

Towards Modelling Real Time Constraints

Amir Ashamalla¹, Ghassan Beydoun¹, Graham Low² and Jun Yan¹

¹*School of Information Systems and Technology, University of Wollongong, Wollongong, Australia*

²*School of Information Systems, Technology and Management, University of New South Wales, Kensington, Australia*

Keywords: i* Requirement Models, Multi Agent System, Call Management Centre (CMC), Relationship Manager (RM), Real Time Multi Agent Systems (RTMAS).

Abstract: Software agents are highly autonomous, situated and interactive software components. They autonomously sense their environment and respond accordingly. Agents behaviours are often constrained by real time constraints such as the time in which the agent is expected to respond .i.e. time needed for a task to complete. Failing to meet such a constraint can result in a task being not achieved. This possibly causes an agent or a system to fail, depending on how critical the task is to the agent or system as a whole. Our research aims at identifying and modelling real time constraints in the early phase of analysis which helps in creating a more reliable and robust system.

1 INTRODUCTION AND RELATED WORK

Agents' key characteristics are autonomy, interactivity, situatedness and cooperativeness (Beydoun et al 2009; Beydoun et al 2006). They are typically designed to meet local objectives as part of a distributed system. A *real-time agent* is such an agent with temporal restrictions in some of its allocated responsibilities or tasks (Botti et al 2004). This paper is motivated by the longstanding view that the earlier you model real time requirements in the software development life cycle, the more reliable and robust the resultant system should be (Boehm 1988; Sadrei et al 2007). Any future issues and conflicts are identified and resolved in the earlier stage of analysis rather than in later stages of design and implementation when it is too late or too hard to resolve. We identify a number of key real time constraints that can be modelled during the requirement analysis of the system. The rest of the paper is organised as follows: We first discuss other academics work on real time multi agent systems. We then sketch modelling real time constrains. This is followed by the details of the identified real time constraints set. We then introduce a call management system as a validation domain to demonstrate how the identified real time constraints set are integrated into the software development life cycle using i* modelling. We finally conclude this

paper with a description of future work and anticipated challenges.

Surprisingly, for MAS systems that are supposed to be decentralised and distributed, a common modelling approach to for ensuring realtime constraints are met is through the use of a *central monitoring* agent (master agent) (Neto 2009) which receives completion reports from the rest of the agents. The monitoring agent typically initiates a redundant task if an agent charged with a task does not report completing it within the required timeframe (*a real-time constraint*) (Neto, 2009). This approach clearly presents a single point of failure and is contrary to the distributedness of MAS and its engendered appeal. The approach pursued in this research seeks to maintain distributedness, fulfilling real time requirements identified during the requirement analysis phase of MAS development. Modelling real time agent interactions has been considered in a number of real-time MAS applications. Notable examples include: The London Underground project Basra (2007) used agent modelling to model messaging and actions taken by other trains to avoid collision, a search and rescue example (Micacchi 2008) modelled how a robot can identify and then plan to avoid obstacles to rescue victims in real-time, target tracking (Sabour 2008), construction (Zhang 2009) and automated car driving (Konrad 2006).

A principal requirement for real time systems is fulfilling time constraints (Vahid 2010). When developing a model for real time MAS, the relative priority of the task should be taken into account, as well as the task deadline. In another model (Zambonelli 2001), agents broadcast their set of tasks to agents they rely on, and negotiate these set of tasks before they start executing their tasks. Our work is closer to Lu (2006) who suggests task negotiation and cooperation should happen on regular basis to update task status. The work is similar to ours in that it promotes a distributed approach to monitor real-time constraints satisfaction. However, it is based on a numerical representation of the conditions that are quite difficult for software engineer to use during the analysis phase. The work actually relies on task sampling frequency which may under some conditions impact the overall performance (dangbing 2004).

Object Management Group (OMG) and IBM have developed a new improved profile, called the UML Profile for Modelling and Analysis of Real-time and Embedded Systems (MARTE) (OMG, 2008). MARTE models the analysis and design of real time systems based the following four fundamental pillars: QoS-aware Modelling, Architecture Modelling, Platform-based Modelling, and Model-based QoS Analysis. MARTE has been integrated into IBM rational rhapsody version 7.5.

Another modelling and analysis suite is UML-MAST (Modelling and Analysis Suite for Real-Time Applications). UML-MAST distributes the load based on the cpu, memory and network utilization, and not on the task priority or deadlines. The suit uses equations and experience to calculate and predict the tasks load or cpu, memory and network usage and then load balanced the tasks based on it. e.g. task data size based on parameter data types indicates network traffic as well as the number of nodes (routers) that are exchanged between the sender and receiver, this enabled predicting traffic on the node, though data size and number of messages/tasks (Vahid 2009). These modelling suits do not have graphical representations for the real time constraints, especially not for multi agent systems which is the focus of our research.

2 MODELLING AGENT REAL-TIME CONSTRAINTS

In this section, we introduce fundamental concepts that underpin modelling of agent real time

constraints. This includes an elaboration on the difference between real time constraints, error handling and fault tolerant systems.

A task taking too long to complete may be regarded as a failed task when a real-time constraint applies. Receiving the right answer too late becomes the wrong answer (Gokhale 2004). Run time error and exception handling in the development phase; typically require a different set of tasks to be initiated when an error occurs (Westley 2004). If the task is mission-critical and takes too long to complete, it can lead to unwanted consequences e.g. dialling a number then having to wait long for an answer cannot be regarded as successful- although the phone rang. The fact that the response time was too long means the task failed, as it did not meet its time constraint. This is different from fault tolerance where the latter focuses on the behaviour of the task following a failure. This may include starting an alternate task to fulfil the application goals. Our research regards tasks taking longer than an expected/accepted time period as “failed” to meet the design goals, regardless of their eventual outcome.








Accurate identification of the violation of a real-time constraint can be complicated. It often requires taking into account task dependencies. For instance, a task A may take too long simply because it is waiting for its required input from another task B. The problem may lie with Task B rather than Task A. In the context of agents within a MAS, this kind of dependency may be compounded and take the form of a chain of dependencies of tasks and agent goals (Neto 2009). In other words, all agent features must be considered and modelled (Cabri 2003) with their time related features. Our research aims at providing a knowledge representation to facilitate identifying a sufficient set of activities to be carried out by requirement analysts to later be able to identify which task has failed to meet its time constraints. We aim to be able to identify the available and the proper behaviour set for the agent to be notified, and to model the required recovery behaviour when a task fails. In other words, two types of knowledge have to identified and modelled: the knowledge to identify the success or failure of the task to meet its real time constraints and the knowledge describing behavioural actions associated with a failed task. It is worth noting that modelling the behaviour criteria alone can lead to modelling a fault tolerant system (Kopetz 2000), as the research focus would be on what actions are needed to recover from a task failure.

Our research will enable better planning to avoid future problems that might arise as a result of not meeting real time constraints. 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. Our goal is not to address fault tolerance issues. We synthesize a reliable and precise analysis process to ensure that we capture the real-time constraints and the concomitant required agent's behaviour. As part of formulating this process, we identify a set of constraints that guide analysts in modelling the real time component of a task in the analysis phase of the software development life cycle. This will facilitate identifying alternative actions to be taken once a task has been identified as failing to meet its real time constraints. This set of behaviour actions can range from logging an error to starting an alternate task. Identifying these constraints in the analysis phase can assist in identifying bottlenecks and better distributing work load between agents. Our approach highlights a higher level of proposed behavioural tasks/ goals to be taken in case the task fails to meet its real time constraints, as identifying the problem is the first step towards fixing or avoiding it.

3 IDENTIFIED REAL TIME CONSTRAINTS

The set of real-time modelling units we pursue should be sufficient to do the following: model tasks time constraints, identify when they are not met and model their behaviour at that time. If the task takes too long (exceeding the real time constraint) then the agent would identify that this task has failed and initiate a suitable behaviour to ensure that this failure does not propagate and cause one or more system goals to fail. We therefore propose two categories of modelling units: one group identifying if the constraint has been met or not and another group describing what actions/behaviour to be taken when a constraint is not met. We propose 2 units in the first category and 10 for the second category. The modelling units will describe if the constraint is soft/hard, its priority, its criticality, estimated duration, warning percentage, error percentage, tier number, periodic occurrences and real time order. Moreover, if the task should be retired or which alternative task should be tried. For a given RT constraint, there is no limit on the number of behavioural criteria imposed. E.g. when a task fails

the model should indicate all possible alternate tasks and arrange them according to a priority sequence. The developer can identify the task priority sequencing during analysis. These identified twelve units are not exhaustive. The developer can always add any new constraints and their graphical representation to the diagrams. The constraints set is summarised below with Identifying or Behaviour indicating the category it belongs to and then a brief explanation of the constraint and the symbol to represent it as follows:

- 1- Identify if an RT constraint exist at all, then the next 11 constraints can be used and the RT constraint presence is marked using a table symbol . Other constraints can be marked on top of this.
- 2- Identify if the constraint is a Soft  or Hard  constraint is identified. A hard RT constraint enforces that the task must complete within the specified time frame and if not is unacceptable or of no value. The value of a task with a soft RT constraint declines steadily after the deadline expires. Tasks completed after their respective soft RT deadlines have less value than those whose deadlines have not yet expired (Vahid 2010).
- 3- Identify Constraint Priority . This is the importance of the task to be completed, the lower the number the higher the priority i.e. P1 is the highest priority task which should be completed first, if at all possible.
- 4- Identify constraint Criticality . This is an indication of how critical a task is i.e. the effect a failure of this task would have on the whole system. If a highly critical task fails to meet its real time requirement, the criticality level is directly related to the priority level but they do not have to be equal. As tasks can have a high priority level, it's important to complete on time. But if it fails, the system in total might not be affected. While in other cases a task failure can cause the whole system to fail.
- 5- Identify Estimated Duration  Estimated Duration, to be used as a guideline to identify if the task has met its real time requirement or not.
- 6- Identify Warning Percentage , to be proactive in identifying the tasks that are unlikely to meet their real time constraint and help them fulfil these constraints by providing them with more resources, or starting the alternate task. (Brazier 2000).

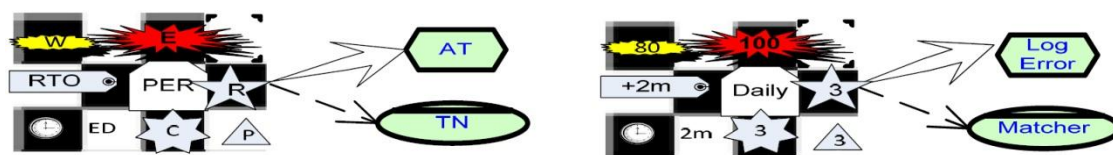







Figure 1: Representing the identified constraints in a table like diagram.


7- Identify Error percentage  is used to identify when a task has failed to meet its real time constraints. If the percentage is exceeded, the task has failed to meet its real time requirements. In most cases this would be 100% but this could vary. E.g. if there is a lag or lap time between 2 tasks i.e. the time between one task ending and another dependent task starting (Brazier 2000).

8- Identify Tier Number , this identifies the affected agent if the task fails to meet its real time constraints (Konrad 2006).

9- Identifying periodic occurrences (PER) , i.e. the schedule on which the task happens on (Konrad 2006).

10- Identify Real time order (RTO) is denoted by . This represents the time lag between instances of the same task or between one task and another dependent task starting (Konrad 2006). This helps in identifying the time buffer required to repeat a sequence of tasks before the system is affected.

11- Behaviour Retry attempts  is the number of times to Retry/restart the task before starting the Alternate task.

12- The Alternate task (if any) to start in case the initial task could not meet its real time constraint is denoted by . This emphasizes the robustness characteristic of the MAS and ensures the system reliability.

4 CALL MANAGEMENT

Beyond a one-to-one communication tool, telephony is 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, handling sales 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 specialized personnel and

training saves the client 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 (Ashamalla et al 2009). To validate the representational adequacy of the above constraints, we model a call centre support MAS. We propose using an intelligent distributed system (known as Multi Agent Systems) to assist in customer relationship management by routing calls and allocate calling duties to the most appropriate relationship manager (in terms of knowledge/skills and availability) to maximize effectiveness.

The goal of this system is to match the relationship managers (RM) (call centre workers receiving and making calls) with the end customers (EC) (the person on the other end of the phone line). ECs receive/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 RMs to increase call centre sales, customer satisfaction and profits (Ashamalla 2009). The system routes the calls to the appropriate RMs based on the EC and RM skills, background, demographics and performance. We will only present one agent (Outbound calling system) due to space requirements, as per below:-

The outbound calling system represents the agent responsible for dialling numbers, detecting call answers and routing calls to the available and appropriate RM, the outbound calling system tasks are:-

- 1- Dial number: The Calling system dials EC's numbers from the available pre loaded calling list.
- 2- Detect call answer: Detecting that a real person answered the call and not an answer machine or a busy line.
- 3- Start voice recording: Once an answer is detected the calling system needs to start voice recording.
- 4- Detect available RM: The calling system detects available RM's in order to route calls to them.

- 5- Route call to matched RM: Once an available RM is detected, the call should be routed to him/her.
- 6- Retrieve EC details: EC details are retrieved and displayed for the RM.
- 7- Retrieve script: The sale script and offers are retrieved and displayed on the screen, for the RM.
- 8- Detect Call outcome: RM logs the call outcome as sale, No sale, do not call or Call back.
- 9- Stop voice recording: Once the call has ended, voice recording for that call should be stopped.
- 10- Reroute call for call back: The system redials/recalls call back calls on the set date/time.
- 11- Reroute unanswered calls: Unanswered calls are logged as a "no answer" call, to be recalled later.

The first phase of developing the CMC MAS is articulating the requirements in order to undertake an appropriate agent oriented analysis. We perform RE activities informally with *i** (Yu 1995), beginning with stakeholder requirement analysis and rationale for the new system. We use the *i** (Yu 1995) modelling framework to represent MAS agents and the relationships between different agents. Our early requirement phase generates a high level description of system goals and roles expressed in the *i** model. In a MAS, agents depend on each other to achieve goals and perform tasks. The resultant *i** 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 (Ashamalla 2009). The choice of *i** as a modelling language is based on previous experience (Bresciani 2004) which has shown that *i** is a good language to express MAS requirements. In particular, the *i** 'actor' lends itself to readily model the actors and agents in a call management centre, our proposed system is composed of a number of Actors (Agents and Roles) (Beydoun et al 2009). OME3 tool was originally used to model the MAS call centre as part of our case study, however when we needed to represent the proposed real time modelling units we preferred using Microsoft Visio. As Visio stencil's provided a more efficient way to visually present our proposed real time modelling units. The values represent each individual task's real time criteria, e.g. The alternate task (AT) for the above task is to log an error, the affected agent (TN) is the Matcher agent which has the following soft constraints: the warning level is 80%, the tolerable error level is 100%, the Real time order (RTO) is +2 minutes

between this task and the successive task, the periodic occurrences (PER) is on a daily rate, the Retry (R) attempts is 3 times, the task estimated duration (ED) is 2 minutes, the task critical level (C) is 3 and its Priority level (R) is also 3.

The case study has found that the matcher and outbound calling system were relatively loaded with rt constraints (13 and 11 rt constraints respectively) making their work load in need to be redistributed, or broken down to multiple agents. While the rm agent has a relatively small number of rt constraints (4 rt constraints).with only one monitoring agent exits for the system, distributing the tasks among agents resulted in a more balanced model. This has identified alternative agents and tasks in the sr diagram, in case the task does not meet its real time constraint. It also highlights the affected agent where bottlenecks might occur and the effects on the system in general .i.e. If all affected agent (tn) links point to one agent. This indicates that the agent has a high probability of failing in case any of the linked tasks fail. The tasks could be the agents or another agent's task that have a direct effect on the agent. this model has led to 2 monitoring agents as not to have a single point of failure. We identified 77 agent tasks for the call centre mas. examining these tasks using the real time constraint set results in identifying 66 of the 77 mas tasks as potentially real time. we will only present the outbound calling agent's tasks due to space requirements.

5 CONCLUSIONS AND FUTURE WORK

This paper is part of our ongoing research aimed at identifying and modelling real time constraints in the early analysis stage of the development life cycle. It also helps in identifying future bottle necks that could arise as a result of overloading an agent with too many real time constraints in the early analysis phase. I.e. having all arrows point to a single agent indicates that this agent is a potential bottle neck and/or it is highly likely to fail. Our research aims to enhance the performance of agent systems to meet any real time constraints requirement. 12 modelling units to represent real time constraints have been identified based on academics and researches recommendations (Brazier 2000, Konrad 2006, Vahid 2010, Tran et al 2006) and others discussed in Section 2 and 3. We also developed a case study and some industry recommendations and we are currently validating these constraints using expert's

reviews and recommendations. The preliminary results are so far encouraging. This result is very dynamic in representing Real time constraints, allowing any newly identified constraints to be added to the model with the appropriate graphical representation. Our next step in this research is to propagate our real time constraints to agent goal models. Further case studies and modelling tools to further validate our results and research outcomes will be needed. Expert reviews in the call centre and MAS domains will be first contacted to review the outcome, before extending to another domain of collaborative e-Learning (Beydoun 2009).

REFERENCES

- Ashamalla, A., Beydoun, G. and Low, G. (2009). 'Agent Oriented Approach to a Call Management System', 18th International Conference on Information Systems Development (ISD 2009), Nanchang, China, September 16-19
- Beydoun, G., Low, G., Mouratidis H. and Henderson-Sellers B. (2009). 'A security-aware metamodel for multi-agent systems (MAS)', *Information and Software Technology*, 51(5): 832-845
- Beydoun, G., Gonzalez-Perez, C., et al. (2006). "Developing and Evaluating a Generic Metamodel for MAS Work Products". *Software Engineering for Multi-Agent Systems IV: Research Issues and Practical Applications*. A. Garcia, R. Choren, C. Lucena et al. Berlin, Springer-Verlag. LNCS 3914: 126-142.
- Beydoun, G. (2009). "Formal concept analysis for an e-learning semantic web". *Expert Systems with Applications* 36(8).
- Basra, R., Lu, K. and Skobelev, P. (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.
- Boehm, W. (1988). "A spiral model of software development and enhancement." *Computer* 21(5).
- Botti, V. and Julian, V. (2004). "Developing real-time multi-agent systems." *Integrated Computer-Aided Engineering* 11(2): 135-149.
- Brazier, F., Cornelissen, F., Jonker, C., and Treur, J. (2000). "Compositional Specification and Reuse of a Generic Cooperative Agent Model." *International Journal of Cooperative Information Systems* 9(3): 171.
- Brazier, F., Mobach, D., Overeinder, B. and Wijngaards, N. (2002). "Supporting Life Cycle Coordination in Open Agent Systems." IIDS Group, Department of Artificial Intelligence, Faculty of Sciences, Vrije Universiteit Amsterdam.
- Bresciani, P., Perini A., Giorgini P., Giunchiglia F. and Mylopoulos J. (2004). *Tropos: An agent-oriented software development methodology*. *Autonomous Agents and Multi-Agent Systems* 8(3): 203-236.
- Cabri, G., Leonardi, L. and Zambonelli, F. (2003). *BRAIN: A Framework for Flexible Role-Based Interactions in Multiagent Systems*. On The Move to Meaningful Internet Systems 2003: CoopIS, DOA, and ODBASE, Springer Berlin / Heidelberg. 2888: 145-161.
- Conitzer V.(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
- Danbing S., John L., Lui S. and Kang S. (2004). "Trade-Off Analysis of Real-Time Control Performance and Schedulability." *Real-Time Systems* 21(3): 199-217.
- Gokhale, S., Natarajan, B., Schmidt, C. and Cross, K. (2004). "Towards real-time fault-tolerant CORBA middleware." *Cluster Computing* 7(4): 331-346.
- Horkoff, J. (2007 Thursday 24 of May, 2007 23:13:15). "Visio." Retrieved 26/04/2011, 2011, from <http://tanne.informatik.rwth-aachen.de:7777/>.
- Konrad, J. (2006). *Model-driven development and analysis of high assurance systems*. Department of Computer Science. Michigan, Michigan State University. Doctor of Philosophy: 443.
- Kopetz, H. (2000). *Software engineering for real-time: A roadmap*. Proc. 22nd Int. Conf. Software Eng, Citeseer.
- Lu, Roman G. and Shourong (2006). "Modeling distributed real-time applications with specification PEARL " *Real-Time Systems* 35(3): 181-208.
- Micacchi, C. and Cohen, R. (2008). "A framework for simulating real-time multi-agent systems." *Knowledge and Information Systems* 17(2): 135-166.
- Neto, A., Sartori, F., Piccolo, F., Vitelli, R., De Tommasi, G., Zabeo, L., Barbalace, A., Fernandes, H., Valcárcel, F. and Batista, N. (2009). *MARTE: a Multi-Platform Real-Time Framework*. Proc. of the 16th IEEE NPSS Real-Time Conference, Beijing, China.
- Object Management Group (OMG). (2008). "UML profile for modeling and analysis of real-time and embedded systems (MARTE)" Retrieved 2-12-2010, from <http://www.omg.org/>.
- Sabour A., Faheem M. and Khalifa E. (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
- Sadrei, E., Aurum, A., Beydoun, G., and Paech, B. "A Field Study of the Requirements Engineering Practice in Australian Software Industry", *International Journal Requirements Engineering Journal* 12 (2007), pp. 145-162.
- Tran, QNN, Low, GC and Beydoun, G., "A Methodological Framework for Ontology Centric Agent Oriented Software Engineering", *International Journal of Computer Systems Science and Engineering*, 21, 117-132, 2006.
- Vahid, G. (2010). "Experience and challenges with UML-driven performance engineering of a Distributed Real-

- Time System." *Information and Software Technology* 52(6): 625-640.
- Vahid, G., Y. Labiche, et al. (2009). "A UML-based quantitative framework for early prediction of resource usage and load in distributed real-time systems." *Software and systems modeling* 8(2): 275-302.
- Westley, W. and George, N. (2004). "Finding and preventing run-time error handling mistakes." *SIGPLAN Not.* 39(10): 419-431.
- Yu, E. (1995). *Modelling strategic relationships for process reengineering*. Computer Science. Toronto, Canada, University of Toronto. Doctor of Philosophy: 131.
- Zambonelli F., Jennings N. and Wooldridge, M. (2001). "Organisational Abstractions for the Analysis and Design of Multi-agent Systems " *International Journal of Software Engineering & Knowledge Engineering* 11(3): 1.
- Zhang, C., Hammad, A. and Bahnassi, H. (2009). "Collaborative Multi-Agent Systems for Construction Equipment Based on Real Time Field Data Capturing." *Electronic Journal of Information Technology in Construction* 14.

