

SE 2890 Software Engineering Practices

Dr. Walter W. Schilling, Jr.
Instructor



Who are you and what do you want to learn in this course?

Practice SW 3 kills

New
Sweet learning

How to write SW.



Understand how to use non Java Libraries

Knowledge for future classes

Learn how to write more efficient code



About the instructor

- **Instructor:** Dr. Walter W. Schilling, Jr.
- **Office:** Walter Schroeder Library 335
- **Office Hours:**
 - Posted on course website
 - While I post office hours, I keep an open door policy. If I am in my office and the door is open, please feel free to stop in.
- **Telephone:** 414 277 7370
- **E-mail:** schilling@msoe.edu 
 - Best method to contact me during non-class days
 - Please prefix subject with SE2890
- **Course Web Page:** 
 - <https://myweb.msoe.edu/~schilling/msoe/Spring2012/se2890/se2890.shtml>



About the Instructor (Continued)

- Ohio Northern University graduate in Electrical Engineering
 - Computer Science Minor
- Masters and PhD. from University of Toledo
 - Specialized in Computer Systems Design and Software Reliability
- Worked in Automotive Industry for approximately 6 years
 - Audio Software Engineer – Embedded Systems Design
 - US Patent 6,707,768
 - “Randomized Playback of Tracks in a Multimedia Player”
- Personal Website: <http://www.walterschilling.org>



vs
Multimedia



Introduction to Software Failure

Outcomes:

- Recognize the economic impacts of software failure
- Discuss the ramifications of software failure
- Understand that to improve software quality, we must learn from our past failures
- Explain the concept of the CMM and CMMI process initiatives.

Dr. Walter W. Schilling, Jr.

Instructor



Introduction

"The most significant problem facing the data processing business today is the software problem that is manifested in two major complaints: software is too expensive and software is unreliable."

Still true

-Glenford J. Myers: **Software Reliability: Principles and Practices** 1976. [Mye76, page 3]

old article



Introduction – Impact of Failure

- Large economic impact to software failure
 - \$59.5 billion annual cost to economy.
- Fiscal year 2003 DOD spent \$21 billion on software development
 - \$8 billion (40%) spent to fix reliability problems in software



Software Failure is
expensive!

Introduction – Impact of Failure

Recall

- Avionics
 - 39% of FAA Air-worthiness directives related to software
- Medical Industry
 - 79% of medical device recalls can be attributed to software
 - 41% of pacemaker recalls 1990-2000 related to software
- Automotive Industry
 - Single software failure resulted in recall of 2.2 million cars and \$20 million cost.
- Consumer electronics
 - Software has become the principle source for reliability problems
- Overall
 - Software driven outages exceed hardware by a factor 10.



The Crisis in Software

- Software is done when its done
 - Only 29% of **ALL** projects succeed
 - 18% fail outright
 - 53% were challenged
 - Cost overruns
 - Late
 - Fewer than desired features
- And this is considered a huge improvement since 1994
 - 16% succeed, 31% fail, and 53% were challenged
- Management is at best an “Art Form”
 - Skills and Techniques are relatively new
 - It just isn’t done

Double

Half

Failures you have been exposed to?

Laptop graphics drivers (crash machine)

of days
since
quantum.

June 30 Software Failure

December 31, 2008

```
while (days > 365)  
{  
    if (IsLeapYear(year))
```

Days == 366?

```
{
```

```
    if (days > 366)
```

```
    {
```

```
        days -= 366;
```

```
        year += 1;
```

```
    }
```

```
    else
```

```
    {
```

```
        days -= 365;
```

```
        year += 1;
```

```
    }
```

```
}
```

End of
a leap year
Day 366

Non Leap year



battery dies. In finite loop until



ATT Long Distance Failure January 15, 1990



```
1: ...
2: switch (message)
3: {
4:   case INCOMING_MESSAGE:
5:     if (sending_switch ==
        OUT_OF_SERVICE)
6:       {
7:         if (ring_write_buffer == EMPTY)
8:           send_in_service_to_smm(3B);
9:         else
10:            break; /* Whoops */
11:       }
12:     process_incoming_message();
13:     break;
14: ...
15: }
16: do_optional_database_work();
17: ...
```

Debug
code

Skipped,
causing queue overflow.

Southern California Air Traffic Control Communications Failure

September 14, 2004

- Communications lost with 400 airplanes in southwest US.
 - Planes visible on radar
 - no voice communications existed.
- Cause(s)
 - Design flaw in application and Windows Server 2000 API caused lockup after 49.7 days continuous operation
 - Failure of technician to reboot system on monthly basis.
 - Backup system incapable of handling demand

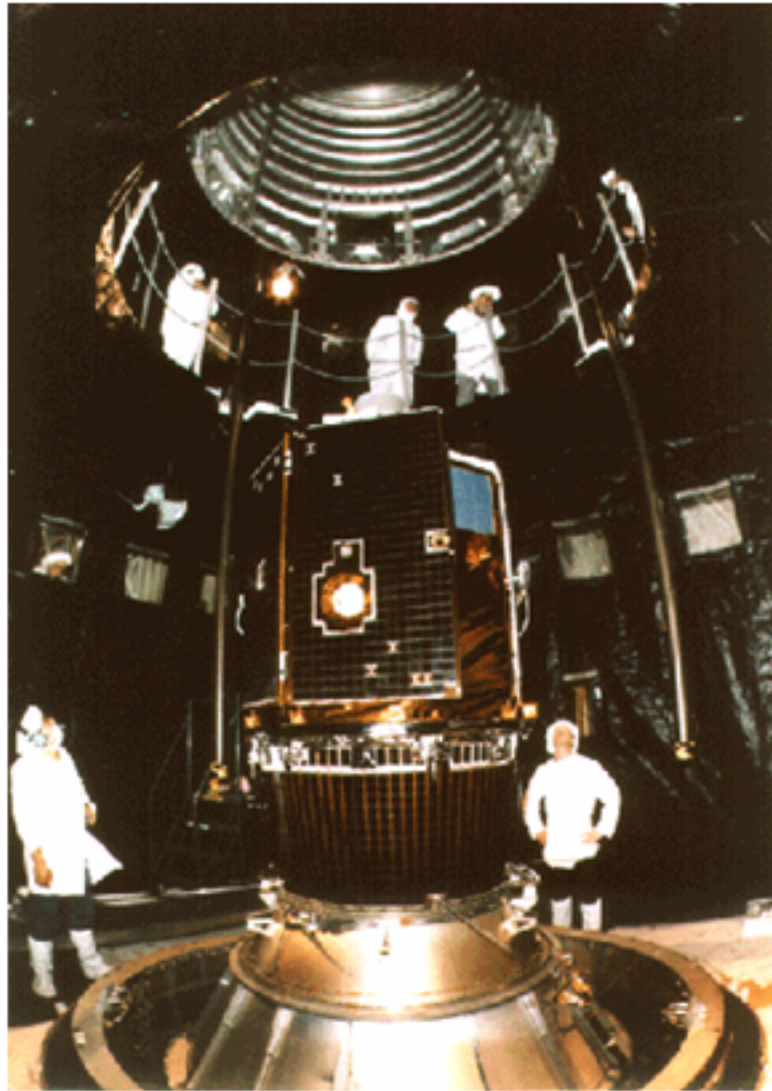


2³² ms

← only

tested @ night

Clementine Mission Failure



- Prototype for “Faster, Better, Cheaper” operation.
 - Scientific objectives were secondary
- Many problems occurred during mission.
 - Over 3000 floating point exceptions during mission.
 - Hardware reset required at least 16 times.
- Mission failed May 7, 1994
 - exception occurred. with a thruster stuck on
 - Burned up all fuel and imparted 80 RPM rotation
- Watchdog safety feature of microcontroller not used.

Bad idea.
Considered too high of risk by team.



Ariane 5

June 4th, 1996

- Total failure of the Ariane 5 launcher maiden flight
- Caused by a typecast from a 64 bit floating point number to a 16 bit int
 - No exception handler associated with the conversion
- Backup system was identical in all regards
 - 37ms later, backup system failed.
- Software Developed in Ada.
 - Had code been developed in C, problem most likely would not have occurred.



Mars PathFinder Priority Inversion Defect

September 27, 1997

- September 27, 1997, communications with Sojourner was suddenly lost.
 - The system encountering periodic total systems resets
 - Data lost and ground commands ignored.
 - Problem was quickly traced to a watchdog reset
 - Reset caused by a priority inversion within the system.



Milstar Launch Failure

April 30, 1999



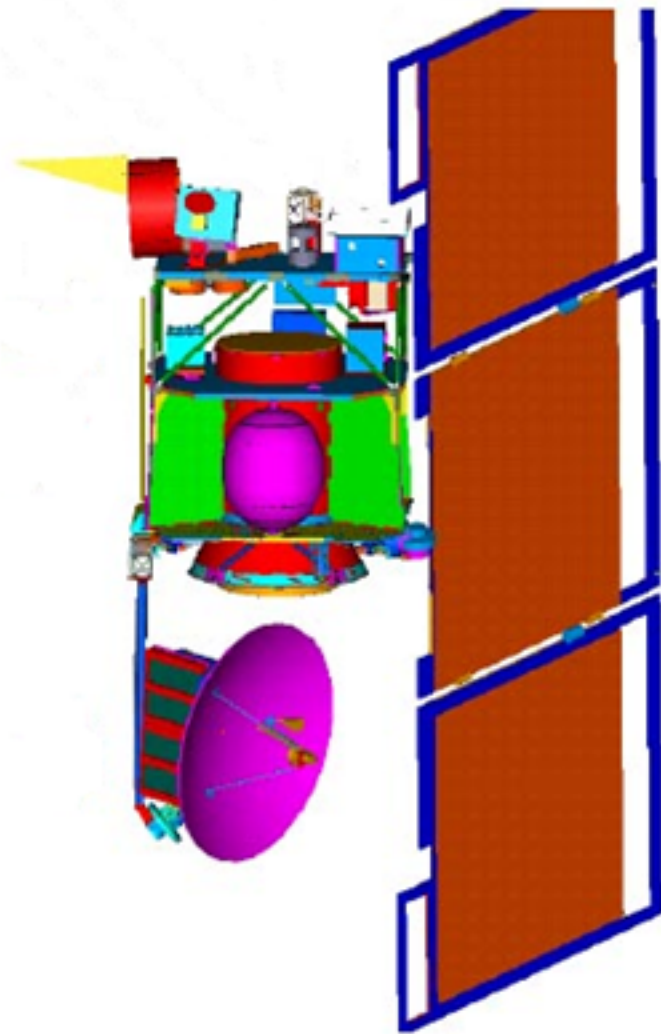
- *\$433.1 million Titan IV rocket launched from Cape Canaveral with third Milstar Satellite.*
- *Launch failed*
 - Satellite did not reach necessary geostationary orbit
- *Why?*
 - *Table entry of -0.1992476 instead of -1.992476*

*Control
Coefficient*



Mars Climate Orbiter Failure

September 23, 1999



- Mars Climate Orbiter Lost
- Causes [Slm+99]
 - Thruster used English units.
 - Model used metric units.

Units
matter



Sea Launch F-1 Satellite Launch Failure

- March 12, 2000
- Zenit-3SL lost with \$200 million ICO Global Communications satellite 8 minutes into mission
- Root cause of failure: logic incorrectly changed Resulted in Helium valve not being closed

Original Source Code	<pre>If ((state is Abort) or (countdown proceeds past timeA)) { close valve a }</pre>
Intended Source Code	<pre>If ((state is Abort) or (countdown proceeds past timeB)) { close valve a }</pre>
Implemented Source Code	<pre>If (state is Abort) { close valve a }</pre>

Minor Logic Miss



Midwestern Blackout

August 14, 2003



- 50 million customers lost power
 - Economic loss between \$4.5 and \$10 billion. [ELC04]
- Software contribution:
 - Energy Management System Failed

Introduction to Software Failure



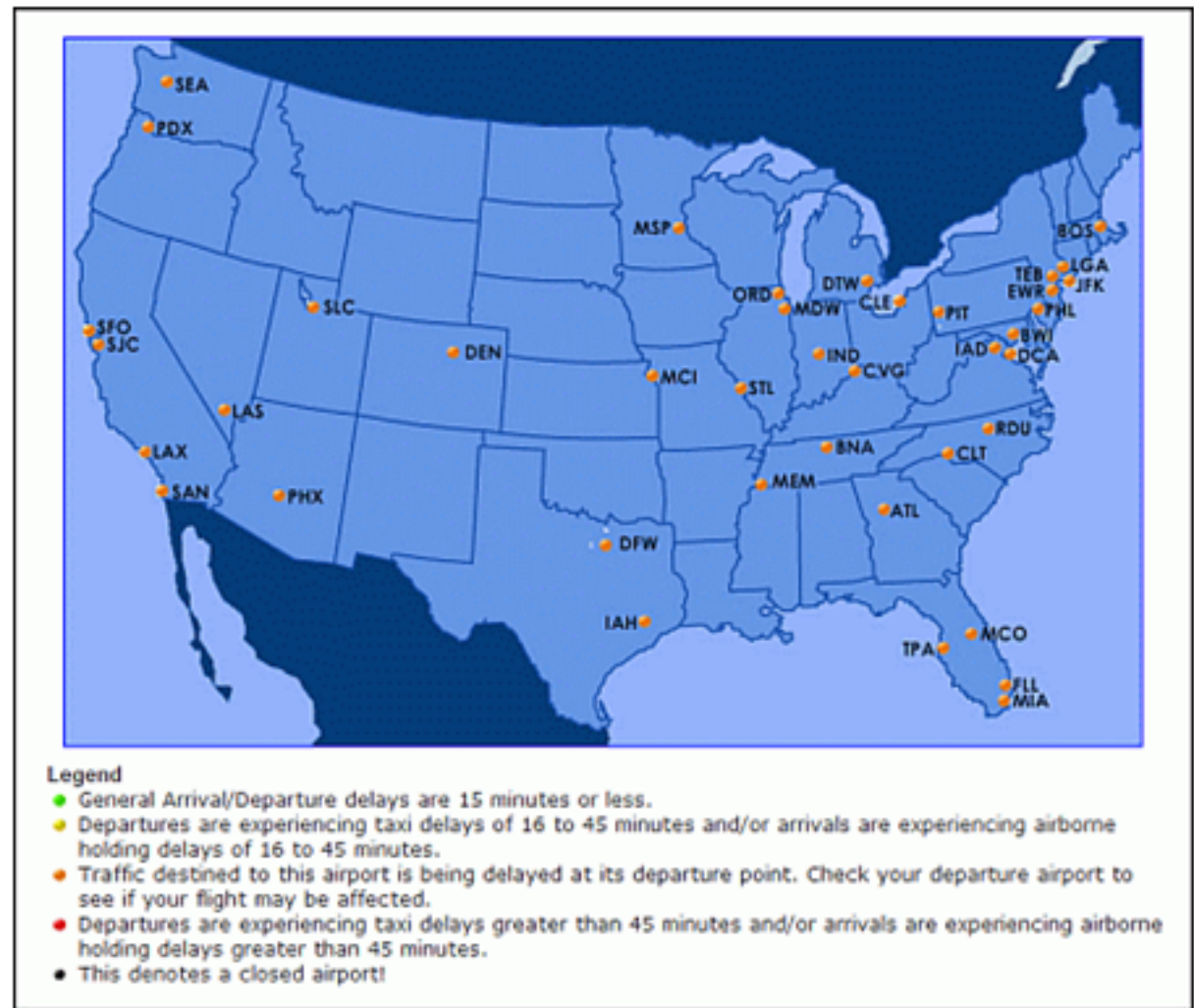
Blackberry Failure

- February, 2008
 - 3.5 hour complete system failure
 - No e-mails sent or received
- Root Cause
 - Unsuccessful infrastructure upgrade by BlackBerry vendor Research In Motion.



Air Traffic Control -> August 28, 2008

- A corrupt file wasn't caught by validation?
- 2 1/2 hours to restart after the failure?
- The "backup" computer couldn't handle the failover load?
- The restored-to-service computer couldn't clear the accumulated backlog until new transactions were suppressed?



Is This Engineering?

- Predictable? ✓
- Scientifically based? ✓
- Professional? - *would we allow*
- Repeatable? *bridges to be*
- Can we and should we do better? *built this way?*

What's the Solution?

- No easy answer
- Requires:
 - Knowledge
 - Buy in
 - Reform
 - Hard work

Understanding
what works
Doing what works

Changing what
we do and how
we do it.

Effort!



Your Part

- Computer Engineers
 - Design computer hardware ✓
 - Design Embedded Systems ✓
 - Implement Software ✓



SE2890 – We Can't Do It All

- Methods
 - Object-oriented process
 - Design patterns
 - CASE tool usage
- Project Experience -> Pseudo embedded
 - RC Robot control over Bluetooth
 - Teamwork



Legos!

