Available online at www.sciencedirect.com
ScienceDirect
Control Engineering Practice 15 (2007) 1403–1415
CONTROL ENGINEERING PRACTICE
www.elsevier.com/locate/conengprac

Improving automation software dependability: A role for formal methods?

Timothy L. Johnson*

GE Global Research, K-1, 5C30A, 1 Research Cir., Niskayuna, NY 12309, USA

Received 29 April 2005; accepted 4 July 2006
Available online 10 October 2006

Abstract

The growth of manufacturing control software from simple NC and PLC-based systems to concurrent networked systems incorporating PCs, PLCs, CNCs, and enterprise databases has created new challenges to the design, implementation, and maintenance of safe and dependable manufacturing systems. Key milestones in this evolution, and the prospects for the use of formal verification methods in achieving enhanced dependability of future manufacturing software, are examined in this paper and presentation.

Keywords: System engineering; Reliability theory; Safety analysis; Automation; Programming theory

1. Introduction

In the US, the Denver International Airport is often used by the Federal Aviation Administration (FAA) as a test site for new technologies. A perennial dread of the air traveler is the baggage handling system: lost bags, delayed bags, and worst of all, bags transferred to the wrong airline, and ending up in remote places. So, the FAA and the Denver businesses and politicians decided that the brand new airport would be a wonderful place to showcase new baggage handling technology. The system requirements were duly prepared, the contract awarded, and millions of dollars committed to a network of computer-controlled conveyors that would whisk luggage immediately to its intended destination (deNeufville, 1994). But then came the control system. The initial indication that something was wrong occurred when the rest of the airport, and conveyors, were in place, but the software design had barely begun. The project became a public burden when it was over 2 years late on delivery (the rest of the airport could not be used without it). Finally, the time for initial testing arrived: The system could not do even the most basic luggage transport correctly. Patience wore thin. Political and business reputations were ruined. Finally, the system was scrapped and a "semi-automated" (viz., conventional) system was used instead! From the control engineer's perspective, the most serious consequence of this type of failure is that the public was left with the impression that automation itself was at fault, and not that (as was undoubtedly the case) the project was mis-managed. Dozens of airports around the country will now opt for less automated systems in favor of more automated ones, and control engineers will have less to do.

The Denver Airport baggage-handling example illustrates the public consequences of perceived lack of *automation software dependability*. As a whole, only about 30–40% of large software projects that are initiated will run to completion (Brooks, 1995), and this was one that did not. Even though the record in manufacturing systems—which are highly structured—is probably better than this average, it still could benefit from substantial improvement (Place and Kang, 1993—selected references from older literature have also been repeated here). Start-ups of new manufacturing and process plants are often notoriously delayed. And increasingly software development is at the heart of most of the problems. With the rapid decrease in cost, and even more rapid increase in the capabilities of computers over the last decade, the computing hardware components of automation have

*Tel.: +1 518 3875096; fax: +1 518 3876845.
E-mail address: johnsontl@research.ge.com.

0967-0661/$ - see front matter © 2006 Elsevier Ltd. All rights reserved.
doi:10.1016/j.conengprac.2006.07.005

become less costly, more versatile, and more reliable. So the drive to shift hardware functions into software has accelerated over the last decade. Manufacturing software itself has expanded from isolated, carefully designed PLC logic systems that operate for months without interruption, to PC-based platforms, where even in the absence of an application, the operating systems must be rebooted every few days!

Not only have manufacturing control applications become rapidly more complex, but also the expectations of timely response have grown increasingly more demanding. At the same time, other design requirements have also grown more demanding. Availability targets have expanded from 95% to 99.99% or higher in some applications (e.g., network broadcasting). The numbers of measurement and control points and size of control programs have exploded. Networks are a part of almost every system (Perrow, 1984). Enterprise integration, as well as sensor-level integration is expected.

In spite of the increasing level of dependence of manufacturers on automation software that is expected to be safe and reliable, very little rigorous statistical data concerning manufacturing software mishaps is available. The best publicly available data in the US appears to be in the area of Occupational Health and Safety incident investigations, and in documented court cases involving software failures. However, in the case of Occupational Health and Safety, many accident root causes have been attributed to process, sensing, or display irregularities—even when software is also involved. In court cases, e.g., those involving personal injury in manufacturing operations, the legal profession is frequently challenged to differentiate between error on the part of a software user, and errors in the software itself: until very recently, end responsibility for safety-related functions has often been delegated by the courts to users or operators of software, even in cases of software malfunction. The vast majority of unscheduled outages are "routine", and the appropriate unit or subunit is investigated, and then reset or restarted within a few minutes; nevertheless, part production runs below capacity during this time interval.

The advent of web-based and distributed software, often with multithreaded (concurrent) operation, and the contemplated use of wireless links for factory networks will create system-level fault modes of a complexity that could only be imagined a few years ago. In spite of this, it is not likely that computing progress will be reversed by these considerations. Instead, what may be required is a host of much more powerful verification and validation methods. The study of more powerful verification methods is bound to become more important as software becomes more complex. The purpose of this presentation is to review some of the fundamental factors underlying manufacturing software dependability, to survey the state of the art in current products and research related to verification, validation, and safety of such systems, and to provide a brief preview of some more recent research that shows promise in improving the quality of service (QOS) of manufacturing automation software. With an understanding of feedback processes and manufacturing system dynamics, control engineers and scientists are well qualified to play vital role in the future of dependable manufacturing systems.

2. The evolution of dependability in manufacturing control

Many of us are familiar with the development of manufacturing automation equipment—but have we ever thought about the evolution of test and verification for such equipment? The purpose of this brief sprint through history is to trace the growth of test and verification processes and issues that have accompanied the better-known improvements in manufacturing automation and computation. This provides a context in which to assess the growth of formal methods in this field.

The earliest machine tool control languages, such as APT, that were invented in the late 1940s (Alford and Sledge, 1976), were deliberately designed for ease of test. The core elements of the language were simple instructions (START, STOP, MOVE). But, even in these relatively simple cases, program verification could be difficult. The developer would be required to pre-compute coordinate transformations with a slide rule! Running them through the machine tool, determining if the part was right, and then modifying the program if necessary tested the early programs.

Data and logic of the programs were combined in the MOVE statement. The entire mechanical structure and coordinate system of the machine tool was presumed to be known to the user. The only transfer of control was via an unconditional GO TO. Instructions were executed according to a fixed, precisely timed clock cycle. No safety checking was performed, so the machining head could collide with any jig or guide, or indeed with a part of the machining table itself. Such machines were commonly found in a state of considerable damage after a few years of use! The machine and software could be verified, and even concurrently calibrated, by running a set of simple calibration routines and then gauging the resulting workpieces to compare them with their intended values. Today's machine tools, of course, support a much higher degree of complexity, and may include built in 3D simulators with collision detection capabilities (Fig. 1).

In the late 1950s and throughout the 1960s and 1970s, a complementary machine was born: the programmable logic controller (PLC). Such machines originally came about to replace racks of relays that had been developed in the 1920s to 1940s to govern the sequential control of complex operations such as telephone line switching, railroad signalling, and some early automation equipment. The development of the PLC was stimulated by several dependability problems with relays racks:

- Relays would fail mechanically after 30,000 or 40,000 operations.