Simulation Modelling Practice and Theory 78 (2017) 28–50

Contents lists available at ScienceDirect

Simulation Modelling Practice and Theory

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

Model-based design of supervisory controllers for baggage handling systems

L. Swartjes[a,b,*], D.A. van Beek[a], W.J. Fokkink[a], J.A.W.M. van Eekelen[b]

[a] Eindhoven University of Technology, Control Systems Technology Group, Postbus 513, 5600 MB, Eindhoven, The Netherlands
[b] Vanderlande Industries B.V., Postbus 18, 5460 AA, Veghel, The Netherlands

ARTICLE INFO

Article history:
Received 28 February 2017
Revised 30 July 2017
Accepted 15 August 2017
Available online 29 August 2017

Keywords:
Model-based design
Formal models
Controller
Supervisory control
Validation
Application
Implementation

ABSTRACT

The complexity of airport baggage handling systems in combination with the required high level of robustness makes designing supervisory controllers for these systems a challenging task.

We show how a state of the art, formal, model-based design framework has been successfully used for model-based design of supervisory controllers for an actual industrial baggage handling system, and for a real-time emulation model of an actual international airport.

The high level modeling elements of the applied CIF model-based design framework allow the modeler to concentrate on implementing the baggage handling system design requirements, instead of programming PLC code. It also allows a modular and hierarchical design of the supervisory controller, and provides flexibility in adapting and extending the model. Validation of the controller and the uncontrolled plant by means of simulation and visualization made it possible to catch all modeling errors, leading to very short modeling, testing and error correction iteration loops.

To the best of our knowledge, this is the first successful employment of formal, model-based design in the context of supervisory control for actual, industrial size baggage handling systems, that covers the entire development process from requirements up to and including validation, real-time PLC code generation and implementation.

We give an overview of the model-based design framework, discuss several modeling issues, and analyze the results of the industrial applications. We do not go into full technical detail, due to nondisclosure agreements, but tell the story, and give lessons learned that we consider useful for practitioners.

1. Introduction

1.1. Baggage handling systems

Airports use dedicated systems to transport baggage items from designated entry points, e.g., a check-in, to designated exit points, e.g., an aircraft. These systems are commonly referred to as baggage handling systems (BHSs). Typically, a BHS consists of many kilometers of conveyor belts to transport baggage items. In addition, it contains many specialized components to screen and (re)route items.

* Corresponding author at: Mechanical Engineering, De Zaale, 5600 MB Eindhoven, Eindhoven, The Netherlands.
E-mail address: l.swartjes@gmail.com (L. Swartjes).

http://dx.doi.org/10.1016/j.simpat.2017.08.005
1569-190X/

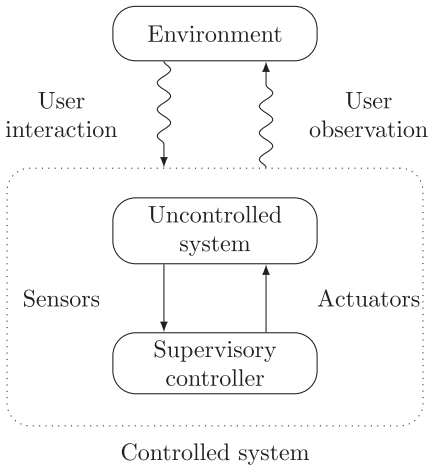

Fig. 1. Schematic representation of a system under supervision.

At the heart of such a BHS is a supervisory controller. The controller reads the observable state of the system via the current values of the sensor signals. Based on this state, and the controllers internal state, it calculates the new internal state and the required values of the actuators. Note that supervisory controllers mainly deal with logic control and sequence control, and are often represented as finite state automata [1]. A schematic overview of a system under supervision is depicted in Fig. 1.

1.2. Control system development process

The current approach to the design and creation of controllers for BHSs can be captured by the well-known V-model [2]. First, the requirements of the system are defined: what should the system do, and how should it complete this task. From these (behavioral) requirements, the specifications of the system are defined by the system architect and written down in a design document. Next, the software engineer encodes the supervisory controller based on the design document. Rigorous testing is employed during development and commissioning of the system. If all tests succeed, the controller is accepted.

In general, errors are found during tests. To resolve these errors, the cause of the error must be found. However, in many cases, finding the cause is difficult and non-trivial. As the controller is based on the software engineer's interpretation of the design document, the specification may have been wrong, the specification may be wrongly interpreted, or the controller may contain a coding error. Therefore, many lengthy design cycles are needed before a controller is accepted.

It may even occur that requirements are forgotten, i.e., not implemented, or that errors remain unnoticed during tests. This may lead to disastrous results when the system goes into operation. A well-known example is Denver International Airport [3,4], where a faulty (software) design resulted in large delays and huge financial losses. More recently, in June 2017, the Heathrow Airport baggage system suffered a failure, causing many passengers to fly without their luggage. This happened only three weeks after a catastrophic IT system failure at British Airways caused a full day cancellation of all flights from Heathrow and Gatwick.

1.3. Challenges in baggage handling control system design

To reduce the occurrence of errors, the building process can be partly automated by means to standardization [5]: standardized components can be associated with standardized code fragments with proven functionality and quality. Nevertheless, creating a proper controller remains challenging and requires a lot of effort. The reason for this is fourfold:

First, conformance of the controller with respect to the design requirements cannot be assessed before an actual implementation of the controller is built, and testing can commence by means of, for instance, emulation. In other words, the quality [6,7] of the controller design remains unknown until it is actually built.

Second, there are many client-specific requirements, as each BHS is tailored to a specific airport. The implementation of these client-specific requirements induces a lot of additional work, as these components are not standardized. As a result, code fragments influenced by these requirements must be (re)created from scratch. In other words, there is little to no flexibility [8,9] in the current controller design.

Third, transportation of baggage of all different kinds and sizes inevitably lead to errors. Such errors need to be efficiently and reliably handled. Operator intervention is often necessary, which leads to stringent safety requirements on equipment and control systems. Hardware failures on all levels must be dealt with predictably. Low level sensor or actuator failures may propagate through all control levels, and may lead to rerouting of baggage on the highest level.