cs2220: Engineering Software

Class 25: Software Disasters

Fall 2010
UVa
David Evans

Why Study Software Disasters?

Exam 2 out today, due at beginning of class Thursday. It should not be a disaster!

Tech Gadgets Mobile Enterprise CrunchBase More

CrunchGear

About Archives Contact Advertise Jobs Reviews

Got a tip for us? Send it to our anonymous, top-secret tip line: tips@crunchgear.com

30GB Zunes all over the world fail en masse

by Matt Burns on December 31, 2008 119 Comments

It seems that a random bug is affecting a bunch, if not every, 30GB Zunes. Real early this morning, a bunch of Zune 30s just stopped working. No official word from Redmond on this one yet but we might have a gadget Y2K going on here. Fan boards and support forums all have the same mantra saying that at 2:00 AM this morning, the Zune 30s reset on their own and doesn't fully reboot. We're sure Microsoft will get flooded with angry Zune owners as soon as the phone lines open up for the last time in 2008. More as we get it.

Update 2: The solution is … kind of weak: let your Zune run out of battery and it'll be fixed when you wake up tomorrow and charge it.

Zune.net, ZuneBoards, ZuneScene, Gizmodo

Update: Reddit adds:

MosquitoWipes 314 points 7 hours ago [-]
I guess now they are frozune.

jones77 210 points 7 hours ago [-]
I wonder when normal service will rezune?

MosquitoWipes 205 points 6 hours ago [-]
Probably not anytime zune.

Macworld

MAC IPHONE & IPAD DIGITAL PHOTO IPOD & ENTERTAINMENT CREATE

News Reviews Help & Tips Shop & Compare

Beat the crowds and save on all of your holiday gifts!

MacUser — News, info, and opinion by Mac users, for Mac users.

62 Comments +32 Recommendations 0 Digg ShareThis

Microsoft Zunes spontaneously dying all over the place

Posted on Dec 31, 2008 11:13 am by Dan Moren, Macworld.com

Oh, Danny boy…the Zunes, the Zunes are calling. As a sign of the impending apocalypse—no doubt caused by the addition of the leap second—a plague has descended upon scads of Microsoft's Zune media players. As of 2AM this morning, Zunes around the world have begun to freeze with full loading bars, resulting in completely unresponsive Zunes. Oh, the horror!

According to Gizmodo:

Right, so this is a weird one: we're getting tons of reports—tons—about failing Zune 30s. Apparently, the players began freezing at about midnight last night, becoming totally unresponsive and practically useless.

SIMILAR ARTICLES:
- HTC Surround: Solid Windows Phone, Laughable "Boombox"
- Jobs speaks! The complete transcript
- Opinion: The iPad stands alone
- HTC HD7
- Live Update: Microsoft E3 press show
- Opinion: Why MS's Zune scares Apple to the core

```
//                                                    http://pastie.org/349916
// Copyright (c) Microsoft Corporation.  All rights reserved.
//
//
// Use of this source code is subject to the terms of the Microsoft end-user
// license agreement (EULA) under which you licensed this SOFTWARE PRODUCT.
// If you did not accept the terms of the EULA, you are not authorized to use
// this source code. For a copy of the EULA, please see the LICENSE.RTF on your
// install media.
//
//------------------------------------------------------------------------------
//
//  Copyright (C) 2004-2007, Freescale Semiconductor, Inc. All Rights Reserved.
//  THIS SOURCE CODE, AND ITS USE AND DISTRIBUTION, IS SUBJECT TO THE TERMS
//  AND CONDITIONS OF THE APPLICABLE LICENSE AGREEMENT
//
//------------------------------------------------------------------------------
//
//  Module: rtc.c
//
//  PQOAL Real-time clock (RTC) routines for the MC13783 PMIC RTC.
//
//------------------------------------------------------------------------------
```

```
//------------------------------------------------------------------------------
// Global Variables
//These macro define some default information of RTC
#define ORIGINYEAR       1980            // the begin year
#define MAXYEAR          (ORIGINYEAR + 100)   // the maxium year
#define JAN1WEEK         2               // Jan 1 1980 is a Tuesday

static const UINT8 monthtable[12] = {31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};
static const UINT8 monthtable_leap[12] = {31, 29, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};
...
```

```
//------------------------------------------------------------------------------
//
// Function: IsLeapYear
//
// Local helper function checks if the year is a leap year
//
// Parameters:
//
// Returns:
//
//
//------------------------------------------------------------------------------
static int IsLeapYear(int Year)
{
   int Leap;

   Leap = 0;
   if ((Year % 4) == 0) {
      Leap = 1;
      if ((Year % 100) == 0) {
         Leap = (Year%400) ? 0 : 1;
      }
   }

   return (Leap);
}
```

MODIFIES: nothing
EFFECTS: Returns 0 if Year is not a Leap year, 1 otherwise.

return

Leap = if x then y else z

if (Year % 400) {
   Leap = 0;
} else {
   Leap = 1;
}

```
#define ORIGINYEAR 1980
…
// Function: ConvertDays
//
// Local helper function that split total days since Jan 1, ORIGINYEAR into
// year, month and day
//
// Parameters:
//
// Returns:
//     Returns TRUE if successful, otherwise returns FALSE.

BOOL ConvertDays(UINT32 days, SYSTEMTIME* lpTime)
{
   int month, year;
   year = ORIGINYEAR;

   while (days > 365) {
      if (IsLeapYear(year)) {
         if (days ≥ 366) {
            days -= 366;
            year += 1;
         } else { return false; }
      } else {
         days -= 365;
         year += 1;
      }
   }
   …
```

+ … days before 2008
366 of 2008

http://pastie.org/349916

We contacted a Microsoft spokesperson, who confirmed the issue with this official statement: "Early this morning we were alerted by our customers that there was a widespread issue affecting our 2006 model Zune 30GB devices (a large number of which are still actively being used). The technical team jumped on the problem immediately and isolated the issue: a bug in the internal clock driver related to the way the device handles a leap year. That being the case, the issue should be resolved over the next 24 hours as the time change moves to January 1, 2009. We expect the internal clock on the Zune 30GB devices will automatically reset tomorrow (noon, GMT). By tomorrow you should allow the battery to fully run out of power before the unit can restart successfully then simply ensure that your device is recharged, then turn it back on."

http://www.pcworld.com/article/156240/microsoft_says_leap_year_bug_caused_zune_failures.html

Questions about Bugs

Immediate

What is going wrong?

What is the bug?

Systemic

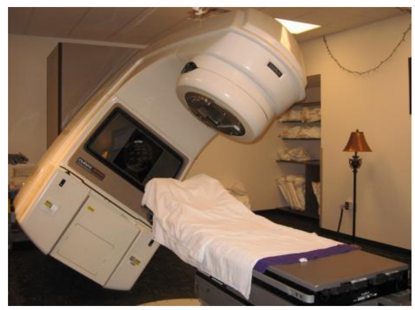
Is this bug a symptom of larger problems in the software design?

Why didn't testing catch this?

Is this bug a symptom of larger problems in the development process, team, etc.?

In this case, the code came from Freescale, integrated into Microsoft Project (without going through MS development process)

Therac-25



Radiation Therapy Machine
Atomic Energy of Canada

1985-1987: gave six patients massive overdoses of radiation (3 died)

Nancy Levenson, *Medical Devices: The Therac-25*
http://sunnyday.mit.edu/papers/therac.pdf

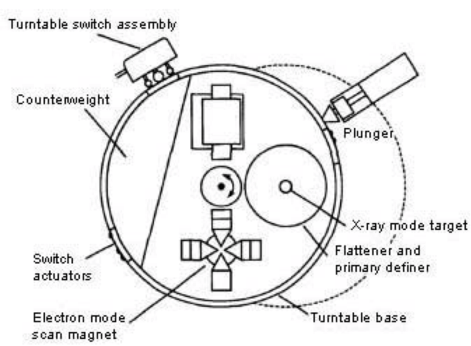


Figure B. Upper turntable assembly

Assumptions in AECL's safety analysis:

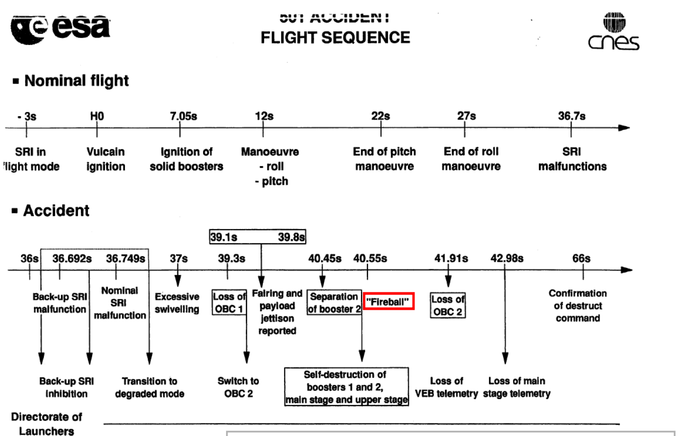
1. Programming errors have been reduced by extensive testing on a hardware simulator and under field conditions on teletherapy units. Any residual software errors are not included in the analysis.
2. Program software does not degrade due to wear, fatigue, or reproduction process.
3. Computer execution errors are caused by faulty hardware components and by "soft" (random) errors induced by alpha particles and electromagnetic noise.

Nancy Levenson, *Medical Devices: The Therac-25*
http://sunnyday.mit.edu/papers/therac.pdf

[Ariane 5 Movie]()

Ariane 5

- $500M rocket developed by European Space Agency
- June 4, 1996: first launch
  - 37s after ignition: lost guidance
  - 40s: exploded



Ariane 5 Inquiry Board Report (Jacques-Louis Lions): http://esamultimedia.esa.int/docs/esa-x-1819eng.pdf

Flight Control System

Inertial Reference System (SRI)

- Calculates angles and velocities from on-rocket sensors (gryos, accelerometers)
- Data sent to On-Board Computer that executes flight program (controls booster nozzles, valves)

Redundancy in design to improve reliability

- Two separate computers running SRIs in parallel (same hardware and software) – one is "hot" stand-by used if OBC detects failure in "active" SRI

Design based on Ariane 4

- **Software for SRI mostly reused from Ariane 4 implementation**

Number Overflow Problems

- 16-bit signed integer
  - 2^{16} = 65536 different values (-32768 – 32767)
- Alignment code converted the horizontal velocity (64-bit floating point value from sensors = up to ~10^{308}) to a 16-bit signed integer
- Overflow produces exception (Operand Error)

Defensive Programming

"The data conversion instructions were not protected from causing an Operand Error, although other conversions of comparable variables in the same place in the code were protected."

It has been stated to the Board that not all the conversions were protected because a maximum workload target of 80% had been set for the SRI computer. To determine the vulnerability of unprotected code, an analysis was performed on every operation which could give rise to an exception, including an Operand Error. **In particular, the conversion of floating point values to integers was analysed and operations involving seven variables were at risk of leading to an Operand Error.** This led to protection being added to four of the variables, evidence of which appears in the Ada code. **However, three of the variables were left unprotected.** No reference to justification of this decision was found directly in the source code. Given the large amount of documentation associated with any industrial application, the assumption, although agreed, was essentially obscured, though not deliberately, from any external review.

The reason for the three remaining variables, including the one denoting horizontal bias, being unprotected **was that further reasoning indicated that they were either physically limited or that there was a large margin of safety, a reasoning which in the case of the variable BH turned out to be faulty.** It is important to note that the decision to protect certain variables but not others was taken jointly by project partners at several contractual levels.

Although the source of the Operand Error has been identified, this in itself did not cause the mission to fail. The specification of the **exception-handling mechanism also contributed to the failure.** In the event of any kind of exception, the system specification stated that: the failure should be indicated on the databus, the failure context should be stored in an EEPROM memory (which was recovered and read out for Ariane 501), and finally, the SRI processor should be shut down.

It was the decision to cease the processor operation which finally proved fatal. Restart is not feasible since attitude is too difficult to re-calculate after a processor shutdown; therefore the Inertial Reference System becomes useless. The reason behind this drastic action lies in the culture within the Ariane programme of only addressing random hardware failures. From this point of view exception - or error - handling mechanisms are designed for a random hardware failure which can quite rationally be handled by a backup system.

Java Version

```
public class Overflow {
  public static void main (String args[]) {
    int x;
    double d = 5000000000.0;

    x = (int) d;
    System.out.println ("d = " + d + "  / " + "x = " + x);
  }
}
```

d = 5.0E9 / x = 2147483647

What is 2147483647 + 1?

-2147483648

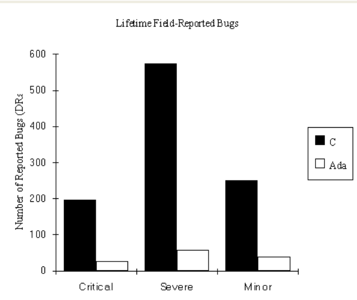
Ada Programming Language

- Developed by a 1970s US DoD effort to create a safe, high-level, modular programming language
- 1987-1997: All DoD software projects **required** to use Ada
- Still fairly widely used in safety-critical software
  - Boeing 777
  - SPARK/Ada (subset with verification)

Ada Package Declaration

```
package Rational_Numbers is
  type Rational is
    record
      Numerator   : Integer;
      Denominator : Positive;
    end record;
  function "="(X,Y : Rational) return Boolean;
  function "/" (X,Y : Integer)  return Rational;
  function "+" (X,Y : Rational) return Rational;
  function "-" (X,Y : Rational) return Rational;
  function "*" (X,Y : Rational) return Rational;
  function "/" (X,Y : Rational) return Rational;
end Rational_Numbers;
```

Zeigler, 1995 http://www.adaic.com/whyada/ada-vs-c/cada_art.html



Type safety and information hiding are valuable: Ada code has 1/10th as many bugs as C code, and cost ½ as much to develop

Ada Exception Handling

```
begin
  ... --- raises exception
end
exception
  when Exception: action
```

If exception raised in block B

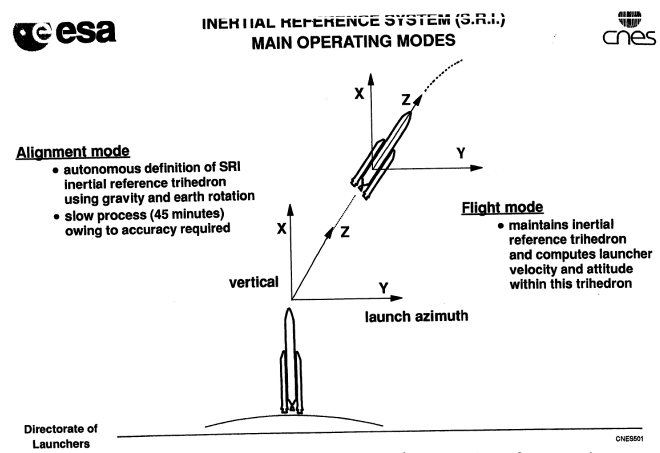
If there is a handler, jumps to its action; if not, exception propagates to call site (and up)

Inertial Reference System

- Exception in alignment code for number conversion
- No handler in procedure
- Propagated up to top level
- SRI response to exception is to shutdown and put error on databus

Why was the alignment code still running?

The error occurred in a part of the software that only performs alignment of the strap-down inertial platform. This software module computes meaningful results only before lift-off. As soon as the launcher lifts off, this function serves no purpose.



p. 36 (appendix of report)

The original requirement accounting for the continued operation of the alignment software after lift-off was brought forward more than 10 years ago for the earlier models of Ariane, in order to cope with the rather unlikely event of a hold in the count-down e.g. between - 9 seconds, when flight mode starts in the SRI of Ariane 4, and - 5 seconds when certain events are initiated in the launcher which take several hours to reset. The period selected for this continued alignment operation, 50 seconds after the start of flight mode, was based on the time needed for the ground equipment to resume full control of the launcher in the event of a hold. **This special feature made it possible with the earlier versions of Ariane, to restart the count-down without waiting for normal alignment**, which takes 45 minutes or more, so that a short launch window could still be used. In fact, this feature was used once, in 1989 on Flight 33.

The same requirement does not apply to Ariane 5, which has a different preparation sequence and it was maintained for commonality reasons, presumably based on the view that, **unless proven necessary, it was not wise to make changes in software which worked well on Ariane 4.**

Why didn't testing find this?

ORIGIN OF THE MALFUNCTION (3/3)

- The fault could not be detected on the ground by any of the static or environment tests performed on the SRIs
- The error could have been detected in testing:
  - → on the software alone. A test of this kind was performed but unfortunately with an unsuitable choice of parameter
  - → by simulating the Ariane 5 trajectories through electronic input to the SRI instead of the sensors. This type of simulation was performed at launcher level, but without actual SRI equipment

What was the real problem?

CONCLUSIONS

- Software design errors:
  - → maintenance after lift-off of pre-launch function incompatible with flight
  - → saturation of capacity to represent a variable
  - → shutdown of processor on detection of malfunction
- Not detected:
  - → BY the series of tests and reviews carried out under the programme, which otherwise demonstrated their effectiveness (making thousands of corrections)
  - → WHY - ground / flight functional interface (different reactions required)
    - tests at equipment and system levels not suffficiently representative
- The system architecture is not implicated

What are the lessons?

Recommendations

ONBOARD SOFTWARE (cont'd)

FPROG 2 ***Update A5-DF-1-X-04 (flight control algorithm definition file), identify any superfluous functions and then check for consistency with the flight programme technical specification***

The update concerns only the consistency of documents. The "algorithmic reference" is, for its part, fully in phase with the flight software. However, verifying consistency will make it possible to reassess whether functions are superfluous or require simplification

FPROG 3 ***Study dual-failure management, without thereby changing the original philosophy (no switchback to equipment found to have failed). The aim is to make functions after the second failure very "tolerant", without any definitive mission termination whether at equipment or flight software level***

CORRECTIVE MEASURES

<u>Specific measures</u>

- Correction of the problem in the SRI that led to the accident
- Reexamination of all software embedded in equipment
- Improvement of the representativeness (vis-à-vis the launcher) of the qualification testing environment
- Introduction of overlaps and deliberate redundancy between successive tests:
  - → at equipment level
  - → at stage level
  - → at system level
- Improvement and systematisation of the two-way flow of information:
  - → up from equipment to system: nominal and failure-mode behaviour
  - → down from system to equipment: use of equipment items in flight

501 ACCIDENT
CORRECTIVE MEASURES

esa — cnes

__SRI__

- Switch-off or inhibition of alignment function after lift-off
- Analysis / modification of processing, particularly on detection of a malfunction (no processor shutdown)
- Testing to check coverage of the SRI flight domain

__System qualification environment (ISF at Les Mureaux)__

- General improvement of representativeness through systematic use of real equipment and components wherever possible
- Simulation of real trajectories on SRI electronics

__General measures__

- Critical reappraisal of all software (flight program + embedded software)
- Review of mechanisms for managing double failures
- Improvement of facilities for acquisition and retrieval of telemetry data
- Improvement of overall coordination relating to software

Directorate of Launchers

CNES501

Bertrand Meyer’s Analysis

“Reuse without a contract is sheer folly! Without a precise specification attached to each reusable component -- precondition, postcondition, invariant -- no one can trust a supposedly reusable component.”

http://archive.eiffel.com/doc/manuals/technology/contract/ariane/page.html

Ken Garlington’s Critique

- Design contracts unlikely to solve this problem:
  - Specification would need to correctly identify precondition
  - Code review would need to correctly notice unsatisfied precondition
  - Or, run-time handler would need to recover correctly

http://home.flash.net/~kennieg/ariane.html

Charge

- Avoid a software disaster for your projects
  - Coordinate with your team closely: all your code should be working together now
  - Make sure simple things work before implementing “fancy features”
- Subscribe to RISKS to get a regular reminder of software disasters: http://catless.ncl.ac.uk/Risks

Exam 2 is out now, due at beginning of class Tuesday
(it should not be a software disaster either!)