Computer Languages, Systems & Structures 50 (2017) 127–139

Contents lists available at ScienceDirect

Computer Languages, Systems & Structures

journal homepage: www.elsevier.com/locate/cl

Model driven approach for real-time requirement analysis of multi-agent systems

Amir Ashamalla[a], Ghassan Beydoun[a,*], Graham Low[b]

[a] School of Systems, Management and Leadership, University of Technology Sydney, Australia
[b] School of Information systems and Technology Management, University of New South Wales, Australia

ARTICLE INFO

Article history:
Received 24 June 2016
Revised 12 April 2017
Accepted 31 May 2017
Available online 22 June 2017

Keywords:
Requirement Modelling
Real Time
Multi Agent Systems
Metamodelling

ABSTRACT

Software systems can fail when requirement constraints are overlooked or violated. With the increased complexity of software systems, software development has become more reliant on model driven development. The paper advocates a model driven approach to ensure real-time requirement constraints are taken into account prior to the design of a multi-agent system (MAS). The paper presents the synthesis of a real-time metamodel to support requirements analysis of a MAS. The metamodel describes a collection of modelling units and constraints that can be used to identify the real-time requirements of a multi-agent system during the analysis phase. The paper takes the view that the earlier you 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 an appropriate balance between competing time requirements will be achieved. The paper also presents a validation of the metamodel in a *Call Management* MAS application. This provides a preliminary evidence of the coverage and validity of the metamodel presented.

1. Introduction

Software modelling processes typically involve a number of phases including analysis, specification, design, implementation, and testing. Each phase creates its own model (system representation) and bring the software system closer to realisation. Collectively the modelling phases represent the system. The class of systems of interest in this research is agent-oriented systems or multi-agent systems. The 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, the agent goals and roles are further analysed to identify agent tasks and agent classes that are closer to the system implementation [19].

In multi-agent systems, 'agents' are at the centre of the modelling processes and these are again typically supported by their own methodologies, e.g. Tropos [13], GAIA [59], MOBMAS [53], BEAST [17], ICTAM [20], and SAVS [43]. Various methodologies focus on different technologies or phases e.g. Gaia methodology focusses on the analysis and design of agent-based systems [59] while TROPOS introduces techniques for validating early requirements via a model checking approach [24]. This research focusses on supporting the requirement analysis phase. More specifically, the focus is on supporting the analysis of real-time requirements for agent-oriented systems.

A principal aim for modelling real-time systems requirements is fulfiling time constraints [4,54]. Real-time systems are critical system, where server consequences would result if the timing responses are not satisfied [11]. When developing a

* Corresponding author.
E-mail addresses: beydoun@uow.edu.au, ghassan.beydoun@uts.edu.au (G. Beydoun), g.low@unsw.edu.au (G. Low).

http://dx.doi.org/10.1016/j.cl.2017.05.006
1477-8424/

model for real-time Multi-Agent Systems (RTMAS), the relative priority of a task is often important as well as the actual task deadline. A real-time agent is defined as "an agent with temporal restrictions in some of its responsibilities or tasks" [10]. A real-time agent is an agent which achieves a maximal set of high priority goals by their deadlines [55]. 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. i.e. when a floor button is pressed, the elevator doors do not immediately close, a minimum delay is actually required. There is also a maximum time for the doors to close. When this maximum 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 [63]. After the doors close, the elevator has a minimum and maximum time to start moving, i.e., the elevator does not move up/down straight after the doors are closed. It waits until the minimum time elapses, and then begins to move. 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 overall. Currently, agent methods do not easily enable analysts to make these distinctions, let alone identify real-time tasks in the midst of the requirement analysis and elicitation activities. This paper works towards filling this gap. 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.

The rest of the paper is organised as follows: Section 2 describes the background and related work of the research focussing on current approaches in developing real-time MAS. Section 3 presents our RT constructs in the form of a set of reusable modelling units that can be applied during the requirement analysis phase in developing a MAS. The section also presents how their use can be interleaved with typical agent based analysis activities. Section 4 presents a validation of the modelling units in a *call management* application. Section 5 concludes with a discussion of limitations and future extension possibilities of this research.

2. Background and related work

A single agent is a software component capable of autonomous actions in an environment. Agents sense their environment and respond accordingly. Agents typically interact and cooperate to meet their goals, which need to be aligned with the system's overall requirements [2,6,7]. They are designed to meet local objectives as part of the overall objectives of the distributed system. 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 [56]. MAS are often employed when a centralised system solution is not feasible such as coordinating manufacturing [35,62], network management (Moon [41]), and electrical load balancing [14]. Time awareness is part of situatedness and is designed into agents. Agents observe time in their interactions and decision-making. The final outcome of their decision making process is partially dependent on time [50].

Modelling agent real-time interactions has gained focus in the last ten years [40]. 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 [1,3]. Agents (and tasks) in these systems might not always be aware of other agent's (and tasks) availability and response times. 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 [5] uses real-time attributes of messages and actions taken by other trains to avoid collision. In other applications, e.g. search and rescue tasks [39], real-time aspects of actions are used for avoiding obstacles in rescuing victims in real-time target tracking [42]. These and other related efforts have illustrate that agents can indeed efficiently interact in real-time to re-plan/reschedule the required tasks in a way that fits unexpected changes in the environment. A gap remains between modelling the MAS requirements and the realisation of the real-time software components required to operationalise real-time constraints of MAS tasks. This research seeks to contribute to bridging this gap. Specifically, it aims at providing a requirements modelling framework to facilitate identifying a sufficient set of activities to identify when a task have failed to meet its real-time constraints.

The framework will ensure that tasks that have failed can be distinguished from those that are late but still likely to succeed. The framework will be domain independent. It will depend 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 is considered failing when a real-time constraint applies. Receiving the right answer too late becomes the wrong answer [25]. Runtime errors and exception handling in the development phase typically require a different set of tasks to be initiated when an error occurs [58]. 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. This is different from fault tolerance: *"the application service must continue even if parts of the control system have failed"* [36]. Fault tolerance focusses on the behaviour of the task after reporting a failure in order to start an alternate task to fulfil the application goals. This research is more concerned with synthesising a reliable and precise analysis process to ensure that the system modeller captures the real-time constraints and the concomitant agent's behaviour.