Dependability Arguments with Trusted Bases

Eunsuk Kang and Daniel Jackson
Massachusetts Institute of Technology
An Investigation of the Therac-25 Accidents
Nancy Leveson and Clark Turner (1993)
Lesson Learned?

"...[NY state] most stringent regulator of radioactive medical devices in the nation...

"...621 mistakes from January 2001 to January 2009..."
Achieving Dependability

Conventional

Process & standard

Post-facto analysis

“All or nothing”

Software for Dependable Systems: Sufficient Evidence?
National Academies Study
Achiving Dependability

Conventional

Process & standard

Post-facto analysis

“All or nothing”

Missing direct link?
<table>
<thead>
<tr>
<th>Conventional</th>
<th>Proposed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Process &amp; standard</td>
<td>Explicit case</td>
</tr>
<tr>
<td>Post-facto analysis</td>
<td></td>
</tr>
<tr>
<td>“All or nothing”</td>
<td></td>
</tr>
</tbody>
</table>
# Achieving Dependability

<table>
<thead>
<tr>
<th>Conventional</th>
<th>Proposed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Process &amp; standard</td>
<td>Explicit case</td>
</tr>
<tr>
<td>Post-facto analysis</td>
<td>Design for dependability</td>
</tr>
<tr>
<td>“All or nothing”</td>
<td></td>
</tr>
</tbody>
</table>
## Achieving Dependability

<table>
<thead>
<tr>
<th>Conventional</th>
<th>Proposed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Process &amp; standard</td>
<td>Explicit case</td>
</tr>
<tr>
<td>Post-facto analysis</td>
<td>Design for dependability</td>
</tr>
<tr>
<td>“All or nothing”</td>
<td>Most critical requirements first</td>
</tr>
</tbody>
</table>
Methodology

Articulate critical requirements

Design for small trusted bases

Implement components

Check dependability argument

Analyze code
In This Talk

- Articulate critical requirements
- Design for small trusted bases
  - Implement components
  - Analyze code
- Check dependability argument
  - Modeling & analysis in Alloy
  - Formal framework
In This Talk

Articulate critical requirements

Design for small trusted bases

Implement components

Check dependability argument

Analyze code

formal framework

case study in electronic voting

modeling & analysis in Alloy
Trusted Base
Example: Radiation Therapy

Francis H. Burr Proton Therapy Center
Massachusetts General Hospital
Door Safety

When the door is opened, must insert the beam block
Trusted Base

Trusted base too large? Too many components to check?

- Door
  - Door Controller
  - Treatment HCI
  - Patient DB

- Beam Block
  - Beam Controller
  - Event Logger
  - Event Handler

Door Opened ⇒ Beam Blocked
Alternative Design

Simpler safety argument & greater confidence!

Key activity: Redesign to reduce trusted base
Formal Framework
Each system consists of:

- domains $D$ & machines $M$
- domain assumptions $A$ & machine specifications $S$
- requirements $R$
Formal notion

Each system consists of:

• domains $D$ & machines $M$
• domain assumptions $A$ & machine specifications $S$
• requirements $R$

Mapping from each requirement to its trusted base

$tb : R \rightarrow \wp(D \cup M)$
Formal notion

Each system consists of:

- domains $D$ & machines $M$
- domain assumptions $A$ & machine specifications $S$
- requirements $R$

Mapping from each requirement to its trusted base

$t_b : R \rightarrow \wp(D \cup M)$

Dependability argument for $R_i$

$A_{tb(R_i)} \land S_{tb(R_i)} \Rightarrow R_i$

where $A_{tb(R_i)} = \bigwedge_{d \in tb(R_i)} (A_d)$ and $S_{tb(R_i)} = \bigwedge_{m \in tb(R_i)} (S_m)$
Requirements Satisfaction

Traditional notion

\[ A \land S \Rightarrow R \]

Proposed notion

\[ \forall R_i \in R \cdot (A_{tb(R_i)} \land S_{tb(R_i)} \Rightarrow R_i) \]
Requirements Satisfaction

Traditional notion

\[ A \land S \Rightarrow R \]

Proposed notion

\[ \forall R_i \in R \cdot (A_{tb}(R_i) \land S_{tb}(R_i) \Rightarrow R_i) \]

Traditional notion often too hard to achieve!

- unrealistic assumptions (e.g. no operator mistakes)
- some specs too difficult or expensive
- simply too many components!
Separation of Requirements
Separation of Requirements

Design X
Separation of Requirements

Design X

Design Y
Decoupling in design cannot be achieved without decoupling in requirements!
Modeling & Analysis
Modeling & Analysis

Why model?
• articulate requirements, assumptions, specs
• check arguments & find mistakes

Alloy
• flexibility for encoding idioms
• support for automated analysis
• other languages may also be suitable!
abstract sig Property {}  
sig OK in Property {}  

abstract sig Domain extends Property {}  
abstract sig Machine extends Property {}  
abstract sig Requirement extends Property {}  
  trustedBase : set (Domain + Machine)  
}  

assert DependabilityArgument {}  
  all r : Requirement |  
    r.trustedBase in OK implies  
    r in OK  
}
Example: Reliable File Transfer

Requirement: Transfer a file from one host to another

*End-to-End Principles in System Design*
Jerome Saltzer, David Reed, and David Clark (1984)
Two FTP Designs

Standard Design

End-to-End Design
File Transfer in Alloy

abstract sig Packet {}
sig Block extends Packet {}
sig Hash extends Packet {}
sig File {
   blocks : set Block
   hash : Hash
}

Assign a property to a component

sig SenderFileSys extends Domain {
    file : File,
    app : SenderApp
}

sig Network extends Domain {
    packetsIn, packetsOut : set Packet
}

this in OK iff app.readFile = file

this in OK iff packetsIn = packetsOut
fact HashingProperties {
    all f1,f2 : File |
    f1.hash = f2.hash iff f1.blocks = f2.blocks
}

No two files have the same hash
sig ReliableTransferReq extends Requirement {
  senderFileSys : SenderFileSys,
  receiverFileSys : ReceiverFileSys,
  receiverApp : ReceiverApp
}

this in OK iff
senderFileSys.file.blocks = receiverFileSys.file.blocks or
receiverApp.receivedHash != receiverApp.computedHash

trustedBase = {senderFileSys + senderFileSys.app +
  receiverFileSys + receiverFileSys.app}
Analysis

```plaintext
assert DependabilityArgument {
    all r : Requirement |
    r.trustedBase in OK implies r in OK
}

check DependabilityArgument for 5
```

State the assertion to be checked
Network contrives the hash for the bad file!
Fixing the Model

```plaintext
fact HashingProperties {
  all f1,f2 : File |
  f1.hash = f2.hash iff f1.blocks = f2.blocks

  all h : Hash & Network.packetsOut |
  h in Network.packetsIn or no f : File | h = f.hash
}
```

New property of hashing added
Case Study
Electronic Voting

Two designs

• **Optical scanner system**: shown vulnerabilities
• **Scantegrity**: *end-to-end* (E2E) verifiable system

Case study

• modeled & analyzed the two designs
• argued why Scantegrity is more dependable
Voting System Designs

Optical Scanner System

Scantegrity
Related Work

Trusted Bases

- trusted computing base (TCB) (Lampson), security kernel (Popek78), separation kernel, safety kernel (Rushby)
- trust assumptions (Haley et. al)
Related Work

Dependability Cases

- Kelly (York), Knight (Virginia), Rushby (SRI), etc.
- Software Certification Consortium (USA)

Requirements Specification

- Problem Frames
- Goal-oriented approaches (KAOS, i*, Tropos, etc)
Conclusions

Articulate and prioritize most critical requirements

Design for small trusted bases, easier arguments

Analyze your arguments