Science of Computer Programming 111 (2015) 339–362

Contents lists available at ScienceDirect

Science of Computer Programming

www.elsevier.com/locate/scico

Implementability of requirements in the four-variable model

Lucian M. Patcas *, Mark Lawford, Tom Maibaum

Department of Computing and Software, McMaster University, 1280 Main St. W., Hamilton, ON L8S4K1, Canada

ARTICLE INFO

Article history:
Received 8 June 2014
Received in revised form 8 March 2015
Accepted 14 May 2015
Available online 27 May 2015

Keywords:
Safety-critical
Four-variable model
Implementability of requirements
Tolerances on requirements
Demonic calculus of relations

ABSTRACT

Many safety-critical computer systems are required to monitor and control physical processes. The four-variable model, which has been used successfully in industry for almost four decades, helps to clarify the behaviors of, and the boundaries between the physical processes, input/output devices, and software. In this model, the acceptable behaviors of the software are constrained by the physical environment, system requirements, and input/output devices. If acceptable software behaviors are possible, then the system requirements are said to be implementable with respect to these constraints. The only acceptability condition proposed in the literature deems as acceptable software behaviors that can lead to undesirable system behaviors, in particular, nondeterministic system behaviors that for the same input sometimes do not produce any results and some other times produce expected results. In this sense, the acceptability condition can be seen as angelic. In this paper we strengthen the acceptability condition using the demonic calculus of relations such that no undesirable system or software behaviors are allowed and prove a necessary and sufficient implementability condition for the system requirements. As a byproduct, we also obtain a mathematical characterization of the least restrictive software specification, which, for all intents and purposes, can play the role of the software requirements.

1. Introduction

Many safety-critical systems in application domains such as aerospace, automotive, medical devices, or nuclear power generation are required to monitor and control physical processes. An example is the shutdown system of a nuclear reactor which monitors the temperature and pressure inside the reactor and commands the reactor to enter a shutdown state whenever abnormal temperature and pressure values have been detected. Such systems are usually implemented using digital computers that are embedded into the larger system of the application and are interfaced with the physical environment using input devices (e.g., sensors, analog-to-digital converters) and output devices (e.g., digital-to-analog converters, actuators). Based on the measured values of the physical parameters of interest, the software commands the actuators to apply stimuli to the environment with the purpose of maintaining certain properties in the environment.

Due to their safety-critical nature, getting these systems right is extremely important. A challenging design task is to find the right combination of input devices, output devices, and software such that their integration produces a system that satisfies the requirements. Systems engineers are responsible for this task and, in particular, for choosing the input and output devices. Software engineers must then determine the software part of the system so that the system requirements

* Corresponding author.
E-mail addresses: patcaslm@mcmaster.ca (L.M. Patcas), lawford@mcmaster.ca (M. Lawford), maibaum@mcmaster.ca (T. Maibaum).

http://dx.doi.org/10.1016/j.scico.2015.05.007
0167-6423/

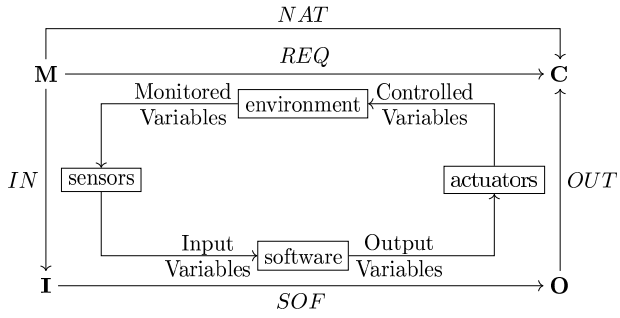

Fig. 1. The four-variable model.

are satisfied. Considering that changes in the specifications of the system requirements and hardware interfaces often arise during the system's development life cycle, the process mentioned above becomes repetitive and thus even more demanding [1], [2, Section 2.6.3]. What if no software can satisfy the constraints imposed by the system requirements and chosen hardware interfaces? Time and resources will be spent trying to develop and verify repeatedly a system that can never satisfy the requirements.

Hence, we ask the following question: is the software part of the system possible at all given a particular choice of hardware interfacing between the system and the physical environment? A positive answer to this question would allow software engineers to proceed with a software design having the confidence that their efforts are not destined to fail from the start. In this case, the requirements of the system are said to be *implementable* with respect to the physical environment and chosen input/output devices, while the software is called *acceptable*. In the case of a negative answer, the next step would be for the systems engineers to understand why that is the case and determine the necessary changes to the specifications of the input and output devices, and possibly to the specification of the system requirements, in order for the software part of the system to become possible. Such a bidirectional interaction between systems engineering and software engineering is stressed in [2, Section 1.2] as being essential in producing dependable software-controlled systems.

In this paper we prove a necessary and sufficient implementability condition for requirements in the four-variable model proposed by Parnas and Madey [3]. This model, depicted in Fig. 1 and described in Section 2, has been used successfully in the development of safety-critical systems in industry and helps to clarify the behaviors of, and the boundaries between, the environment, sensors, actuators, and software. To be implementable, the system requirements must be feasible with respect to the environment (i.e., should specify only behaviors that obey the environmental constraints) and acceptable software behaviors must be possible given the chosen input/output devices. In Section 2, we discuss why the feasibility and acceptability conditions given in Parnas and Madey [3], which may be seen as angelic, are too weak and allow undesirable system and software behaviors. In Section 3 we introduce the demonic calculus of relations [4–6], which will be used to strengthen these conditions in Section 4. Using the strengthened feasibility and acceptability conditions, we will then give a necessary and sufficient implementability condition for the system requirements, along with a mathematical characterization of the software requirements. In Section 5 we describe a detailed analysis of the implementability of the requirements for a pressure sensor trip computer, a subsystem in the shutdown system of a nuclear power plant. This analysis also demonstrates the usefulness of the implementability conditions as rigorous and systematic guiding tools in determining the tolerances needed on the requirements of the pressure sensor trip computer.

This paper is an extended version of a previous paper by us [7]. Sections 2, 3 and 4 give more details, examples, and proofs; in particular, a more thorough comparison between demonic and angelic semantics is given. Section 5 is completely new.

2. The four-variable model

The model was used as early as 1978 as part of the Software Cost Reduction (SCR) program of the Naval Research Laboratory for specifying the flight software of the U.S. Navy's A-7 aircraft [8]. The ideas from SCR were later extended into the Consortium of Requirements Engineering (CoRE) methodology, which was used for specifying the avionics system of the C-130J military aircraft in the 1980s [9]. Another significant example of a successful use of the four-variable model is the redesign of the software in the shutdown systems of the Darlington nuclear power plant in Ontario, Canada in the 1990s [10–12]. In 2009, the four-variable model was used extensively in the *Requirements Engineering Handbook* [13] that was put together at the request of the U.S. Federal Aviation Administration.

2.1. System requirements and environmental constraints

In the four-variable model, *REQ* models the *system requirements*. At the system requirements level, a system is seen as a black-box that relates physical quantities measured by the system, called *monitored variables*, to physical quantities controlled by the system, called *controlled variables*. For example, monitored variables might be the pressure and temperature inside a nuclear reactor while controlled variables might be visual and audible alarms, as well as the trip signal that initiates