TwinSPIN: Future of Software Engineering

Mats Heimdahl
Mike Whalen
‘The Internet of Things’ is a concept originally coined and introduced by MIT, Auto-ID Center and intimately linked to RFID and electronic product code (EPC)

“... all about physical items talking to each other..”

Like RFID it is a concept that has attracted much rhetoric, misconception and confusion as to what it means and its implications in a social context
Interdependency

Outage of one power line affects 12 nations; 15 M Europeans (04/11/2006)

Slide from: R. Hutter: Vulnerability of SCADA Systems
SCADA cyber security threats

• **Insiders**
  – The “disgruntled employee” scenario, where a knowledgeable insider may be motivated to damage and/or to corrupt the system

• **Hackers** (benign, political, criminal)
  – Outsiders who want to break into SCADA systems because they are attracted by the challenge

• **Cyber Terrorists**
  – A SCADA system is the ideal target of well-funded terrorist groups seeking to cause widespread damage to a large portion of the population

• **States**
  – SCADA systems or sensitive infrastructures become entry points for a new type of power projection and attack
Security Incidents in SCADAs

**BCIT Industrial Security Incident Database (ISID)**

Statistics is one side, but what about quality?

*The Myths and Facts behind Cyber Security Risks for Industrial Control Systems*, Eric Byres, P. Eng., Justin Lowe
Speculations on Stuxnet
(Symantec; Kaspersky et al.)

Prime function: “To penetrate and disable dedicated SCADA systems”

<table>
<thead>
<tr>
<th>Motivation</th>
<th>Objective</th>
<th>Assessment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Political</td>
<td>Sabotage</td>
<td>most likely</td>
</tr>
<tr>
<td>Commercial</td>
<td>Espionage; a test case?</td>
<td>possible</td>
</tr>
<tr>
<td>Criminal</td>
<td>Gains, e.g. blackmailing</td>
<td>unlikely</td>
</tr>
<tr>
<td>Private</td>
<td>Demonstration</td>
<td>most unlikely</td>
</tr>
</tbody>
</table>
Many questions are still open on Stuxnet

<table>
<thead>
<tr>
<th>Question</th>
<th>Answer</th>
</tr>
</thead>
<tbody>
<tr>
<td>How big was the potential?</td>
<td>Very big to catastrophic; But no concrete scenarios....</td>
</tr>
<tr>
<td>Who developed it?</td>
<td>Effort, complexity and targets are indicators to a gov. controlled effort</td>
</tr>
<tr>
<td>How was it distributed?</td>
<td>Commonplace media: USB &amp; Internet</td>
</tr>
<tr>
<td>Why was its effect not really malicious?</td>
<td>Early warning? Good detection mechanisms? Political reasons not to activate?</td>
</tr>
<tr>
<td>Are we at the threshold of a CYBER Weapons armaments race?</td>
<td>IT security experts believe so; Governments are still reluctant; ...and the energy industry?</td>
</tr>
<tr>
<td>Where does Cyber war / terrorism go?</td>
<td></td>
</tr>
</tbody>
</table>

*Slide from: R. Hutter: Vulnerability of SCADA Systems*
The concept is also central to Commission thinking on RFID and associated research funding in Europe

“... a new phase of the Information Society – the Internet of Things in which the web will not only link computers but potentially every object created by mankind.” – Viviane Reding –
On RFID: The next step to The Internet of Things – Lisbon Conference 2007

Even in concept some thought has to be given to the implications of such statements in respect of population-partitioning of identifiable objects and connectivity dynamics – Analysis of the Concept!
How We Develop Software

Concept Formation

Requirements

Is object code:
- free from runtime errors?
- correct w.r.t. functional requirements?
- free from unintended functionality?
- fast enough?
Are requirements right?

Test

Analysis

Object Code

Implementation

12/9/2010

10
What’s Wrong With That?
The Zune Killer

Code to switch from ‘days since 1980’ to ‘years since 1980 + days in year’

```c
year = ORIGINYEAR; /* = 1980 */
while (days > 365) {
    if (IsLeapYear(year)) {
        if (days > 366) {
            days -= 366;
            year += 1;
        }
    } else {
        days -= 365;
        year += 1;
    }
}
```

What happens on the last day of a leap year?

Testing, even using a rigorous structural coverage metric such as MCDC, is very unlikely to catch this bug. MCDC + boundary value testing would catch it though.

Example from John Rushby: Introduction to Static Analysis for Assurance, UMN Summer Software Symposium, 2009
How We Will Develop Software: Static Analysis

- Concept Formation
- Requirements Specification
- Design
- Implementation
- Object Code
- Analysis: Timing Runtime Errors
- Analysis: Runtime Errors
- Test

Is object code:
- correct w.r.t. functional requirements?
- free from unintended functionality?
Are requirements correct?
Static Analysis

- Testing only samples set of possible behaviors
- And unlike physical systems (where many engineers gain their experience), software systems are discontinuous
  - There is no sound basis for extrapolating from tested to untested cases
  - So we need to consider all possible cases... how is this possible?
- It’s possible with symbolic methods
- Cf. $x^2 - y^2 = (x - y)(x + y)$ vs.
  
  $5*5-3*3 = (5-3)*(5+3)$
- Static Analysis is about totally automated ways to do this
Symbolic Encodings

year = ORIGINYEAR; /* = 1980 */
assume (days > 0)
while (days > 365) {
    if (isLeapYear(year)) {
        if (days > 366) {
            days -= 366;
            year += 1;
        }
    } else {
        days -= 365;
        year += 1;
    }
}
FAIL!
Approximations

- We were lucky that we could do the previous example with full symbolic arithmetic
- Usually, the formulas get bigger and bigger as we accumulate information from loop iterations (we’ll see an example later)
- So it’s common to approximate or abstract information to try and keep the formulas manageable
- Here, instead of the natural numbers 0, 1, 2, . . . , we could use
  - zero, small, big
  - Where big abstracts everything bigger than 365, small is everything from 1 to 365, and zero is 0
  - Arithmetic becomes nondeterministic
    - e.g., small + small = small | big
Approximations

- Allows analysis of 1M+ SLOC in minutes or hours (Coverity)
- Can be used on BIG things: Microsoft Windows 7 security
- But…many false positives
A Hasty Fix

Code to switch from ‘days since 1980’ to ‘years since 1980 + days in year’

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

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

- Passes Static Analysis (terminates; no runtime errors)
- Still incorrect! Increments year one day too soon for leap year!

Example from John Rushby: *Introduction to Static Analysis for Assurance*, UMN Summer Software Symposium, 2009
How we Will Develop Software: Static Analysis + Proof

Concept Formation

Requirements

Properties

Specification/Model

Analysis

Integration

System Test

Integration Test

Specification Test

Autogenerated Tests

Object Code

Does object code conform to specification model? Is it fast enough?

Analysis: Timing & Runtime Errors

Is model:
- correct w.r.t. functional requirements?
- free from unintended functionality?
Are requirements correct?

Is model:
- correct w.r.t. functional requirements?
- free from unintended functionality?
Are requirements correct?
Fixed Code and a Property

Code to switch from ‘days since 1980’ to ‘years since 1980 + days in year’
year = ORIGINYEAR; /* = 1980 */

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

ensures (year >= 1980 && days > 0 &&
         (days <= 365 ||
          (IsLeapYear(year) && days == 366)))
Example Requirement:
Drive the Maximum Number of Display Units Given the Available Graphics Processors

Counterexample Found in 5 Seconds

Checked 573 Properties - Found and Corrected 98 Errors in Early Design Models
CerTA FCS Phase I

- Sponsored by AFRL
  - Wright Patterson VA Directorate
- Compare FM & Testing
  - Testing team & FM team
- Lockheed Martin UAV
  - Adaptive Flight Control System
  - Redundancy Management Logic
  - Modeled in Simulink
  - Translated to NuSMV model checker

<table>
<thead>
<tr>
<th>Subsystem/ Blocks</th>
<th>Charts / Transitions / TT Cells</th>
<th>Reachable State Space</th>
<th>Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>Triplex voter</td>
<td>10 / 96</td>
<td>3 / 35 / 198</td>
<td>6.0 * 10^13</td>
</tr>
<tr>
<td>Failure processing</td>
<td>7 / 42</td>
<td>0 / 0 / 0</td>
<td>2.1 * 10^4</td>
</tr>
<tr>
<td>Reset manager</td>
<td>6 / 31</td>
<td>2 / 26 / 0</td>
<td>1.32 * 10^11</td>
</tr>
<tr>
<td>Totals</td>
<td>23 / 169</td>
<td>5 / 61 / 198</td>
<td>N/A</td>
</tr>
</tbody>
</table>

... for each of ten control surfaces

**Phase I Results**

<table>
<thead>
<tr>
<th>Effort (% total)</th>
<th>Errors Found</th>
</tr>
</thead>
<tbody>
<tr>
<td>Testing</td>
<td>60%</td>
</tr>
<tr>
<td>Model-Checking</td>
<td>40%</td>
</tr>
</tbody>
</table>

Slide © Rockwell Collins, 2008
The Problem with Proof

We analyze code long after it is written

Costly to fix bugs late

Our tools struggle to handle large programs

Humans are overloaded with mostly false warnings
The Solution

We analyze the code as it is written

Find bugs as they happen

Small programs to analyze efficiently

Humans are in the moment when confronted with possible errors
Conclusion

- Security and new hardware architectures (multicore) will force changes in Software Engineering
- Multicore requires parallel SW to exploit.
  - Parallel SW is very hard to get right with current languages and V&V tools
  - Research in both of these areas will be required
- Security will require systems be better “hardened” to a variety of attacks
  - Very difficult to test for exploits
  - Static analysis / formal methods can make a big difference
- Symbolic V&V methods are starting to become mainstream
  - Static analysis: Coverity, Klokwork, Astree
  - Model checking: Pex, Prover, Java Pathfinder (JPF)
- Fundamentals are still hard: requirements gathering, process issues
Backup Slides
The Issues

Analyses Human Factors

\[
\sin^2(x) = \frac{3}{4} \sum_{k=1}^{\infty} \frac{(-1)^k (1 + 9^k) x^{2k}}{(1 + 2k)!}
\]

\[
\sin^2(x) = \frac{1}{4} \sum_{k=0}^{\infty} \frac{(-1)^k ((3 + 9^k) \left(\frac{x}{2} + x^2\right)^k)}{(2k)!}
\]
Human FITE Interaction

Pair Programming: You x FITE

FITE: “Are you assuming OBJECT is not null here or did you forget the null check?”

You: “It is obviously not null!”

FITE: “So I’ll add a precondition then…it is not that I don’t trust you, but you remember what happened last time”
Required Analyses

Incremental and Compositional Symbolic Execution

+
User Configurable Domains

Unexpected Bugs

Numerical Precision

Performance

Security

```matlab
>> x = 9007199254740994.0;
>> y = 1.0 - 1/65536.0;
>> z = x + y;
>> z-x
ans =
     0.
```
Are we there yet?

We have point solutions but no integration
The primary purpose of the Stuxnet worm is to take control of industrial facilities. This is without any doubt the most sophisticated targeted attack we have seen so far. 

Percentage of Hits from W32.Stuxnet by Country

Source: Symantec 22July 2010; 72 hrs survey
One obvious Stuxnet target: Iran's Bushehr nuclear power plant

(UPI Photo/Mohammad Kheirkhah 12/9/2010)