Information and Software Technology 57 (2015) 169–185

Contents lists available at ScienceDirect

Information and Software Technology

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

# Testing robot controllers using constraint programming and continuous integration

Morten Mossige [a,b,c,*], Arnaud Gotlieb [b,1], Hein Meling [c,2]

[a] ABB Robotics, 4349 Bryne, Norway
[b] Simula Research Laboratory, Lysaker, Norway
[c] University of Stavanger, 4036 Stavanger, Norway

ARTICLE INFO

*Article history:*
Received 24 March 2014
Received in revised form 19 September 2014
Accepted 19 September 2014
Available online 2 October 2014

*Keywords:*
Constraint programming
Continuous integration
Robotized painting
Software testing
Distributed real time systems
Agile development

ABSTRACT

*Context:* Testing complex industrial robots (CIRs) requires testing several interacting control systems. This is challenging, especially for robots performing process-intensive tasks such as painting or gluing, since their dedicated process control systems can be loosely coupled with the robot's motion control.
*Objective:* Current practices for validating CIRs involve manual test case design and execution. To reduce testing costs and improve quality assurance, a trend is to automate the generation of test cases. Our work aims to define a cost-effective automated testing technique to validate CIR control systems in an industrial context.
*Method:* This paper reports on a methodology, developed at ABB Robotics in collaboration with SIMULA, for the fully automated testing of CIRs control systems. Our approach draws on continuous integration principles and well-established constraint-based testing techniques. It is based on a novel constraint-based model for automatically generating test sequences where test sequences are both *generated* and *executed* as part of a continuous integration process.
*Results:* By performing a detailed analysis of experimental results over a simplified version of our constraint model, we determine the most appropriate parameterization of the operational version of the constraint model. This version is now being deployed at ABB Robotics's CIR testing facilities and used on a permanent basis. This paper presents the empirical results obtained when automatically generating test sequences for CIRs at ABB Robotics. In a real industrial setting, the results show that our methodology is not only able to detect reintroduced known faults, but also to spot completely new faults.
*Conclusion:* Our empirical evaluation shows that constraint-based testing is appropriate for automatically generating test sequences for CIRs and can be faithfully deployed in an industrial context.


## 1. Introduction

A complex industrial robot (CIR) is defined as a classical industrial robot with an additional control system attached to perform a given process. This additional control system is typical responsible for controlling the process, which is typically painting, gluing, welding, and so forth.

Developing reliable software for CIRs is a complex task, because typical CIRs are comprised of numerous components, including control computers, microprocessors, field-programmable gate arrays, and sensor devices. These components usually interact through a range of different interconnection technologies, for example, Ethernet and dual port RAM, depending on delay and latency requirements on the communication. As the complexity of robot control systems continues to grow, the development and validation of software for CIRs is becoming increasingly difficult.

The problem is even worse for robots performing process-intensive tasks such as painting, gluing, or sealing, since their dedicated process control systems can be loosely coupled with the motion control system. In particular, a key feature of robotized painting is the ability to precisely activate the process equipment along a robot's programmed path. However, many of the processes involved in robotized painting are relatively slow compared to the process of moving the mechanical robot. Consequently, advanced computation-based techniques have been set up to take advantage of knowledge of the slower physical processes to

* Corresponding author at: University of Stavanger, 4036 Stavanger, Norway. Tel.: +47 514 89 247.
*E-mail addresses:* morten.mossige@uis.no (M. Mossige), arnaud@simula.no (A. Gotlieb), hein.meling@uis.no (H. Meling).
[1] Tel.: +47 406 26 077.
[2] Tel.: +47 518 32 080.

http://dx.doi.org/10.1016/j.infsof.2014.09.009
0950-5849/

compensate for these latencies. Validation of such a paint control system, called an Integrated Painting System (IPS), is therefore challenging. Current testing practices to reduce the number of software faults apply techniques such as the manual design of unit and integration testing, where both the test inputs and expected output are defined by validation engineers. Testing the IPS requires access to the physical layer to activate many of the painting robot's features. Much of the testing is based on running the full-scale system with a moving robot and measuring IPS outputs with instruments such as an oscilloscope. This results in long round-trip times and little automation. In addition, many of the tests produced for one configuration of the IPS cannot easily be reused to test another configuration, since manual test configuration is required. These techniques are labor intensive and error prone. Consequently, software faults may still be detected late in the IPS design process, often close to release date, leading to increased validation costs.

In this paper, we report on a methodology to fully automate the testing of ABB's CIR control systems. The work builds on initial ideas sketched in a poster presentation [1]. Our approach draws on continuous integration principles and well-established constraint-based testing techniques. It is based on an original constraint-based model for automatically generating test sequences that are both *generated* and *executed* as part of a continuous integration process. By performing a detailed analysis of experimental results over a simplified version of our constraint model, we determine the most appropriate parameterization of the operational version of the constraint model. This version is now deployed at ABB Robotics's CIR testing facilities and used on a permanent basis. This paper presents the empirical results obtained when automatically generating test sequences for CIRs at ABB Robotics. In a real industrial setting, the results show that our methodology is not only able to detect reintroduced known faults, but also to spot completely new faults. Our empirical evaluation shows that constraint-based testing is appropriate to automatically generate test sequences for CIRs and can be faithfully deployed in an industrial context.

### 1.1. Contributions

The contributions of the paper can be summarized as follows:

1. Our testing methodology introduces a new constraint-based mathematical model focusing on IPS timing aspects. The constraints are used to describe both normal behaviors of the IPS, as well as abnormal behaviors, so that it is possible to target error states when generating test cases. The model is generic and expressed using simple mathematical notions, which makes it reusable in other contexts.
2. A full-scale implementation of the model is presented with constraint programming tools [2]. The paper presents how the model is integrated in a live industrial setting to test the IPS. To the best of our knowledge, this is the first time a constraint model and its solving processes are used in a continuous integration environment to test complex control systems.
3. An empirical evaluation is conducted to analyze the model's deployment. During this evaluation, reinserted old, historical faults are found by this new approach, as well as new faults. Comparing this constraint-based approach with current IPS testing practices reveals that the time from a source code change to the time that a relevant test is executed is dramatically reduced.

### 1.2. Organization

We start by providing background information and presenting related work in Section 2. In Section 3 we introduce robotized painting. We describe some of the design choices made when developing ABB's paint control system and how these affect testing of the system. We present how the IPS is currently tested in Section 4. We describe the paint control systems' mathematical properties in Section 5 and, based on these properties, we present the constraints used as a basis for generating a model that can be used for test case generation in Section 6. In Section 7, we describe how the model is implemented and how it is integrated with a continuous integration system. We then present the results this new test strategy in Section 8. We present a thoroughly experimental evaluation of the model recommendations of how to use the model. In Section 9, we suggest ideas for improvement and further work.

## 2. Background and related work

The methodology proposed in this paper is tightly coupled with continuous integration and model-based testing (MBT). This section recalls the basics of continuous integration and gives a brief overview of the most recent advances in the field by looking at how continuous integration influences verification and validation activities. This section also reviews usage of MBT, with a particular focus on constraint programming in software testing.

### 2.1. Continuous integration

Continuous integration [3] is a software engineering practice aimed at uncovering software errors at an early stage of software development, to avoid problems during integration testing. Even if there is no general consensus of what continuous integration is exactly, a typical continuous integration infrastructure includes source control repository tools, automated build, build servers,[3] and test servers. Fitzgerald and Stol [4] describe continuous integration as "a process which is typically automatically triggered and comprises inter-connected steps such as compiling code, running unit and acceptance tests, validating code coverage, checking compliance with coding standards, and building deployment packages." There is therefore a common understanding that the time from a continuous integration cycle being triggered to a developer receiving feedback should be as short as possible [5,6]. Therefore, one of the key ideas behind continuous integration is to build, integrate, and test the software as frequently as possible. Developers working under continuous integration are encouraged to submit small source code changes to the source code repository instead of waiting and occasionally submitting larger sets of changes.

If we consider test execution part of a continuous integration cycle, various testing activities could, in principle, be included. For example, automatic test case generation, test suite minimization, or prioritization [7–11] could be included to reduce the time needed to execute a test suite without reducing the quality of the overall test process. Interestingly, Hill et al. [12] report on the inclusion of system execution modeling tools to test distributed real-time systems as part of continuous integration. However, to the best our of knowledge, very few results evaluate the impact of including more testing activities in continuous integration. Our work, incorporating systematic automated test case generation methodology in continuous integration, is a first step toward more automation in the software validation of complex software control systems.

[3] A build server is a machine that fetches source code from the source control repository and performs building, testing, integration, and so forth. All steps are carried out completely automatically and typically triggered by a source code commit or a timer.