Lessons from the real-world

Crafting Quality Software

Ganesh Samarthyam
sgganesh@gmail.com
www.designsmells.com

Why care about software quality?

Reason #1. Software defects cost huge money!

Cambridge University Study (2012)

http://www.cambridge-news.co.uk/Education/Universities/Experts-battle-192bn-loss-to-computer-bugs-18122012.htm

computer defects cost the global economy more than £192 billion a year (or US $307 bn)!

Global 'IT Debt' - $500 billion for the year 2010; potential to grow to $1 trillion by 2015

Gartner Study (2010)

http://www.gartner.com/newsroom/id/1439513

Reason #2. Software defects cause project failures

”The most common reason for schedule slippages, cost overruns, and outright cancellation of major systems is that they contain too many bugs or defects to operate successfully”

Capers Jones in “Why Projects Fail?”, CrossTalk, The Journal of Defense Software Engineering, 2006.

http://en.wikipedia.org/wiki/Capers_Jones

Reason #3. Software defects result in losses

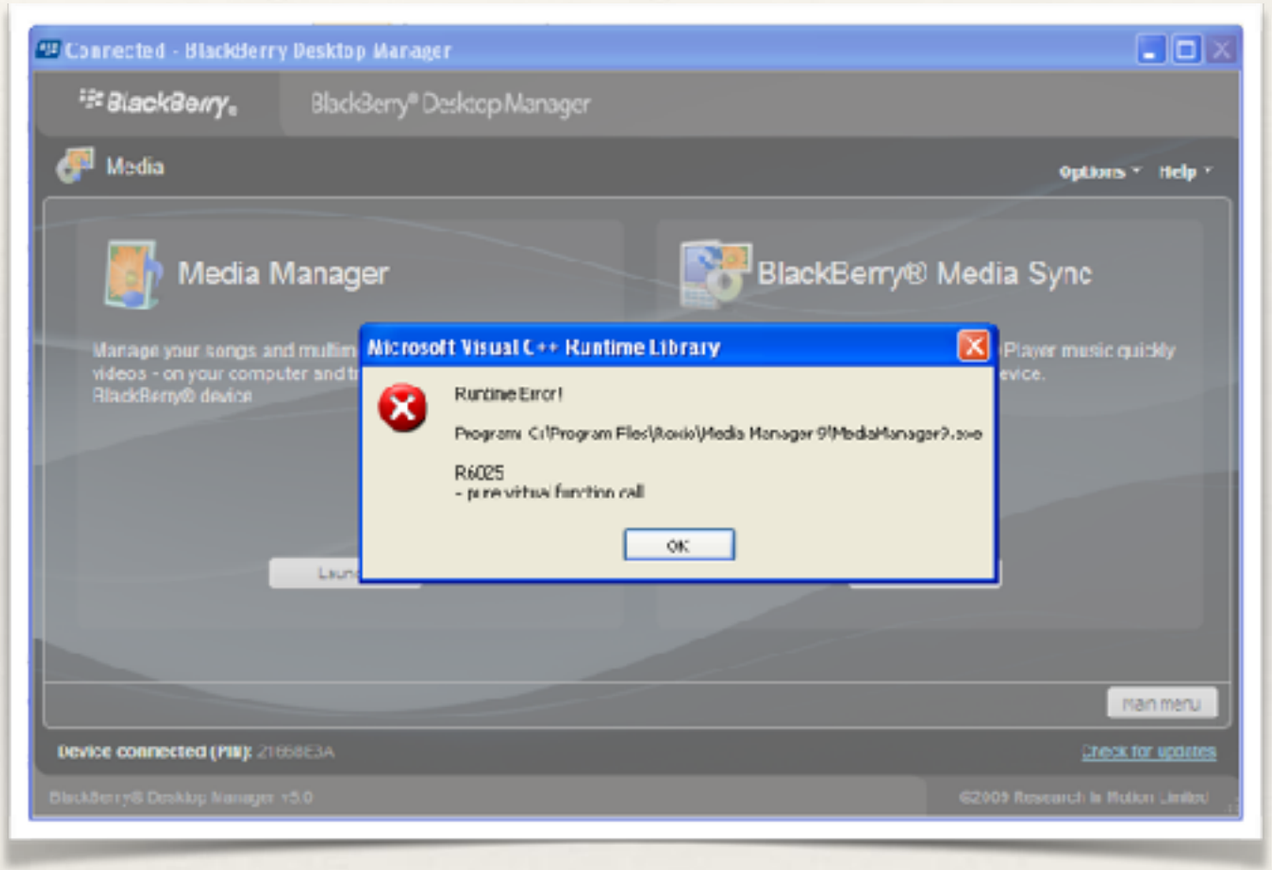
- Depending on the criticality of the software, losses could be in the form of
  - lower-sales (in case of software products)
  - financial losses (in business critical systems)
  - even human lives (in case of safety critical systems such as medical devices, or mission-critical systems such as rockets)

Some real-world software failures

Crash in Blackberry phone

ignorance on design practices

- Supporting multimedia software for my Blackberry crashed with this design error!
- Classic OO design problem of "polymorphic call in constructors"
  - Constructors do not support fully as the derived objects are not constructed yet when base class constructor executes



Zune media player freeze

- On Dec 31 2008, all Zune media players froze!

- Cause traced back to a non-terminating loop error in code

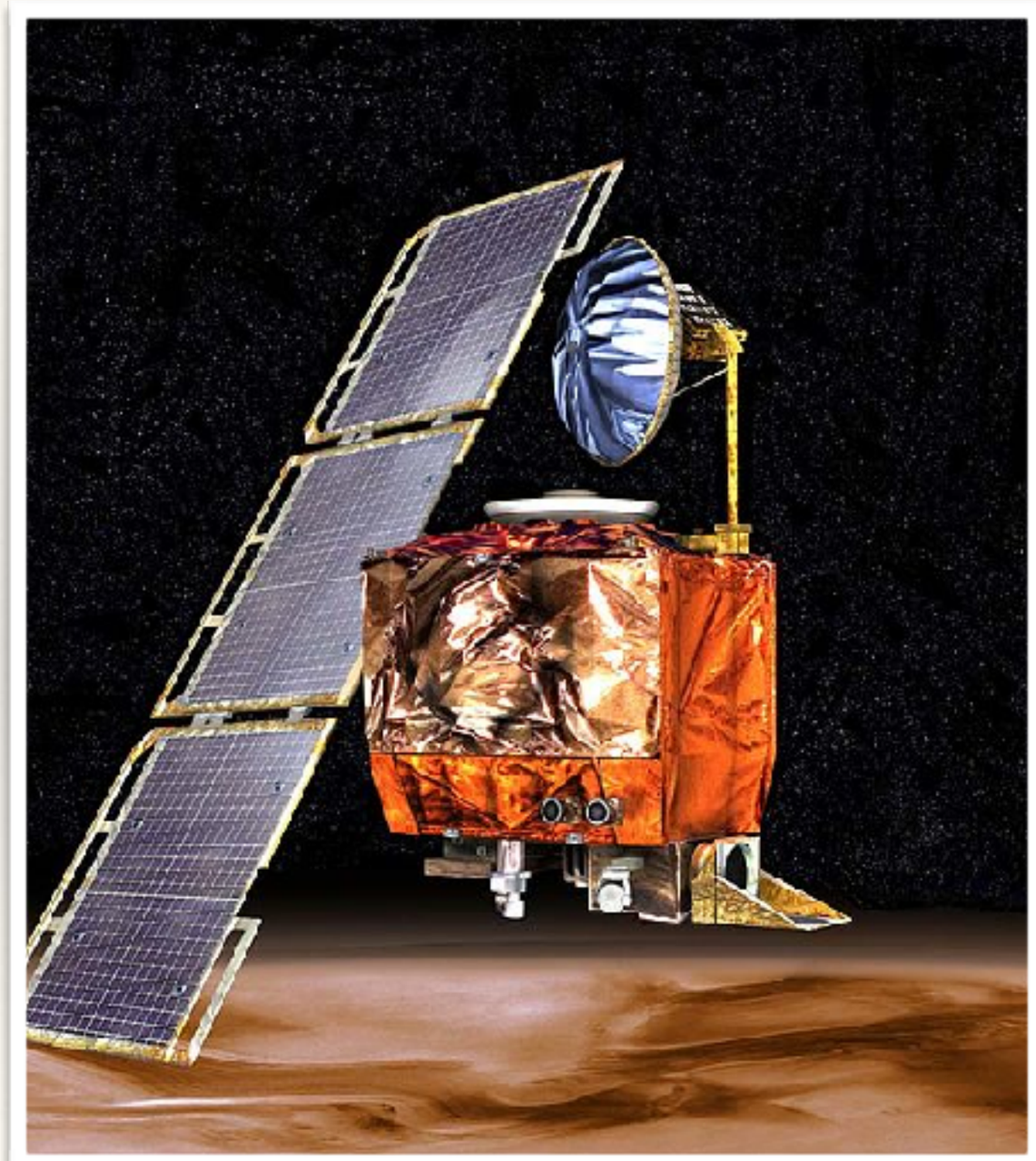
```
year = ORIGINYEAR; /* = 1980 */

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

Mars Climate Orbiter loss

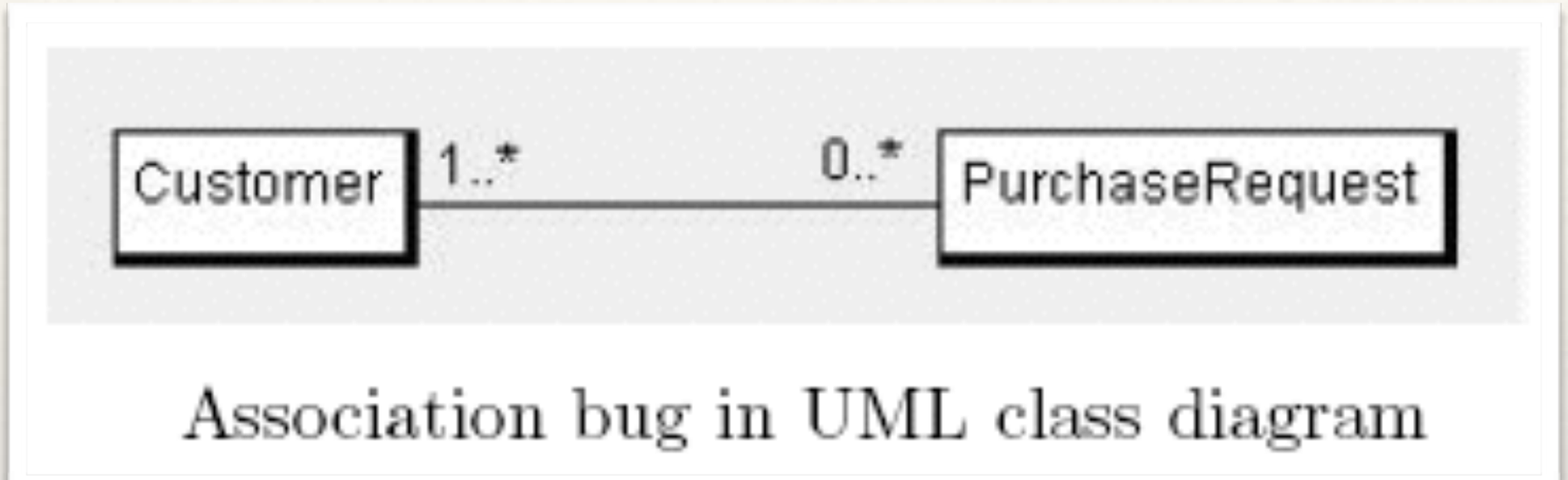
- Mars Climate Orbiter reached Mars (1999)
  - Lost after entering the atmosphere

- Ground software used “metric units” (‘Newton-seconds’) instead of “English units” (‘pounds-seconds‘)
  - Jet Propulsion Laboratory (JPL) and Lockheed Martin Astronautics (LMA) located in different places and used different units



Wrong generated code

omission



Association bug in UML class diagram

- Missing {ordered} constraint for association
  - in UML, the association by default is {unordered}
- Automated UML code generator used for generating Java code
  - java.util.HashSet generated instead of java.util.TreeSet by the tool
- Defect: PurchaseRequests given by customers processed in some unspecified order

Unreadable Customer ID

wrong datatype to save space

ICICI Lombard
—GENERAL INSURANCE—
WE KEEP YOU GOING

ICICI Lombard General Insurance Company Limited
Corp. Office: Zenith House, Keshavrao Khadye Marg Opp. Race Course,
Mahalaxmi, Mumbai - 400 034

RENEWAL NOTICE

RN Date : 12 May, 2007

Insured name : Mr. GANESH SAMARTHYAM GOVINDACHETTY

Insured Address : #1043 2ND FLOOR 7TH A MAIN
KORMANGALA 1ST BLOCK NEAR TAB
MALL BANGALORE KORAMANGALA
KORAMANGALA BANGALORE - 560034
KARNATAKA

Contact Nos :

CustomerID : 1.00E+11

Ref :- Renewal of your Private Car Vehicle, Policy Number " 3001/50020905/00/000 "

- Simple mistake - using wrong data type for representing Customer ID!
- Programmer wanted to save few bytes of space
  - Because a String object occupy few more bytes compared to a floating point number!

Stock market undervaluation

unintuitive floating-points

- In 1984, the Vancouver stock exchange was under-valued by 48% (index value was 524.811 instead of correct 1098.892) compared to its real value because of an accumulating error
  - Slow-accumulation of round-off error
  - 3 decimal places instead of 4 decimal places

http://en.wikipedia.org/wiki/Vancouver_Stock_Exchange

Practical and effective approaches for developing high-quality software

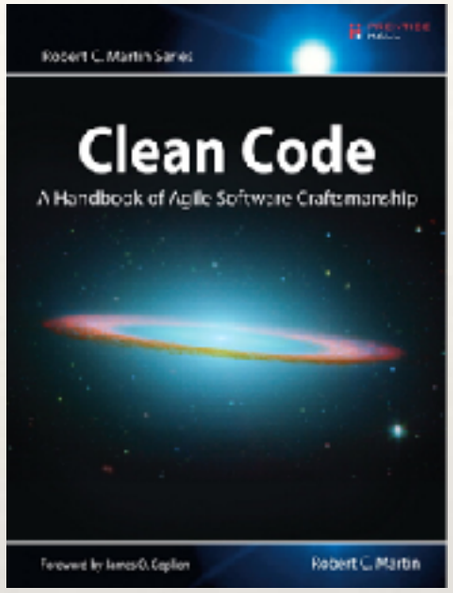
More and more testing? No!

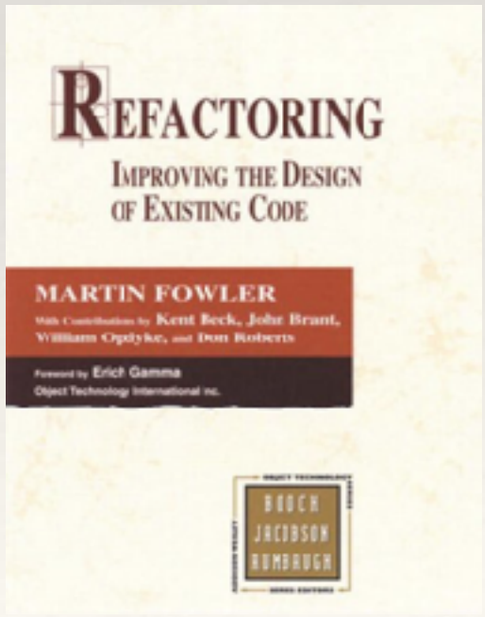
reactive,
costly, too late

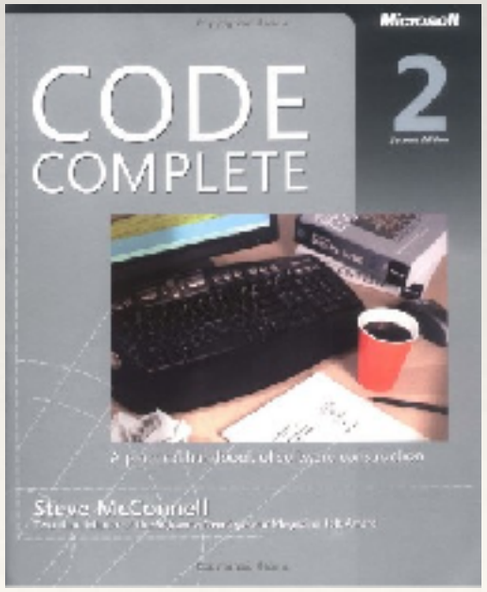
- Testing is necessary but not sufficient
  - "Program testing can be used to show the presence of defects, but never their absence" - Dijkstra
- Already testing takes 30% to 75% of development cost!
  - Considerably difficult to test complex software
  - Return-On-Investment (ROI) on testing goes down over time

Disciplined personal practices

- Study results from Barry Boehm & Victor R. Basili (Well-respected Researchers in Empirical Software Engineering): “Disciplined personal practices can reduce defect introduction rates by up to 75 percent”!
- Approaches for realising it: peer-reviews, automated on-the-fly analysers, incentivise writing quality code





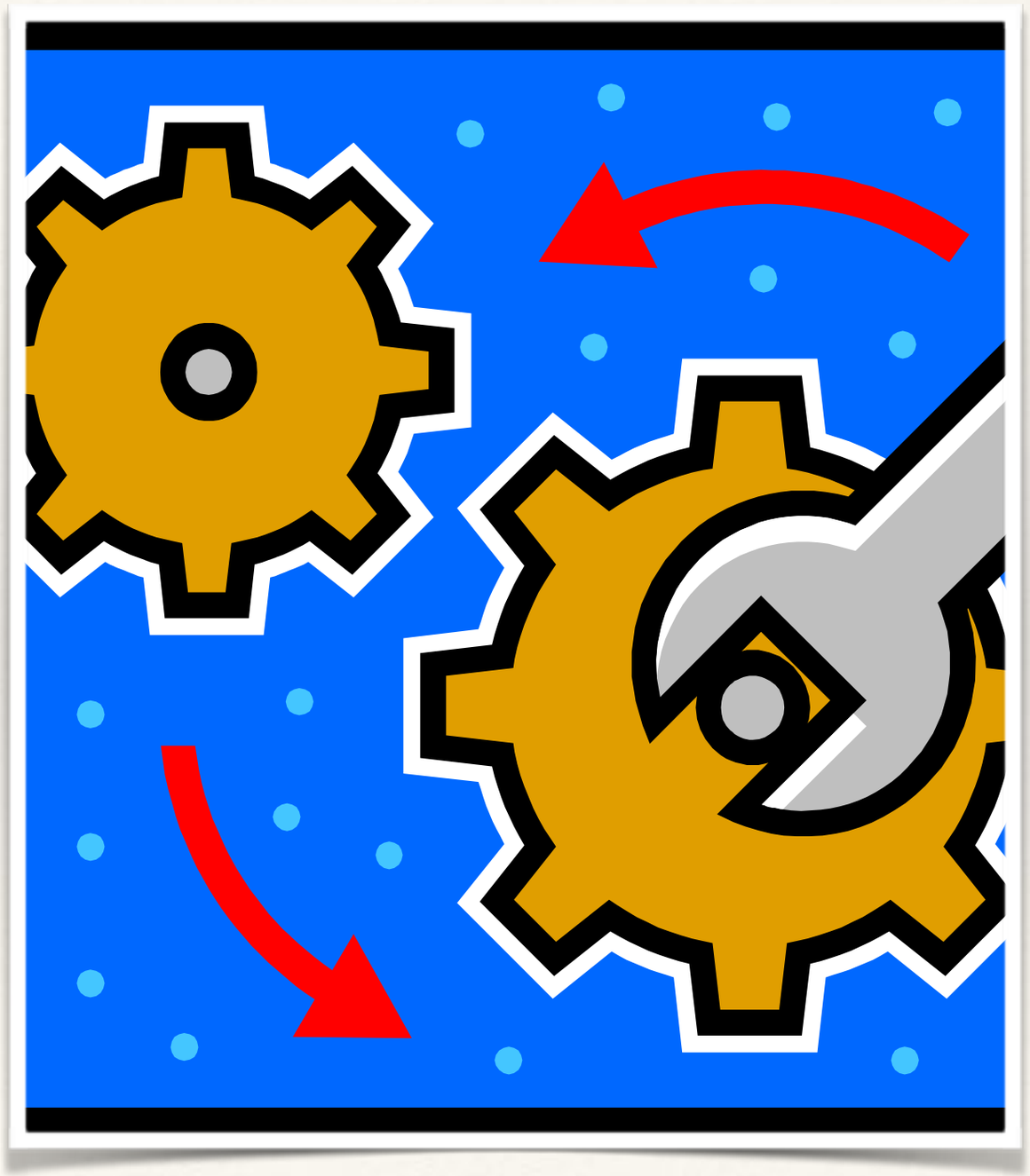


Solution #1. Extensive peer-reviews

- Just like daily exercises, peer-reviews are unattractive
- Time and again, studies show they are *most effective* in catching most defects
- Not just code reviews, but design and architectural reviews

Solution #2. Institutionalise automated analyzers

- Automated analysis tools can find up to 70% of the defects that can be caught using manual reviews!
- Not just code analysers - design, architecture, clone, and metric tools are equally important
- Not just integration, but institutionalise using them as part of the development process



A *must-to-have* list of useful analysers

- **Clone detectors (C++, Java, C#)**
  - PMD Copy Paste Detector
  - Simian
- **Metric tools (C++, Java, C#)**
  - RSM Metrics
  - Understand
  - McCabe Metrics
- **Architectural analyzers (C++, Java, C#)**
  - Sotograph
  - Structure101
- **Design analysers**
  - Stan4J (Java)
  - InFusion (C++, Java)
  - NDepend (C#)
  - JDepend (Java)
  - CppDepend (C++)
- **Code analyzers**
  - PMD (Java)
  - FindBugs (Java)
  - Goanna (C++)
  - PC-lint (C++)
  - FxCop (C#)
  - Gendarme (C#)
  - ReSharper (C#)

Marked in underscore are free tools

Solution #3. Encourage creation of quality code

- War story:
  - Crisis in my earlier company when two patches recalled (with customers screaming)
  - After dousing fire, management decided to improve processes to ensure creating quality code
    - Developers encouraged to file defects on software (number of defects filed considered in performance evaluation)
    - Target ratio of development:testing: customer defects set to 64:27:9
- No patch-recalls, reduced testing costs, customer filed defects reduced by 25%!

Solution #4. Quality first. Features next!

- Usual approach: Features first, quality next (or as after-thought)
- Perform impact-analysis seriously before accepting feature requests or making defect fixes
  - Example: Look out for architecturally-significant requirements

Summary & Key takeaways

- Why care about software quality?
  - Software defects cost huge money!
  - Software defects cause project failures
  - Software defects result in various kinds of losses
- Practical and effective approaches for developing high-quality software
  - Extensive (arch., design, code) peer-reviews
  - Institutionalize (arch, design, code, metric, and clone) analysers
  - Encourage and incentivise developers to create quality code
  - Adopt quality first, features next approach

email	sgganesh@gmail.com
website	www.designsmells.com
twitter	@GSamarthyam
linkedin	bit.ly/sgganesh
slideshare	slideshare.net/sgganesh