

Journal of
System
Safety

eEdition

A publication of the System Safety Society

Home
Subscriptions & Memberships
Contact
About eJSS
System Safety Society

Vol. 47, No. 6 • November-December 2011

In the Spotlight

Download printable PDF of this page

Safety Implications of Software in Safety-Critical Devices

by Amber D.Schauf

Pages 1 | 2 | 3 | 4 | 5

Introduction

Software programs used in safety-critical devices are one of many concerns when designing a product to be used by the public. Corporations must consider the risks presented by the products they create, as well as the safety measures that must be taken to ensure the least amount of harm done due to faulty software design.

Multiple factors lead to software failure. Companies are under extreme pressure to be the first entity to get a functioning product on the market. In addition, the pace at which technology is created continues to increase at an exponential rate. Because traditional safety methods do not work in a fast-paced development and acquisition market, this results in critical errors in products. Examples of these errors can be found across all engineering disciplines. This paper will discuss three specific cases in which a software error was the culprit in a safety critical device.

The Electronics Race

There are many aspects of creating a product intended for mainstream usage — the main factors in product development are the costs associated with designing the product and the time necessary to create a prototype for mass production purposes. These are the concerns that every company in both the private and government sectors take into consideration before funneling funds into a project. If the basic needs are not met, it could mean great financial loss for that company.

There are multiple factors that lead to software failures. Companies are under extreme pressure to be the first entity to get a functioning product on the market. In addition, the pace at which technology is created continues to increase at an exponential rate. Because traditional safety methods do not work in a fast-paced development and acquisition market, this results in critical errors in products.

In addition to the design of the new product, there are also fees incurred through legal departments, the building of scaled prototypes, building real-sized test and the costs of testing those products. Contract bids are placed with companies to see who would like to buy the product. Consumer data-gathering studies are done to make the product more alluring to potential clients. Patents are acquired to protect designs from other companies trying to create the same product first. All of this is done without the guarantee that the creator will be successful in finding someone to purchase the product.

While safety is a highly regarded measure, companies are under great strain to beat the competition

System Safety
Engineering
Summer Class

Instructor:
Dr. Nancy Leveson

Date:
June 18 – 22, 2012

Location:
Seattle, Washington

For more information,
please go to:
<http://sunnyday.mit.edu/announce12.html>

President's Message

From the Executive Vice President

From the Editor's Desk

Outside the Lines

In the Spotlight:

The Use of Safety Cases in Certification and Regulation

Safety Implications of Software in Safety-Critical Devices

System Safety in Healthcare

Swiss Cheese Model for Investigating the Causes of Adverse Events

Announcements

Gains from Losses:

Facts, Fiction and Public Perception

Book Review:

Murder by Electrocutation, by David MacCollum

Unintended Consequences:

TWA Flight 800 Accident

Opinion (MacCollum)

Upcoming Conferences/Calls for Papers

Chapter News

Mark Your Calendar

About this Journal

Advertising in eJSS

Contact Us

Puzzle

to the mainstream marketplace. The supply and demand for newer, better consumer products, for a military weapon that is more efficient and more powerful than the enemy's, or for faster, more accurate healthcare equipment far exceeds the current wait period for necessary patents, rigid safety testing and complete data gathering. The economics of placing a competitive product on the market first creates an environment where shortcuts are taken at the expense of safety.

[next page »](#)

Copyright © 2011 by the System Safety Society. All rights reserved. The double-sigma logo is a trademark of the System Safety Society. Other corporate or trade names may be trademarks or registered trademarks of their respective holders.





Vol. 47, No. 6 • November-December 2011

In the Spotlight

Safety Implications of Software in Safety-Critical Devices

by Amber D.Schauf

Pages 1 | 2 | 3 | 4 | 5

The Birth of Electronics

Electricity is a fundamental force of nature that is present in every aspect of daily life. It is defined as a "form of kinetic or potential energy created by the free or controlled movement of charged particles such as electrons, positrons and ions" [Ref. 1]. It consists of two charged particles, positive and negative; the positive charges will repel other positive charges, but attract negative charges. The same occurs for negative charges; a negatively charged particle attracts only positively charged particles. Only one negative and one positive particle can attract to each other, creating a net charge of zero. This is explained by the Law of the Conservation of Energy: electric charge is conserved; it cannot be created nor destroyed, and the sum of all charge in a system remains unchanged.

1. Early Devices

The study of electricity has been validated back to the writings of Thales of Miletus in 600 BCE. It was found that if you rubbed fossilized amber, the substance would become charged. One device that came to the attention of archeologists in 1938 was the Battery of Babylon, also known as the Battery of Baghdad, which is located in Iraq. The device is a six-inch high clay pot that contained a copper cylinder soldered shut with a lead-tin alloy.

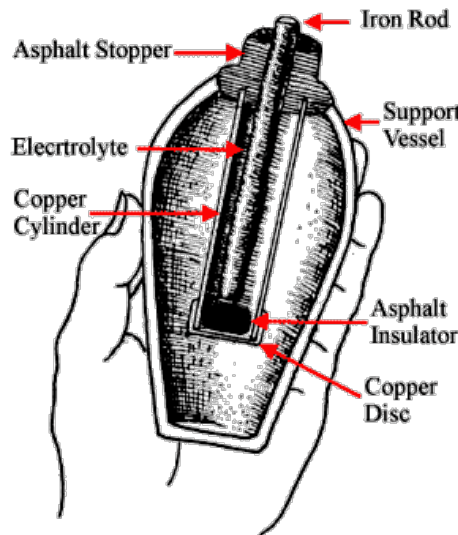


Figure 1 — Battery of Babylon [Ref. 2].

- President's Message
- From the Executive Vice President
- From the Editor's Desk
- Outside the Lines
- In the Spotlight:
 - The Use of Safety Cases in Certification and Regulation
 - Safety Implications of Software in Safety-Critical Devices
- System Safety in Healthcare
 - Swiss Cheese Model for Investigating the Causes of Adverse Events
- Announcements
- Gains from Losses:
 - Facts, Fiction and Public Perception
- Book Review:
 - Murder by Electrocutation, by David MacCollum
- Unintended Consequences:
 - TWA Flight 800 Accident
- Opinion (MacCollum)
- Upcoming Conferences/Calls for Papers
- Chapter News
- Mark Your Calendar
- About this Journal
- Advertising in eJSS
- Contact Us
- Puzzle

30 Years of System Safety Leadership

SAFETY SERVICE FOR MISSION & PRODUCT ASSURANCE

SAFETY SOFTWARE FOR HIGHLY INTEGRATED SYSTEMS

SAFETY TRAINING FOR PROFESSIONALS

ALD Software
 RAM Commander
 Ultimate Tool for Safety Assessment

FavoWeb
 World Standard in FRACAS

D-LCC
 Dynamic Cost Control

Contact Us Today for Free Web DEMO or Evaluation:
 USA Office: 5721 W Slauson Ave., Suite 140, Culver City, CA 90230
 Tel: 1-310-338-0990
 Fax: 1-310-338-0999
www.aldservice.com



When an acidic agent was poured into the pot, a chemical reaction occurred that produced a small, electric current. When several of these pots were placed in series, there was enough voltage created to allow for the electroplating of metals. Scientists also believe that this device could have been used for religious purposes.

Other than these two documented occurrences, there were no real advances until 1600 AD. An English scientist by the name of William Gilbert began documenting the electrical properties of several known substances and named the phenomenon electricity, derived from the Greek word for amber. It is through his work that he has been named the Father of Modern Electricity.

Following Gilbert's work, a series of electrical experiments took place. In 1660, a German scientist, Otto von Guericke, invented the first electrostatic generator, the "Elektrisiemaschine" [Ref. 2]. This device (Figure 2) could produce static electricity (high voltage, continuous current) via friction.



Figure 2 — Elektrisiemaschine [Ref. 3].

Many electrical studies ensued after the invention of the electrostatic generator. In 1745, Pieter van Musschenbroek developed the Leyden (Leiden) Jar, the first electrical capacitor.



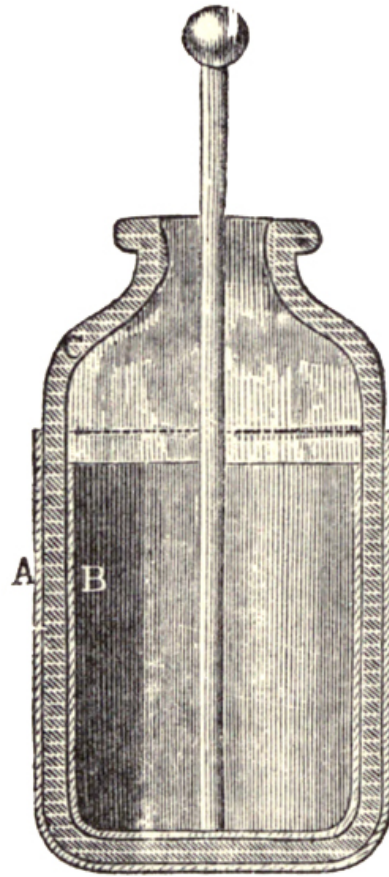


Figure 3 — Leyden Jar [Ref. 4].

The glass jar had a metal coating on both its inside and outside, but the surfaces did not come in contact with each other. An electrode was connected to the inner foil and protruded out of the opening of the jar. The electrostatic generator was then connected to the electrode and the outer foil was connected to a grounded source. The inner surface would store the positive charge, while the outer surface stored the negative charge; the net charge of the entire system was equal to zero. These two inventions provided scientists with instant electrical current. This allowed for controlled experimentation with electrical devices. Much of the work of Nikola Tesla, Thomas Edison and Benjamin Franklin came from working with these two devices, and has led us to the modern world in which we now live.

[« previous page | next page »](#)

Copyright © 2011 by the System Safety Society. All rights reserved. The double-sigma logo is a trademark of the System Safety Society. Other corporate or trade names may be trademarks or registered trademarks of their respective holders.





Vol. 47, No. 6 • November-December 2011



In the Spotlight

Safety Implications of Software in Safety-Critical Devices

by Amber D.Schauf

Pages 1 | 2 | 3 | 4 | 5

2. Modern Development

Turning the clocks ahead to present day, we find that electric devices that were large and cumbersome to carry around have now been reduced in size to devices not capable of being seen by the naked eye. These products now exist in the majority of what we touch and use, and are an integral part of our lives. Instead of paper and pen to write letters, electronic messages are sent through invisible waves to international destinations in the blink of an eye. Modern warfare is now conducted from behind a computer screen, as opposed to lines of soldiers packed shoulder to shoulder on a front line. Medicine is delivered at the click of a button, and heartbeats are measured by blipping lines on a screen. All of these marvels take place on a micro- and nano-scale field.

Transistors were first patented in 1925, but did not come to fruition until 1947 by a couple of American physicists working for AT&T's Bell Labs. The first silicon transistor was developed in 1954; in 1958, the first integrated circuit (in which a transistor is incorporated) was created and only two years later, the Metal Oxide-Semiconductor Field Effect Transistor (MOSFET) was invented.

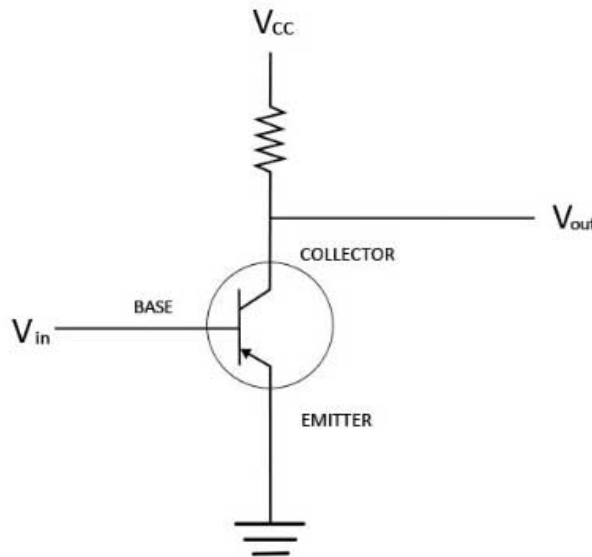


Figure 4 — Transistor Schematic [Ref. 5].

Since the 1970s, the complexity of integrated circuits has doubled every two years, while the size decreases exponentially. Today, we are able to build circuits on the nanoscale (nanotechnology) that

- President's Message
- From the Executive Vice President
- From the Editor's Desk
- Outside the Lines
- In the Spotlight:
 - The Use of Safety Cases in Certification and Regulation
 - Safety Implications of Software in Safety-Critical Devices
- System Safety in Healthcare
 - Swiss Cheese Model for Investigating the Causes of Adverse Events
- Announcements
- Gains from Losses:
 - Facts, Fiction and Public Perception
- Book Review:
 - Murder by Electrocutation, by David MacCollum
- Unintended Consequences:
 - TWA Flight 800 Accident
- Opinion (MacCollum)
- Upcoming Conferences/Calls for Papers
- Chapter News
- Mark Your Calendar
- About this Journal
- Advertising in eJSS
- Contact Us
- Puzzle

isograph

The Professionals' Choice

Integrated Software for
Safety, Reliability
Availability, Maintainability



- Fault/Event Tree Analysis
- Prediction
- FMECA/FMEA
- Reliability Block Diagrams
- Markov Analysis
- FRACAS
- Hazop
- Availability Simulation
- RCM
- Life Cycle Costing
- Network Availability
- Weibull
- Attack Tree/Threat Analysis

Contact us today for a free trial CD and discover how Isograph can help you.

Call
949 798 6114
or e-mail
sales@isograph.com

are being used for various purposes — some examples are electronic products, cancer treatments and accelerometers. However, with all of these electrical advances we are still in need of a command system that tells the brain of the circuit what to do — i.e., software.

Software and Programming Development

The concept of programming a device can be traced to Greek Mythology; the Antikythera Mechanism is an ancient mechanical computer that was built around 150-100 BCE and was used to calculate astronomical positions. Programs have gone from analog/mechanical machinery to the lines of digital code used in today's products. Software programs are a necessary part of integrated circuits; without a set of explicit instructions, the circuit is unable to function as designed. While there are several tools designed to facilitate staying on task in developing a program, one of the most widely used methods is the waterfall method.

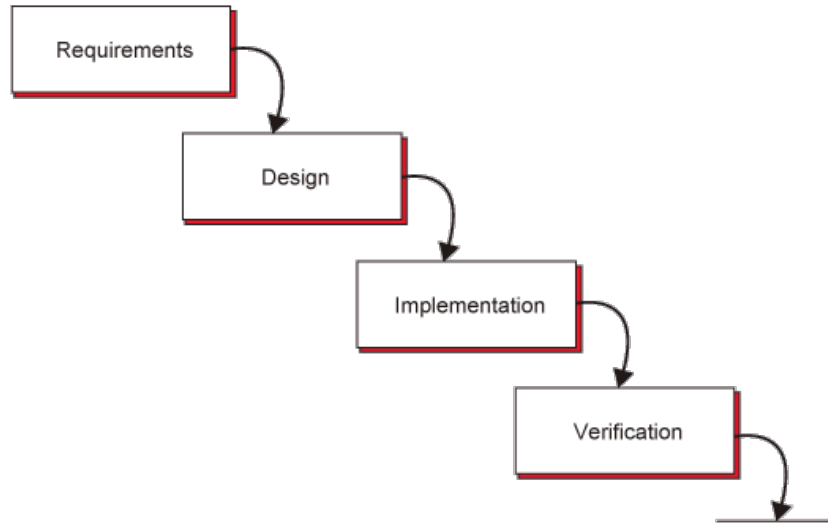


Figure 5 — Waterfall Model for Software Development [Ref. 6].

The waterfall model details the order that a software developer should follow to be able to complete a successful program:

- **Requirements:** Verify what the customer is asking for and what the expected end product is supposed to do.
- **Design:** Create a program around the general requirements and estimate the cost of the project based on the work effort needed to complete it. This is usually referred to as a scope document and can function as a legal contract between parties.
- **Implementation:** The code for the program is written. It is then tested for defects in the code line and documented for future usage.
- **Verification:** The software engineer double checks the scope document to verify that the program has been designed and developed to the required specifications.
- **Maintenance:** After the program has been tested and released to the client, it is important to supply software training and support to the client to make certain that the software is functioning properly. If any enhancements need to be made, they are done in this time period.

Safety Engineering and Software

Safety engineering must be an integral part of software design. Many of the safety-critical devices on the current market are dependent on reliable software. For example, almost all commercial airlines "fly by wire." Traditionally, aircraft control surfaces, throttle controls, landing gear, etc. were operated by a series of mechanical devices and cables but, due to the increase in size and weight of aircraft, it became necessary to replace these systems with more sophisticated electromechanical devices driven by software logic. When the controls are activated by the pilot, an electrical pulse runs through the wire and is translated by the software as a command to move the control surfaces, or increase or decrease acceleration. There are also several systems within the aircraft that run built-in tests to ensure that the onboard computer systems are functioning properly prior to take off.

Other systems that rely heavily on software are autonomous (and semi-autonomous) weapon



systems. They are a new and rapidly developing area of the warfare industry. The use of these systems requires a significant level of analysis and test to verify and validate that the system safety requirements are achieved to an acceptable level. Once operational, the systems typically run a "Built-In Test" (BIT) to ensure that the system startup has been successfully completed and that safety features are in place. After successful BIT and in an operational state, the weapon will autonomously follow whatever program has been placed within the system. One of the functions of the weapon is to use sensors to gather data from a target zone and an image processor, then to take that information and process it to identify potential targets. It then passes that information to a fire control system that calculates the data necessary to fire its payload at the potential target.

The healthcare industry also relies on safety-critical devices to monitor, medicate and save patients in the healthcare system. These devices range from IV infusion pumps and vital monitors to machines responsible for delivering accurate doses of medication for cancer treatment. As these devices directly interact with patients, it is paramount that the software that regulates the machinery be free from defect to avoid catastrophic results.

[« previous page](#) | [next page »](#)

Copyright © 2011 by the System Safety Society. All rights reserved. The double-sigma logo is a trademark of the System Safety Society. Other corporate or trade names may be trademarks or registered trademarks of their respective holders.





Vol. 47, No. 6 • November-December 2011

In the Spotlight

Safety Implications of Software in Safety-Critical Devices

by Amber D.Schauf

Pages 1 | 2 | 3 | 4 | 5

Hazard Assessment

There are two basic design considerations when developing a software system: performance and safety. Meeting performance requirements is normally measured during the test and validation verification phase and does not consider many safety aspects of the design. Consider an automobile with "accelerate by wire" capability. The simple Boolean matrix in Table 1 reflects the possible combinations of message traffic across the software interface for pressing the accelerator, and the importance of including safety considerations in the design.

Table 1 — Boolean Matrix.

1	1	Right Message, Right Time	Performance
1	0	Right Message, Wrong Time	Safety
0	1	Wrong Message, Right Time	Safety
0	0	Wrong Message, Wrong Time	Safety

The matrix indicates that other testing and analysis needs to be considered to fully characterize the implications of receiving erroneous messages that could potentially cause fatal accidents due to software errors within the "accelerate by wire" functionality. These errors could manifest as a result of either coding errors and/or timing and sequence problems associated with the hardware design.

There are several methods used to identify hazards and risks in a program. The two most commonly used forms, both private and government sector, are the Fault Tree Analysis (FTA) and the Failure Modes and Effects Analysis (FMEA).

- **Fault Tree Analysis:** This is an analytical method that starts from the complete system and funnels down through all possible errors the system could suffer. FTA starts with one well-defined, undesirable event and then models the combinations of events that will produce it. It consists of Boolean logic gates (AND, OR, NOT) that trace primary events that cover human error, system failures and outside influences on a system. A fault tree can be evaluated as a qualitative (cut sets and likelihood judgments) or a quantitative (probability of it happening) approach to failure assessment. FTA's are sometimes used in conjunction with an event tree.

**30th Annual
International
System Safety
Conference**

**Aug. 6-10, 2012
Loews Atlanta Hotel
Atlanta, Georgia**

**Check
[http://issc2012.
system-safety.org](http://issc2012.system-safety.org)
for the latest
information!**

- President's Message
- From the Executive Vice President
- From the Editor's Desk
- Outside the Lines
- In the Spotlight:
 - The Use of Safety Cases in Certification and Regulation
 - Safety Implications of Software in Safety-Critical Devices
- System Safety in Healthcare
 - Swiss Cheese Model for Investigating the Causes of Adverse Events
- Announcements
- Gains from Losses:
 - Facts, Fiction and Public Perception
- Book Review:
 - Murder by Electrocutation, by David MacCollum
- Unintended Consequences:
 - TWA Flight 800 Accident
- Opinion (MacCollum)
- Upcoming Conferences/Calls for Papers
- Chapter News
- Mark Your Calendar
- About this Journal
- Advertising in eJSS
- Contact Us
- Puzzle



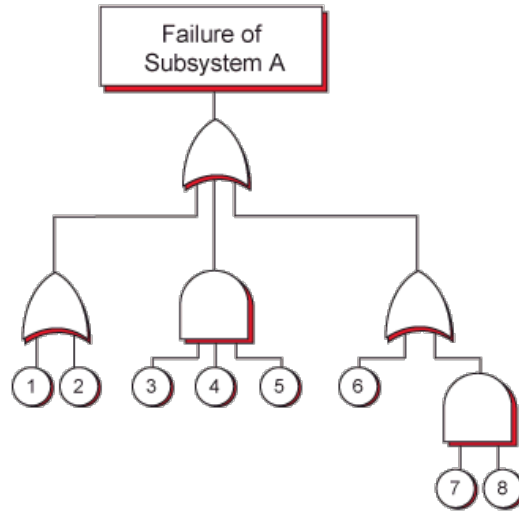


Figure 6 — Fault Tree Analysis Diagram [Ref. 7].

- Failure Modes and Effects Analysis: Initially developed by the military in the 1940s, FMEA is extensively used by a broad range of industry as a tool to study and reduce failure rates of products. There are three steps in assessing a product using FMEA. The first is severity, to determine all failure modes depending on the functional requirements of the system and their effects as a whole. The second step is occurrence; what is the cause of a failure mode and how many times is it reached. The third is detection. After examining the entire system and the self-detection measures built into the program, what is the level of risk that the failure will escape detection?

Each step has a value from 1 (no danger) to 10 (critical danger) assigned to it. The numbers are then tallied and the system is assigned a Risk Priority Number (RPN). Based on the value of the RPN, corrective measures are then taken to make the safety-critical device safer.

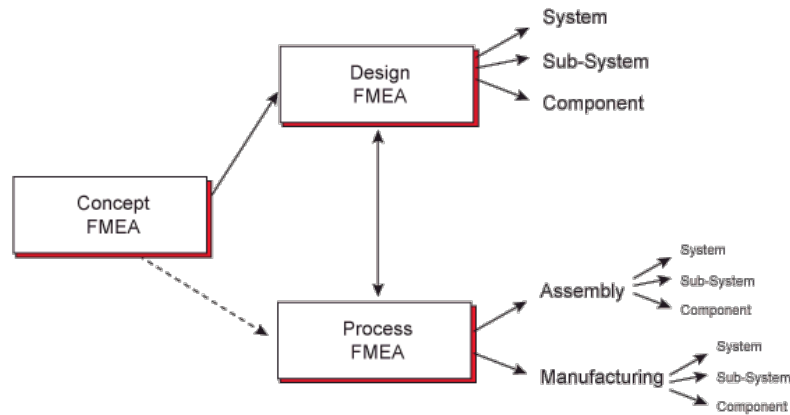


Figure 7 — FMEA Diagram [Ref. 8].
[Click to enlarge](#)

« previous page | next page »



Journal of
System
Safety

eEdition

A publication of the System Safety Society

Home
Subscriptions & Memberships
Contact
About eJSS
System Safety Society

Vol. 47, No. 6 • November-December 2011

In the Spotlight

Safety Implications of Software in Safety-Critical Devices

by Amber D.Schauf

Pages 1 | 2 | 3 | 4 | 5

Case Study 1: Patriot Missile Defense Weapon Software Failure

On February 25, 1991, during Operation Desert Storm, a Patriot missile defense system operating in Dhahran, Saudi Arabia failed to track and destroy an incoming enemy Scud missile. Due to the failure of the equipment, the missile hit an Army barracks and killed 28 Americans. After a review of the system and the facts of the case, it was determined that the software in the system-critical device was faulty. Reviewers found that the longer the system was left on (operating), the capability of the system to accurately track and identify incoming missiles became unreliable. At the time of this incident, they found the system had been operating for more than 100 hours and was looking in the wrong place for enemy fire. It originally located the incoming missile; however, when it recalculated for the position, the clock timing in the sensor was off, and when it could not locate the missile (again, it was looking in the wrong location), it took it off the radar [\[Ref. 9\]](#).

It was also noted in the official report that two weeks prior to the incident, there had been complaints from the Israeli Army (using the same technology) of inaccuracies with the Patriot system after only eight hours of operation. The Army had created a software modification to improve accuracy (not correct the actual problem) and the modification did not reach Dhahran until February 26, 1991 — one day after the catastrophic loss. After repairs to the software, the Patriot system was again deployed in Operation Iraqi Freedom in 2003. The system did not have the same failure.

Case Study 2: Therac-25

The Therac-25 was a radiation therapy machine produced by Atomic Energy of Canada Limited. Its purpose was to deliver predetermined doses of radiation therapy to cancer patients. The error was a software glitch that did not notify the radiation technician that a dose had been delivered to the patient. Instead, it gave an error message that the dose was not delivered when, in fact, it had been dispensed. The technician would then clear the error and hit the button again to deliver the dose. This caused the patient to receive up to 100 times the intended dose of radiation, having lethal results.

During a review of the Therac-25, it was found that AECL did not have an independent review of the software code used by the machine. The user manual did not address the error code displayed to the radiation technician and allowed him or her to override the system without clearing the code. There were also several engineering issues with the device. Used between 1985 and 1987, it was directly responsible for the deaths of six patients and the radiation injuries of several more. As a result of the procedural mitigations, AECL claimed to have achieved a solution but admitted they still didn't have a known cause. All of the machines were recalled in 1987 [\[Ref. 10\]](#).

Case Study 3: Royal Air Force Chinook

The CH-47 Chinook is a twin-engine, tandem rotor helicopter used for troop movement, the placement of artillery and battlefield re-supply efforts. It is now produced by Boeing Integrated

In the Spotlight

Safety Implications of Software in Safety-Critical Devices

by Amber D.Schauf

Pages 1 | 2 | 3 | 4 | 5

Case Study 1: Patriot Missile Defense Weapon Software Failure

On February 25, 1991, during Operation Desert Storm, a Patriot missile defense system operating in Dhahran, Saudi Arabia failed to track and destroy an incoming enemy Scud missile. Due to the failure of the equipment, the missile hit an Army barracks and killed 28 Americans. After a review of the system and the facts of the case, it was determined that the software in the system-critical device was faulty. Reviewers found that the longer the system was left on (operating), the capability of the system to accurately track and identify incoming missiles became unreliable. At the time of this incident, they found the system had been operating for more than 100 hours and was looking in the wrong place for enemy fire. It originally located the incoming missile; however, when it recalculated for the position, the clock timing in the sensor was off, and when it could not locate the missile (again, it was looking in the wrong location), it took it off the radar [\[Ref. 9\]](#).

It was also noted in the official report that two weeks prior to the incident, there had been complaints from the Israeli Army (using the same technology) of inaccuracies with the Patriot system after only eight hours of operation. The Army had created a software modification to improve accuracy (not correct the actual problem) and the modification did not reach Dhahran until February 26, 1991 — one day after the catastrophic loss. After repairs to the software, the Patriot system was again deployed in Operation Iraqi Freedom in 2003. The system did not have the same failure.

Case Study 2: Therac-25

The Therac-25 was a radiation therapy machine produced by Atomic Energy of Canada Limited. Its purpose was to deliver predetermined doses of radiation therapy to cancer patients. The error was a software glitch that did not notify the radiation technician that a dose had been delivered to the patient. Instead, it gave an error message that the dose was not delivered when, in fact, it had been dispensed. The technician would then clear the error and hit the button again to deliver the dose. This caused the patient to receive up to 100 times the intended dose of radiation, having lethal results.

During a review of the Therac-25, it was found that AECL did not have an independent review of the software code used by the machine. The user manual did not address the error code displayed to the radiation technician and allowed him or her to override the system without clearing the code. There were also several engineering issues with the device. Used between 1985 and 1987, it was directly responsible for the deaths of six patients and the radiation injuries of several more. As a result of the procedural mitigations, AECL claimed to have achieved a solution but admitted they still didn't have a known cause. All of the machines were recalled in 1987 [\[Ref. 10\]](#).

Case Study 3: Royal Air Force Chinook

The CH-47 Chinook is a twin-engine, tandem rotor helicopter used for troop movement, the placement of artillery and battlefield re-supply efforts. It is now produced by Boeing Integrated

eJSS
Archive is

ALWAYS
OPEN

for
business!

Each issue
of the eJSS
is available at
[www.system-safety.org/
jss/ejss.php](http://www.system-safety.org/jss/ejss.php)

Defense Systems and is mainly used by the U.S. Army and the Royal Air Force. On June 2, 1994, a Royal Air Force Chinook helicopter crashed into the Mull of Kintyre, Scotland, killing all 29 people onboard. The first review that came out cited gross negligence on the part of the flight crew for flying in dense fog. However, after a second review in 1996 (in an attempt to clear the flight crew of wrongdoing), the software code responsible for controlling the fly-by-wire system was examined. After going through "only 18 percent of the code, they found 486 anomalies and stopped the review" [Ref. 11]. They also found intermittent engine failures of the Chinook helicopter in question.

Similarity in Cases

When reviewing the three case studies, it was found that regardless of whether it was in the government or private sector, software was a key problem in each accident. These problems point to the following possible pitfalls associated with the design and development of safety-critical software:

Inadequate engineering rigor within the design:

- Lack of understanding of the intended use of the system
- Inexperienced or no safety professionals involved
- Inappropriate system safety program definition

Design and development issues:

- Inadequate requirement definition
- Lack of coding standards or guidelines

Test and analyses deficiencies:

- Lack of necessary hazard analyses (PHA, SHA, SSHA, etc.)
- No peer-review code analyses process
- No fault insertion, boundary value, static or dynamic tests

Configuration management issues:

- Multiple versions of unregulated software released
- Inability to trace software changes between versions

Conclusion

Software intensive systems are becoming more pervasive in all segments of our society, and we rely on these products to provide for our needs in many areas of our lives. It is becoming increasingly necessary for us to consider the negative aspects of operating these systems in a safe manner. It is incumbent upon those producing the products to ensure that safety has been considered during the design and development of potentially dangerous systems.

References

1. "Electricity." Def. 1, <http://encarta.msn.com/encnet/features/dictionary/dictionaryhome.aspx>, 2009, April 22, 2010.
2. Janku, Lumir. *Batteries of Babylon*, Photograph, *Ancient Batteries and Electric Devices*, 1996, http://www.bibliotecapleyades.net/ciencia/ciencia_hitech02a.htm, April 22, 2010.
3. "Otto Von Guericke," *Wikipedia, the Free Encyclopedia*, http://en.wikipedia.org/wiki/Otto_von_Guericke, accessed April 21, 2010.
4. "Leyden Jar." *Wikipedia, the Free Encyclopedia*, http://en.wikipedia.org/wiki/Leyden_jar, accessed April 22, 2010.
5. "Transistor." *Wikipedia, the Free Encyclopedia*, <http://en.wikipedia.org/wiki/Transistor>, accessed April 22, 2010.
6. "Waterfall Model." *Wikipedia, the Free Encyclopedia*, http://en.wikipedia.org/wiki/Waterfall_model, accessed April 22, 2010.
7. "File: FaultTree.png." *Wikipedia, the Free Encyclopedia*, http://en.wikipedia.org/wiki/File:Fault_tree.png, accessed April 21, 2010.
8. "Design FMEA, Process FMEA, Concept FMEA - FMEA / FMECA Types," *FMEA - Failure Mode and Effects Analysis, FMECA*. 2009, <http://www.fmea-fmea.com/types-of-fmea.html>, accessed April 22, 2010.
9. "PATRIOT MISSILE DEFENSE: Software Problem Led to System Failure at Dhahran, Saudi Arabia - Storming Media," *Storming Media: Pentagon Reports and Documents*, February 1, 1992, <http://www.stormingmedia.us/56/5684/A568443.html>, accessed April 22, 2010.
10. "Therac-25 Case Materials," *Welcome to ComputingCases.org*, http://computingcases.org/case_materials/therac/therac_case_intro.html, accessed April 22, 2010.
11. "1994 Scotland RAF Chinook Crash," *Wikipedia, the Free Encyclopedia*, http://en.wikipedia.org/wiki/1994_Scotland_RAF_Chinook_crash, accessed April 21, 2010.

