The Journal of Systems and Software 117 (2016) 317–333

Contents lists available at ScienceDirect

The Journal of Systems and Software

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

An approach to modeling and developing teleo-reactive systems considering timing constraints

Pedro Sánchez*, Bárbara Álvarez, José Miguel Morales, Diego Alonso, Andrés Iborra

Systems and Electronic Engineering Division (DSIE), Universidad Politécnica de Cartagena Campus Muralla del Mar s/n, Cartagena, Spain

ARTICLE INFO

Article history:
Received 2 November 2015
Revised 21 March 2016
Accepted 24 March 2016
Available online 2 April 2016

Keywords:
Teleo-reactive
Timing constraints
TRiStar
Requirements engineering

ABSTRACT

Context: TeleoR is an extension and implementation of teleo-reactive (TR) language for defining the behavior of reactive systems when the consideration of timing constraints is a matter of interest.
Objective: This paper analyzes how to consider real-time constraints when a TR approach is followed from modeling to implementation.
Method: After carrying out a study of the type of timing constraints from the TR perspective, the possibility of using TeleoR for incorporating such constraints was considered. Some extensions on TRiStar notation were then made to represent temporal requirements. A drone-based case study was carried out to demonstrate the usefulness of this approach. Finally, a survey was conducted to validate the approach.
Results: TeleoR can, to a great extent, support the kind of real-time constraints required for developing real-time systems, offering a direct solution to five of the eight temporal requirements identified, which can be implemented using the basic features of the language.
Conclusions: Considering real-time requirements should be part of the specification of reactive systems implemented when using the TR approach and should be supported by the implementation platform. In this regard, TeleoR offers reasonable possibilities that should be extended by taking into account the limitations identified here.

© 2016 Elsevier Inc. All rights reserved.

1. Introduction

Reactive systems are real-time systems that interact with the physical world and react to stimuli from the environment within finite and specified time intervals. The teleo-reactive paradigm (henceforth TR) designed by Prof. Nils Nilsson (Nilsson, 1994, Nilsson, 2001) provides a programming model based on high-level agents to develop reactive systems. This model is particularly focused on the robotic vehicle domain. The TR approach offers engineers a goal-oriented formalism for the development of reactive systems, allowing them to define system behavior while taking into account goals and changes in the state of the environment. This approach has been expanded with new capabilities by numerous authors in order to model systems in different domains. In Morales et al. (2014) the authors present a systematic literature review of the TR paradigm in which high-quality research related to its use is identified.

In Nilsson (1994) Nilsson introduces the notion of the TR program, consisting of an ordered set of production rules, as an agent control sequence that directs agents toward goals while taking into account changes in the environment. This list of rules is continuously scanned from the first rule whose condition is satisfied, leading to the execution of the corresponding action. In other words, a TR program, as a reactive system, is a set of rules that continuously observes the environment and takes decisions based on these observations. The state of this environment is dynamic and can be changed when actions are triggered. The main advantage of this approach is its robustness, due to the continuous computation of the conditions on which agent actions are based. In short, the TR paradigm offers a useful approach for developing systems when a goal-oriented specification is employed (see (Rajan et al., 2010, Gubisch et al., 2008, Broda et al., 2000) for further details of this approach).

The original approach defined by Nilsson does not allow the consideration of timing requirements. When the TR approach is used to develop a reactive system, the question arises: *how can we incorporate timing requirements into the system?* The timing requirements, which are part of the system specification, will imply a deterministic behavior during their execution. The different threads will have to meet their deadlines, specified by time intervals.

* Corresponding author. Tel.: +34968326460.
E-mail addresses: pedro.sanchez@upct.es (P. Sánchez), balvarez@upct.es (B. Álvarez), josemiguel.morales@upct.es (J.M. Morales), diego.alonso@upct.es (D. Alonso), andres.iborra@upct.es (A. Iborra).

http://dx.doi.org/10.1016/j.jss.2016.03.064
0164-1212/© 2016 Elsevier Inc. All rights reserved.

This paper contributes to the state of the art by providing: (1) a detailed study of types of timing constraints and their formal representation considered when specifying reactive systems using the TR paradigm; (2) a study on how TeleoR (the most well-known TR programming language) allows the considered types of requirements to be incorporated and a solution for those requirements which do not have a direct representation in the language; (3) a goal-oriented graphical notation (called TRiStar+) to capture timing requirements; (4) a preliminary method for developing reactive systems with timing constraints; and (5) validation by a complex example and a satisfaction survey on the usability of the notation.

The paper is organized as follows. Section 2 discusses the related work. Section 3 summarizes the original TR approach as defined by Nilsson. Section 4 summarizes the main contributions of TeleoR, an extension of the approach proposed by Clark. In Section 5, the correspondence between timing requirements and TR requirements is pointed out. In Section 6, the ability of TeleoR to incorporate the above requirements is described. In Section 7, the TRiStar+ notation for specifying timing requirements is given. In order to validate the proposal, a practical example is described in Section 8. Section 9 discusses the results obtained in the evaluation of TRiStar+ notation and finally our conclusions are presented.

2. Related and previous work

As stated before, the TR approach as defined by Nilsson does not allow the consideration of timing requirements. Among the different extensions of the TR approach, Prof. Keith Clark has made an important proposal, TeleoR (Clark and Robinson, 2014), which provides a real implementation of the system that can be compiled and executed on different platforms. In addition, this initiative extends the initial approach with first-level syntactic constructors to consider timing requirements. In this paper we will therefore consider only such an implementation. Although TeleoR allows reactive systems specifications to be compiled, a detailed study of how the TR paradigm allows timing requirements to be modeled has not been published to date.

There are many other approaches for designing agent-based systems similar to the teleo-reactive approach, amongst which it is worth highlighting the Belief-Desire-Intention (BDI) (Rao and Georgeff, 1995) approach. In the BDI architecture, an agent carries out intentions according to its current belief in order to achieve its desires. Intentions are revised every time beliefs or desires change as a consequence of changes in the environment. BDI Desires are considered distinct from goals, as at any given moment there exists the possibility of mutually incompatible desires. Though both the BDI and teleo-reactive approaches develop the same kind of applications, they can nevertheless be combined to overcome their limitations when used in isolation. In this vein, the work by Coffey and Clark (2006) proposes a BDI-style cognitive layer with a graded behavioral layer composed of hierarchies of teleo-reactive programs, so that the reactive behavior of a BDI system can be achieved with teleo-reactive programs.

There are some other alternatives to specifying the behavior of agent-based systems. TROPOS (Castro et al., 2002, Bresciani et al., 2004) is an agent-oriented software engineering methodology that covers the whole software development process and has been used to model and simulate Embedded Real-time Control Systems (Darragi et al., 2013, January). TROPOS adopts the i* modeling framework (Yu, 1997) and has the same limitations in specifying TR systems as those found in i* (see (Mouratidis and Giorgini, 2007) for a comprehensive description of these limitations). TROPOS addresses these problems by using UML diagrams which allow developers to continue the development process. In this context, to facilitate the specification of reactive systems and the exchange of specifications among different developers, the authors proposed an i* extension called TRiStar (Morales et al., 2015), a goal-oriented modeling language suitable for early phases of system modeling in order to understand the problem domain). TRiStar makes it possible to represent graphically a platform-independent TR specification by focusing on goals, conditions, actions, etc. The TRiStar+ approach introduced in this article constitutes a further step in this direction as it allows the behavior of TR systems to be specified considering timing constraints. For understanding the contribution of the article an introduction to the TR approach is first needed.

3. The teleo-reactive paradigm

This section is devoted to introducing the TR paradigm. Further information on the paradigm can be found in the literature (Morales et al., 2014). A TR sequence is an agent control program that directs the agent towards a goal (hence *teleo*) taking into account changes in the state of the environment (hence *reactive*) (Nilsson, 1994). *Teleo* means to bring to an end or to achieve a goal. *Reactive* implies continuous sensing of the environment, i.e. the effects of task actions or changes brought about by external events via sensorial data and communication with other agents. This monitoring allows quick reactions to new information so as to change or modify a task action, or to interrupt a sub-task. TR programs can be seen as a set of prioritized condition/action rules that trigger actions whose continuous execution leads the system to satisfy a goal. Table 1 shows the TR program structure consisting of an ordered set of rules. For each rule, K_i are the conditions either in sensory inputs or in a model of the environment and a_i are the actions that can change the model.

The list of rules is continuously evaluated, starting from the top and when a rule condition is true its corresponding action is executed, so that the system can interact with its environment. Actions can be either *durative* or *discrete* actions. A durative action is one that continues indefinitely while its condition is true. For example, in the robotic vehicle domain, a mobile robot is capable of executing the durative action “move forward”, which makes the robot move forward indefinitely. A discrete action is one that finishes after a short period of time and cannot be interrupted, e.g. “open gripper”.

Table 2 shows the TR program of a robot that moves forward until it detects an obstacle (sensor input), and then “rotates” until its path is clear. Rule #1 has higher priority than rule #2, so when an obstacle is detected the durative action “move forward” is interrupted. Once the condition of the rule #1 becomes false, because the obstacle is no longer in front of the robot, rule#2 continues the “move forward” action.

An interesting capability in defining a TR program is the possibility of including hierarchies to improve many properties such as modularity, reuse or tests. In a hierarchical TR program, each action (for example, “search object”) can also be another TR program

Table 1
TR program structure.

Priority	Rule (condition → action)
The highest priority	$K_1 \rightarrow a_1$ $K_2 \rightarrow a_2$ …
The lowest priority	$K_m \rightarrow a_n$

Table 2
A simple TR program for a moving robot.

Id	Rule (condition → action)
rule #1:	Obstacle_detected→ rotate
rule #2:	True → move forward

Download English Version:

https://daneshyari.com/en/article/6885509

Download Persian Version:

https://daneshyari.com/article/6885509

Daneshyari.com