calisti and D. Greenwood (2005). *AML: Agent modeling language toward industry-grade agent-based modeling. Agent-Oriented Software Engineering V*, Springer: 31-46.
37. Cha, S. K., B. D. Park, S. J. Lee, S. H. Song, J. H. Park, J. S. Lee, S. Y. Park, D. Y. Hur and G. B. Kim (1995). Object-oriented design of main-memory DBMS for real-time applications, *Proceedings of the Second International Workshop on Real Time Computing Systems and Applications*
38. Chen, B. and H. H. Cheng (2010). "A review of the applications of agent technology in traffic and transportation systems." *Intelligent Transportation Systems, IEEE Transactions on* 11(2): 485-497.
39. Cheng, P. D.-M. (2012). "RTMAS: An Expert System for Real Time Monitoring and Analysis." *Expert Systems: Applications to Urban Planning*: 105.
40. Chung, L., B. A. Nixon, E. Yu and J. Mylopoulos (2012). *Non-functional requirements in software engineering*, Springer Science & Business Media.
41. Chise, C., I. Jurca and Ieee (2009). Towards Early Performance Assessment Based on UML MARTE Models for Distributed Systems, *proceedings of the*



- 5th International Symposium on Applied Computational Intelligence and Informatics (SACI '09), New York, IEEE: 511-516.
42. CityRail (2014). "Sydney Trains Timetables." from <http://www.sydneytrains.info/>.
  43. Colin, A., K. Thomas, et al. (2003). "Model-Driven Development: A Metamodeling Foundation." *IEEE Software* 20(5): 36-41.
  44. Collinot, A., A. Drogoul and P. Benhamou (1996). Agent oriented design of a soccer robot team. *Proceedings of the Second International Conference on Multi-Agent Systems (ICMAS-96)*.
  45. Colson, C. M. and M. H. Nehrir (2013). "Comprehensive real-time microgrid power management and control with distributed agents." *IEEE Transactions on Smart Grid* 4(1): 617-627.
  46. Cossentino, M., V. Hilaire, N. Gaud, S. Galland and A. Koukam (2014). The ASPECS Process. *Handbook on Agent-Oriented Design Processes*, Springer: 65-114.
  47. Cossentino, M. and V. Seidita (2014). PASSI: Process for agent societies specification and implementation. *Handbook on Agent-Oriented Design Processes*, Springer: 287-329.
  48. Cross, N. (2007). "Forty years of design research." *Design studies* 28(1): 1-4.
  49. Dalessandro, L., V. J. Marathe, M. F. Spear and M. L. Scott (2007). Capabilities and limitations of library-based software transactional memory in C++. *Proceedings of the 2nd ACM SIGPLAN Workshop on Transactional Computing*.
  50. Damir Isovici and G. Fohler (2009). "Handling mixed sets of tasks in combined offline and online scheduled real-time systems." *Real-Time Systems* 43(3): 296-325.
  51. DeLoach, S. A., M. F. Wood and C. H. Sparkman (2001). "Multiagent systems engineering." *International Journal of Software Engineering and Knowledge Engineering* 11(03): 231-258.
  52. DeLoach, S. A. and J. C. Garcia-Ojeda (2014). The O-MaSE Methodology. *Handbook on Agent-Oriented Design Processes*, Springer: 253-285.
  53. Demathieu, S., F. Thomas, C. André, S. Gérard and F. Terrier (2008). First experiments using the UML profile for MARTE, 11th IEEE International

- Symposium on Object Oriented Real-Time Distributed Computing (ISORC), IEEE.
54. Denaro, G., A. Polini and W. Emmerich (2004). Early performance testing of distributed software applications. ACM SIGSOFT Software Engineering Notes, ACM.
  55. DiPippo, L. C., V. Fay-Wolfe, L. Nair, E. Hodys and O. Uvarov (2001). A real-time multi-agent system architecture for e-commerce applications. Fifth International Symposium on Autonomous Decentralized Systems, Dallas, Texas, IEEE Computer Society Washington, DC, USA.
  56. Doe, J. (2007). "Call Center Simulation." Retrieved 03/09/2008, from <http://contactcenter.limra.com/Products/Samples/Ccs.pdf>.
  57. Elsayah, S., J. H. Guillaume, T. Filatova, J. Rook and A. J. Jakeman (2015). "A methodology for eliciting, representing, and analysing stakeholder knowledge for decision making on complex socio-ecological systems: From cognitive maps to agent-based models." *Journal of environmental management* 151: 500-516.
  58. Ephrati, E. and J. S. Rosenschein (1992). Constrained intelligent action: Planning under the influence of a master agent, In *Proceedings of the Tenth National Conference on Artificial Intelligence*, San Jose, California: 263-268.
  59. Estefan, J. A. (2007). "Survey of model-based systems engineering (MBSE) methodologies.", Rev. B, INCOSE-TD-2007-003-01, MBSE Initiative, International Council on Systems Engineering, Seattle, WA.
  60. Faliagka, E., P. Karkoulias, M. Rigkou, S. Sirmakessis, G. Tzimas and A. Tsakalidis (2011). Investigating the Integration of Hand-Held Haptic Devices in Daily Work Activities: The Case of a Tennis Coaching Assistant on iPhone. *Design, User Experience, and Usability. Theory, Methods, Tools and Practice*, Springer: 555-563.
  61. Franch, X., C. Quer, S. Renault, C. Guerlain and C. Palomares (2013). *Constructing and Using Software Requirements Patterns. Managing requirements knowledge*, Springer: 95-116.
  62. Fuentes-Fernández, R., I. García-Magariño, A. M. Gómez-Rodríguez and J. C. González-Moreno (2010). "A technique for defining agent-oriented

- engineering processes with tool support." *Engineering Applications of Artificial Intelligence* 23(3): 432-444.
63. Garcia-Ojeda, J. C., S. A. DeLoach, W. H. Oyenand and J. Valenzuela (2008). O-MaSE: a customizable approach to developing multiagent development processes, In *Proceedings of the 8th international workshop on agent oriented software engineering (AOSE 2007)*, 1-15.
  64. Garijo, F. J., J. J. Gomez-Sanz and P. Massonet (2005). "The MESSAGE methodology for agent-oriented analysis and design." *Agent-Oriented Methodologies*, IDEA Group Publishing 8: 203-235.
  65. Garousi, V., L. C. Briand and Y. Labiche (2009). "A UML-based quantitative framework for early prediction of resource usage and load in distributed real-time systems." *Software & Systems Modeling* 8(2): 275-302.
  66. Gary B, C., Plano; Paul H. Lemon, McKinney (2000). Skills-based scheduling for telephone call centers. US Patent. USA. 6044355.
  67. Genesys (2009). "Call centres matching demographics." Retrieved 1/4/2009, 2009, from <http://callcentres.net/CALLCENTRES/LIVE/me.get?site.sections&CALL1575>.
  68. Georgeff, M. P. and A. L. Lansky (1986). "Procedural knowledge." *Proceedings of the IEEE* 74(10): 1383-1398.
  69. Ghaisas, S. and N. Ajmeri (2013). Knowledge-assisted ontology-based requirements evolution. *Managing requirements knowledge*, Springer: 143-167.
  70. Ghidini, C., B. Kump, S. Lindstaedt, N. Mahbub, V. Pammer, M. Rospocher and L. Serafini (2009). Moki: The enterprise modelling wiki. *The Semantic Web: Research and Applications*, Springer: 831-835.
  71. Giorgini, P., J. Mylopoulos and R. Sebastiani (2005). "Goal-oriented requirements analysis and reasoning in the tropos methodology." *Engineering Applications of Artificial Intelligence* 18(2): 159-171.
  72. Giunchiglia, F., J. Mylopoulos and A. Perini (2003). "The tropos software development methodology: processes, models and diagrams." *Agent-Oriented Software Engineering III*: 162-173.

73. Gokhale, A. S., B. Natarajan, D. C. Schmidt and J. K. Cross (2004). "Towards real-time fault-tolerant CORBA middleware." *Cluster Computing* 7(4): 331-346.
74. Goknil, A., I. Kurtev, K. Van den Berg and J.-W. Veldhuis (2011). "Semantics of trace relations in requirements models for consistency checking and inferencing." *Software & Systems Modeling* 10(1): 31-54.
75. Goldman, G. H. (2000). *Characterization of the Effects of Cavities and Canopies on Radar Target Signatures*, US Army Research Laboratory, Adelphi, MD, ARL-TN-154.
76. Golpelwar, M. K. (2015). *Global Call Center Employees in India: Work and Life between Globalization and Tradition*, Springer.
77. Gómez, R. (2009). A compositional translation of timed automata with deadlines to uppaal timed automata. *Formal Modeling and Analysis of Timed Systems*, Springer: 179-194.
78. Gómez-Sanz, J. J., C. R. Fernández and J. Arroyo (2010). "Model driven development and simulations with the INGENIAS agent framework." *Simulation Modelling Practice and Theory* 18(10): 1468-1482.
79. Google (2014). "Google Maps." 2014, from <https://www.google.com.au/maps/>.
80. Gumzej, R. and M. Colnaric (2001). An approach to modeling and verification of real-time systems. *Proceedings of the Fourth IEEE International Symposium on Object-Oriented Real-Time Distributed Computing (ISORC)*.
81. Hahn, C., C. Madrigal-Mora and K. Fischer (2009). "A platform-independent metamodel for multiagent systems." *Autonomous Agents and Multi-Agent Systems* 18(2): 239-266.
82. Hassine, J., J. Rilling and R. Dssouli (2010). "An evaluation of timed scenario notations." *Journal of Systems and Software* 83(2): 326-350.
83. Hevner, A. R., S. T. March, J. Park and S. Ram (2004). "Design Science in Information Systems Research." *Mis Quarterly* 28(1): 75-105.
84. Hiroyuki, N., K. Takuya and H. Shinichi (2006). Analysis of multi-agent systems based on KAOS modeling. *Proceedings of the 28th international conference on Software engineering*. Shanghai, China, ACM: 926 - 929

85. Horkoff, J. (2007). "Visio." Retrieved Thursday 24 of May, 2007 23:13:15, 2007, from <http://tanne.informatik.rwth-aachen.de:7777/tiki-index.php?page=VISIO>.
86. Hsin-Wen, W., H. Pei-Chi, C. Hsung-Pin and S. Wei-Kuan (2005). Scheduling real time information in a broadcast system with non real time information, Proceedings of the 11th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications.
87. Hull, M. E. C., S. Ewart and J. R. P. Hanna (2004). "Modeling Complex Real-Time and Embedded Systems—The UML and DORIS Combination." *Real-Time Systems* 26(2): 135-159.
88. Iglesias, C. A., M. Garijo, J. C. González and J. R. Velasco (1996). A methodological proposal for multiagent systems development extending CommonKADS. Proceedings of the 10th Banff knowledge acquisition for knowledge-based systems workshop, Citeseer.
89. Jayashankar, M. S., F. S. Stephen and M. S. Norman (1998). "Modeling supply chain dynamics: A multiagent approach." *Decision sciences* 29(3): 607.
90. Jurisica, I., J. Mylopoulos and E. Yu (2004). "Ontologies for Knowledge Management: An Information Systems Perspective." *Knowledge and Information Systems* 6(4): 380-401.
91. Karangelen, N. E. and N. T. Hoang (1994). Representing system behavior in design and analysis of large complex real-time systems, Proceedings of the Second Workshop on Parallel and Distributed Real-Time Systems.
92. Kardas, G. (2013). "Model-driven development of multiagent systems: a survey and evaluation." *The Knowledge Engineering Review* 28(04): 479-503.
93. KARDAS, G., A. GOKNIL, O. DIKENELLI and N. Y. TOPALOGLU (2009). "MODEL DRIVEN DEVELOPMENT OF SEMANTIC WEB ENABLED MULTI-AGENT SYSTEMS." *International Journal of Cooperative Information Systems* 18(02): 261-308.
94. Kellogg, M. (2016). "Word Reference." from <http://www.wordreference.com/definition/task>.
95. Kim, H. J., I. Kim and H. Lee (2016). "Third-party mobile app developers' continued participation in platform-centric ecosystems: An empirical

- investigation of two different mechanisms." *International Journal of Information Management* 36(1): 44-59.
96. Kitchenham, B., O. P. Brereton, D. Budgen, M. Turner, J. Bailey and S. Linkman (2009). "Systematic literature reviews in software engineering—a systematic literature review." *Information and Software Technology* 51(1): 7-15.
  97. Koji Iwamura, N. M., Yoshitaka Tanimizu and Nobuhiro Sugimura (2009). "A Study on Real-Time Scheduling for Holonic Manufacturing Systems – Determination of Utility Values Based on Multi-agent Reinforcement Learning." *Lecture Notes in Computer Science* 5696/2009: 135-144.
  98. Konrad, S. and B. H. C. Cheng (2005). Real-time specification patterns. *Proceedings of the 27th international conference on Software engineering*. St. Louis, MO, USA, ACM: 372-381.
  99. Konrad, S. J. (2006). Model-driven development and analysis of high assurance systems. Department of Computer Science. Michigan, Michigan State University. DOCTOR OF PHILOSOPHY: 443.
  100. Kopetz, H. (2000). Software engineering for real-time: A roadmap. *Proc. 22nd Int. Conf. Software Eng*, Citeseer.
  101. Larue, D. L. C. R., IA), Ivey, James Brian (Marion, IA), Leonard, Timothy M. (Coralville, IA) (1999). Generalized customer profile editor for call center services. U. S. Patent. United States, MCI Communications Corporation (Washington, DC).
  102. Laugesen, J. and Y. Yuan (2010). What factors contributed to the success of Apple's iPhone?, *Proceedings of the Ninth International Conference on Mobile Business and 2010 Ninth Global Mobility Roundtable (ICMB-GMR)*, IEEE.
  103. Ling, R. and P. R. Sundsoy (2009). The iPhone and mobile access to the internet. *ICA pre-conference on mobile communication*, Chicago, IL.
  104. Lisa Cingiser DiPippo, V. F.-W., Lekshmi Nair, Ethan Hodys, Oleg Uvarov (2001). "A Real-Time Multi-Agent System Architecture for E-Commerce Applications." *Proceedings of the Fifth International Symposium on Autonomous Decentralized Systems*: 357-364.
  105. Llega, A. G. T. J. H. (2016). "Computer programming: Know How to Flowchart." Retrieved 12/10/2016, from

<http://www.slideshare.net/Dkiceiceice/computer-programmingknow-how-to-flowchart>.

106. Lopez-Lorca, A. A., G. Beydoun, L. Sterling and T. Miller (2011). Ontology-mediated Validation of Software Models. *Information Systems Development*, Springer: 455-467.
107. Lopez-Lorca, A. A., G. Beydoun, R. Valencia-Garcia and R. Martinez-Bejar (2016). "Supporting agent oriented requirements analysis with ontologies." *International Journal of Human-Computer Studies* 87: 20-37.
108. Mallet, F. (2008). "Clock constraint specification language: specifying clock constraints with UML/MARTE." *Innovations in Systems and Software Engineering* 4(3): 309-314.
109. March, S. T. and V. C. Storey (2008). "Design science in the information systems discipline: an introduction to the special issue on design science research." *Management Information Systems Quarterly* 32(4): 6.
110. Marin, O., P. Sens, J.-P. Briot and Z. Guessoum (2001). Towards adaptive fault tolerance for distributed multi-agent systems. *Proceedings of ERSADS*.
111. Mayer, D. A., D. Teubert, S. Wetzel, U. Meyer and G. Neugebauer (2010). appoint-A Distributed Privacy-Preserving iPhone Application. 3rd ACM Conference on Wireless Security (WISEC), Poster Session.
112. Micacchi, C. and R. Cohen (2008). "A framework for simulating real-time multi-agent systems." *Knowledge and Information Systems* 17(2): 135-166.
113. Miller, T., S. Pedell, L. Sterling and B. Lu (2011). Engaging Stakeholders with Agent-Oriented Requirements Modelling. *Agent-Oriented Software Engineering XI*. D. Weyns and M.-P. Gleizes, Springer Berlin Heidelberg. 6788: 62-78.
114. Moon Hae, K., L. Sun-Hwa and K. Jung-Guk (2003). Modeling of a real-time distributed network management based on TMN and the TMO model, *Proceedings of the Eighth International Workshop on Object-Oriented Real-Time Dependable Systems (WORDS 2003)*.
115. Moraitis, P. and N. Spanoudakis (2006). "The Gaia2Jade process for multi-agent systems development." *Applied Artificial Intelligence* 20(2-4): 251-273.

116. Morandini, M., F. Dalpiaz, C. D. Nguyen and A. Siena (2014). The Tropos Software Engineering Methodology. Handbook on Agent-Oriented Design Processes, Springer: 463-490.
117. Mouratidis, H. and P. Giorgini (2007). "Secure tropos: a security-oriented extension of the tropos methodology." International Journal of Software Engineering and Knowledge Engineering 17(02): 285-309.
118. N. A. Sabour, H. M. F. a. M. E. K. (2008). "Multi-Agent Based Framework for Target Tracking Using a Real Time Vision System." International Conference on Computer Engineering and Systems, ICCES 2008 355-363
119. Nakashima, H., H. Fujii and M. Suwa (2014). Designing methodology for innovative service systems. Serviceology for Services, Springer: 287-295.
120. Navarroa, M., V. Julian, S. Heras, J. Soler and V. Botti (2006). Multi-Agent systems over RT-Java for a mobile robot control. International Conference on Intelligent Data Engineering and Automated Learning, Springer.
121. Neto, A., F. Sartori, F. Piccolo, R. Vitelli, G. De Tommasi, L. Zabeo, A. Barbalace, H. Fernandes, D. F. Valcárcel and A. J. N. Batista (2009). MARTE: a Multi-Platform Real-Time Framework. Proc. of the 16th IEEE NPSS Real-Time Conference, Beijing, China.
122. Nguyen, T. M., J. Schiefer and A. M. Tjoa (2007). "ZELESSA: an enabler for real-time sensing, analysing and acting on continuous event streams." International Journal of Business Intelligence and Data Mining 2(1): 105-141.
123. NoahGans, G. K., Avishai Mandelbaum (2003). "Telephone call centers :tutorial,review and research prospects." Manufacturing and service operations management 5: 79-141.
124. OMG, O. M. G. (2008). "UML profile for modeling and analysis of real-time and embedded systems (MARTE)." Retrieved 2-12-2010, from <http://www.omg.org/>.
125. Othman, S. H. and G. Beydoun (2013). "Model-driven disaster management." Information & Management 50(5): 218-228.
126. Palopoli, L., G. Buttazzo and P. Ancilotti (1999). A C language extension for programming real-time applications. Sixth International Conference on Real-Time Computing Systems and Applications (RTCSA '99).



127. Papasimeon, M. and C. Heinze (2001). Extending the UML for designing JACK agents. 13th Australian Software Engineering Conference (ASWEC'01), Canberra, Australia
128. Pavón, J., J. Gómez-Sanz and R. Fuentes (2006). Model Driven Development of Multi-Agent Systems. Model Driven Architecture – Foundations and Applications: Second European Conference, ECMDA-FA 2006, Bilbao, Spain, July 10-13, 2006. Proceedings. A. Rensink and J. Warmer. Berlin, Heidelberg, Springer Berlin Heidelberg: 284-298.
129. Peffers, K., T. Tuunanen, M. A. Rothenberger and S. Chatterjee (2007). "A design science research methodology for information systems research." *Journal of Management Information Systems* 24(3): 45-77.
130. Picard, G. (2003). UML Stereotypes Definition and AUML Notations for ADELFE Methodology with OpenTool. First European Workshop on Multi-Agent Systems (EUMAS'03).
131. Pires, P. F., F. C. Delicato, R. Cobe, T. Batista, J. G. Davis and J. H. Song (2011). "Integrating ontologies, model driven, and CNL in a multi-viewed approach for requirements engineering." *Requirements Engineering* 16(2): 133-160.
132. Prasad, M. N. and V. R. Lesser (1999). "Learning situation-specific coordination in cooperative multi-agent systems." *Autonomous Agents and Multi-Agent Systems* 2(2): 173-207.
133. Riemsdijk, M. B. v., D. Mehdi, M. John-Jules Ch and S. d. B. Frank (2006). Goal-oriented modularity in agent programming. Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems. Hakodate, Japan, ACM.
134. Rodrigues da Silva (2015). "Model-driven engineering: A survey supported by the unified conceptual model." *Computer Languages, Systems & Structures* 43: 139-155.
135. Rodriguez, J. A. (2003). "On The Design and Construction of Agent-Mediated Electronic Institutions." Artificial Intelligence Research Institute. Barcelona, UAB - Universitat Autònoma de Barcelona.
136. Roger, S. Pressman. (2010). "Software engineering: a practitioner's approach." McGraw-Hill International Edition.

137. Rong-he, D., L. I. Xiao-an and D. Jun-hua (2009). "A Study of the Real-Time MAS Architecture and the Modeling Method in Dynamic Environments." *Computer Engineering & Science*, Vol. 01.
138. Sadraei, E., A. Aurum, G. Beydoun and B. Paech (2007). "A field study of the requirements engineering practice in Australian software industry." *Requirements Engineering* 12(3): 145-162.
139. Saehwa, K., C. Sukjae and H. Seongsoo (2000). Schedulability-aware mapping of real-time object-oriented models to multi-threaded implementations. *Proceedings of the Seventh International Conference on Real-Time Computing Systems and Applications*.
140. Selic, B. (2003). Model-driven development of real-time software using OMG standards, *Sixth IEEE International Symposium on Object-Oriented Real-Time Distributed Computing*.
141. Selic, B. (2003). "The pragmatics of model-driven development." *Software, IEEE* 20(5): 19-25.
142. Shamshirband, S., N. B. Anuar, M. L. M. Kiah and A. Patel (2013). "An appraisal and design of a multi-agent system based cooperative wireless intrusion detection computational intelligence technique." *Engineering Applications of Artificial Intelligence* 26(9): 2105-2127.
143. Shen, J., G. Beydoun, G. Low and L. Wang (2014). "Aligning ontology-based development with service oriented systems." *Future Generation Computer Systems* 32: 263-273.
144. Shih, G., P. Lakhani and P. Nagy (2010). "Is android or iPhone the platform for innovation in imaging informatics." *Journal of Digital Imaging* 23(1): 2-7.
145. Siegemund, K., E. J. Thomas, Y. Zhao, J. Pan and U. Assmann (2011). Towards ontology-driven requirements engineering. *Workshop Semantic Web Enabled Software Engineering at 10th International Semantic Web Conference (ISWC), Bonn*.
146. Silly-Chetto, A. M. a. M. (2006). "Dynamic Real-time Scheduling of Firm Periodic Tasks with Hard and Soft Aperiodic Tasks " *Real-Time Systems* 32(1-2): 21-47.
147. Simon, H. A. (1996). *The sciences of the artificial*, MIT press.

148. Soh, L.-K. and C. Tsatsoulis (2005). "A real-time negotiation model and a multi-agent sensor network implementation." *Autonomous Agents and Multi-Agent Systems* 11(3): 215-271.
149. Thomas R. Robbins, D. J. M. a. P. D. (2006). Evaluating arrival rate uncertainty in call centers. *Simulation Conference, WSC 06. Proceedings of the Winter. Monterey, California Winter Simulation Conference: 2180 - 2187*
150. Thomas S. Fisher, R. A. J. a. M. I. R. (2003). System for automatically predicting call center agent work time in a multi-skilled agent environment U. S. Patent. USA. US 6,553,114 B1.
151. Tran, N., G. Beydoun and G. Low (2007). Design of a Peer-to-Peer Information Sharing MAS Using MOBMAS (Ontology-Centric Agent Oriented Methodology). *Advances in Information Systems Development: New Methods and Practice for the Networked Society*. W. Wojtkowski, W. G. Wojtkowski, J. Zupancic, G. Magyar and G. Knapp. Boston, MA, Springer US: 63-76.
152. Tran, Q. N. N., G. Low and G. Beydoun (2006). "A methodological framework for ontology centric oriented software engineering." *International Journal of Computer Systems Science & Engineering* 21(2): 117-132.
153. Tran, Q.-N. N., G. Beydoun, G. Low and C. Gonzalez-Perez (2008). Preliminary validation of MOBMAS (ontology-centric agent oriented methodology): design of a peer-to-peer information sharing MAS. *Agent-Oriented Information Systems IV*, Springer: 73-89.
154. Tran, Q.-N. N. and G. Low (2008). "MOBMAS: A methodology for ontology-based multi-agent systems development." *Information and Software Technology* 50(7): 697-722.
155. Vikhorev, K., N. Alechina and B. Logan (2009). The ARTS Real-Time Agent Architecture. *Proceedings of Second Workshop on Languages, Methodologies and Development Tools for Multi-agent Systems*.
156. Villaplana, E. a. (2005). "A proposal for an organizational MAS methodology." *International Conference on Autonomous Agents*.
157. Vincent Conitzer, T. S. (2007). "AWESOME: A general multiagent learning algorithm that converges in self-play and learns a best response against stationary opponents." *Machine Learning* 67 (1-2/may 2007): 23 - 43

158. Wagner, G. and K. Taveter (2004). Towards radical agent-oriented software engineering processes based on AOR modeling. Proceedings. IEEE/WIC/ACM International Conference on Intelligent Agent Technology. (IAT 2004).
159. Wagner, G. (2000). Agent-object-relationship modeling. In Proc. of Second International Symposium - from Agent Theory to Agent Implementation together with EMCRS 2000, Vienna, Austria
160. Wallace, S., M. Clark and J. White (2012). "'It's on my iPhone': attitudes to the use of mobile computing devices in medical education, a mixed-methods study." *BMJ open* 2(4): e001099.
161. Wang, S., M. Van Schyndel, G. Wainer, V. S. Rajus and R. Woodbury (2012). Devs-based building information modeling and simulation for emergency evacuation. Proceedings of the 2012 Winter Simulation Conference (WSC), IEEE.
162. Wautelet, Y. and M. Kolp (2016). "Business and model-driven development of BDI multi-agent systems." *Neurocomputing* 182: 304-321.
163. Webber, R. (2007). "Using names to segment customers by cultural, ethnic or religious origin " *Journal of Direct, Data and Digital Marketing Practice* 8(3): 226-242.
164. West, J. and M. Mace (2010). "Browsing as the killer app: Explaining the rapid success of Apple's iPhone." *Telecommunications Policy* 34(5): 270-286.
165. Westley, W. and C. N. George (2004). "Finding and preventing run-time error handling mistakes." *SIGPLAN Not.* 39(10): 419-431.
166. Weyns, D., A. Omicini and J. Odell (2007). "Environment as a first class abstraction in multiagent systems." *Autonomous Agents and Multi-Agent Systems* 14(1): 5-30.
167. Whipple, J., W. Arensman and M. S. Boler (2009). A public safety application of GPS-enabled smartphones and the android operating system. International Conference on Systems, Man and Cybernetics, SMC 2009, IEEE.
168. Wolfgang. Halang, R. G., Matjaz Colnaric and Marjan Druzovec (2004). "Measuring the Performance of Real-Time Systems " *Real-Time Systems* 18(1): 59-68.

169. Wooldridge, M. and P. Ciancarini (2001). "Agent-oriented software engineering: The state of the art." *Lecture Notes in Computer Science*: 1-28.
170. Wooldridge, M., N. R. Jennings and D. Kinny (2000). "The Gaia methodology for agent-oriented analysis and design." *Autonomous Agents and Multi-Agent Systems* 3(3): 285-312.
171. Xu, D., C. Wijesooriya, Y.-G. Wang and G. Beydoun (2011). "Outbound logistics exception monitoring: A multi-perspective ontologies' approach with intelligent agents." *Expert Systems with Applications* 38(11): 13604-13611.
172. Yu, E. S. K. (1995). *Modelling strategic relationships for process reengineering*. Computer Science. Toronto, Canada, University of Toronto. Doctor of Philosophy: 131.
173. Zahariev, A. (2009). "Google app engine." Helsinki University of Technology.
174. Zahia Guessoum, J.-P. B., Pierre Sens, and Olivier Marin (2001). "Toward Fault-Tolerant Multi-agent Systems." from <http://www-poleia.lip6.fr/~briot/publications/ft-mas-maamaw01.pdf>.
175. Zambonelli, F., N. Jennings, M. Wooldridge and P. Ciancarini (2001). *Organisational Abstractions for the Analysis and Design of Multi-agent Systems*. Agent-Oriented Software Engineering, Springer Berlin / Heidelberg. 1957: 407-422.
176. Zambonelli, F., N. R. Jennings and M. Wooldridge (2003). "Developing multiagent systems: The Gaia methodology." *ACM Transactions on Software Engineering and Methodology (TOSEM)* 12(3): 317-370.
177. Zambonelli, F., N. R. Jennings and M. Wooldridge (2005). "Multi-agent systems as computational organizations: the Gaia methodology." *Agent-Oriented Methodologies* 6: 136-171.
178. Zeynep Aksin, M. A., Vijay Mehrotra (2007). "The Modern Call-Center: A Multi-Disciplinary Perspective on Operations Management Research " *Production and Operations Management* 16(6): 665-689.
179. Zhang, L. (2006). "Development Method for Multi-Agent Real-Time Systems." *International Journal of Information Technology* 12(5).
180. Zhang, X. and H. Xu (2006). *Towards Automated Development of Multi-Agent Systems Using RADE*. IC-AI.

181. Zhang, C., A. Hammad and H. Bahnassi (2009). "COLLABORATIVE MULTI-AGENT SYSTEMS FOR CONSTRUCTION EQUIPMENT BASED ON REAL-TIME FIELD DATA CAPTURING." *Electronic Journal of Information Technology in Construction* 14: 204-228.
182. Zhang, Y. and A. K. Mackworth (1993). *Design and analysis of embedded real-time systems: An elevator case study*, Citeseer.