Evolution of Computer-based Systems

Massimo Felici
LFCS, School of Informatics
The University of Edinburgh
UK

massimo.felici@ed.ac.uk
DIRC – www.dirc.org.uk

- Interdisciplinary Research Collaboration on Dependability of Computer-based Systems (DIRC)
  - 6-year project funded by the UK EPSRC
  - 5 UK universities involved + industry
- various Project Activities (e.g., Dependable Human Machine Interaction in Real-time Systems, Design for Dependability, Impact of Organizational Culture and Trust on Dependability, etc.)
- Research Themes: Structure, Diversity, Timeliness, Responsibility, Risk
Overview

- Evolution... of Computer-based Systems
  - Taxonomy of Evolution
    - ...Dependable Evolution
  - Requirements Evolution
    - A Case Study
A Taxonomy of Evolution Dependability Perspectives

Massimo Felici
Overview

- Evolutionary Dependability
- Evolutionary Layering
- Evolutionary Modelling
- Dependable Evolution
- Taxonomy of Evolution
  - Practical issues
  - Practical Challenges
Evolutionary Dependability?

- "Dependability is that property of a computer system such that reliance can justifiably be placed on the service it delivers."

- **Issue**: Static definition of Dependability

- **Issue**: A computer system can just be to a certain extent Dependable or Undependable
Dependability Basis

- Dependability consists of a set of attributes
- Dependability means:
  - fault prevention, tolerance, removal and forecasting
- Dependability impairments:
  - faults, errors and failures
  - others use an equivalent reshuffled model based on errors, faults and failures
Dependability implies Evolution
...but Evolution is not taken into account into Dependability
Evolutionary Layering

- Soft Evolution
  - Organization Evolution
  - Computer-based System Evolution
  - Requirements Evolution
  - Architecture (Design) Evolution
  - Software Evolution

- Hard Evolution

Evolution in Design
Evolution in Use
M. Lehman defines E-type programs (software systems), which:
- continuously evolve in order to be satisfactory
- need to accommodate environmental changes
- have an increasing complexity
- represent multi-level, multi-loop and multi-agent feedback systems
Architecture Evolution

- Architecture is (should be) a stable part of a system (Anderson and Felici, Safecomp 2000)
  - Issue: expensive to change
  - Issue: imply high risk
- Product-line architecture represents the extent to which a product-line will be able to evolve
  - Trade-off: General vs. Specific
  - In practice: Identified variability points
- Definition: unclear definition of architecture evolution. There are in general two (three) types of evolution;
  - the architecture evolves or components evolve (everything evolve).
Requirements Evolution

- It is impossible to frozen requirements. But we should (and could) identify the extent to which requirements evolve in order to identify the stable ones and the most likely to change. (Anderson and Felici, Safecomp 2000, Profes 2001, COMPSAC 2002)

- Requirements evolution has been considered mainly a management problem
  - little emphasis on product features
  - Research Hypothesis: Product features enhance our ability in understanding (requirements) evolution.

- Requirements evolution is unstructured
  - Research Hypothesis: There are structures for evolution (There are structures in Requirements, Changes and Evolution)
Computer-based System Evolution

- Did I speak just about software?
- Holistic viewpoint
Computer-based System Evolution

- Social Learning:
  - Domestication and Innofusion
- Emergent behaviour

- Distributed Cognition:
  - Human Cognition as result of interactions (knowledge distribution) between (among) individuals and artefacts

- Disappearing Computer (D. Norman):
  - The system (design) becomes less intrusive and evolves according to human needs
Organization Evolution

- (System) Evolution and (Organization) Co-evolution or vice versa

- System-Organization reflection (Conway's Law)

- Evolution as Corporate Knowledge
Evolutionary Modelling

- Software Evolution
  - E-type systems
  - code modification
- Architecture Evolution
  - product-line variability
- Requirements Evolution
  - Goal-structures framework based on first-order logic with Prolog resolution (Proteus Project)
  - First-order logic model of Requirements Evolution (Zowghi)
  - Traceability
- Management process
- Quantitative approaches:
  - (few) Metrics; (little) Probabilistic Distribution
Evolutionary Modelling

- Computer-based System Evolution
  - Human Evolution
    - Social Learning
    - Emergent Behaviour
- Organization Evolution
  - Business models ??? ;-((
  - Software Engineering Economics (e.g., COCOMO II Boehm)
Dependability Related Modelling

- Reliability Growth models
- Bayesian (probabilistic) models
- Fault Tree Analysis
- Event Tree Analysis
- Domino effect
- Cheese model
- ??? ;-(

...
A Taxonomy of Evolution

### Dependability Perspectives

<table>
<thead>
<tr>
<th>Evolution</th>
<th>Dependability Perspective</th>
</tr>
</thead>
<tbody>
<tr>
<td>Software Evolution</td>
<td>Software evolution can affect dependability attributes (e.g., Reliability). Nevertheless software evolution can improve dependability attributes by faults removal and maintenance to satisfy new arising requirements.</td>
</tr>
<tr>
<td>Architecture (Design)</td>
<td>Architecture evolution is usually an expensive phenomenon. It does not affect directly dependability, but there is high risk if the evolution process is unclear and little understood. Architecture evolution may be needed to support specific system properties (e.g., redundancy, performance, etc.).</td>
</tr>
<tr>
<td>Evolution</td>
<td>Requirements evolution does not directly affect dependability, but non-effective management of the requirement process may allow undesired changes to fall down into the product affecting its dependability. On the other hand requirements evolution may enhance system dependability across subsequent releases.</td>
</tr>
<tr>
<td>Computer-based System</td>
<td>System evolution may give rise to undependability. This is due to incomplete evolution of system resources. Evolution of some resources (e.g., software) should be taken into account by the other resources (e.g., liveware and hardware) in order to register a new configuration for the system. Hence the interactions among resources serve to effectively deploy a new system configuration. On the other hand human can react and learn how to deal with undependable situations, but continuous changes in the system configuration may give rise to little understanding about the system. Hence the human-computer interaction may become quite undependable as well.</td>
</tr>
<tr>
<td>Organization Evolution</td>
<td>Organization evolution should reflect system evolution. Little coordination between system evolution and organization evolution may give rise to undependability.</td>
</tr>
</tbody>
</table>
A Taxonomy of Evolution

- **Practical Issues**
  - There are many different assumptions about evolution embedded into methodologies
  - Unclear definition of evolution
  - Little coordination giving rise to undependability in place
  - Evolutionary data are difficult to capture and analyse
    - Incomplete, distributed, unrelated, unclear,...
A Taxonomy of Evolution

- Practical Challenges
  - Classify environments according to evolutions in place
  - Model evolution
  - Link evolutionary layers
  - Link evolutionary models
Requirements Evolution
An Avionics Case Study

Massimo Felici
Overview

- Why Requirements Evolution?
- Case Study
  - System Requirements
  - Development Process
  - Requirements related Issues
- Empirical Analysis
  - General Requirements Evolution
  - Taxonomy of Requirements Changes
  - Requirements Maturity Index
  - Ageing Requirements Maturity
    - Empirical evidence
  - Functional Requirements Evolution
  - Requirements Dependencies
  - Visualizing Requirements Evolution
Requirements Evolution Rationale

- Requirements Evolution
  - affects:
    - cost (both in term of money and man-power)
    - project risk
    - system dependability (and quality of service, system quality, etc.)
    - process effectiveness
- So far... little investigated and supported
- Current practice: reactive (passive); non-proactive (e.g., traceability)
- Differently:
  - Requirements Evolution as system feature
An Avionics Safety-Critical Case Study

- Safety Requirements
- Certification and Maintenance
- Product Line Aspects and Standards
- Functional and Operational Requirements
- Software Development Process
An Avionics Safety-Critical Case Study
An Avionics Safety-Critical Case Study

- Requirements related issues:
  - Requirements changes are collected just for certification
  - Changes are collected by a simple table
  - None support to analyze changes
  - Changes management is process oriented
  - None standard classification for describing changes
  - ...

Measuring Requirements Evolution

…so Requirements evolve... over subsequent releases

There is a majority of added (new) requirements among changes
Requirements Evolution
Taxonomy of Requirements Changes

- Add, Delete and Rename parameters / variables
- Range Modification
- Hardware Modification
- Partial Compliance
- Add, Delete and Modify Requirements
- Explanation
- Rephrasing
- Traceability
- Non-compliance
Requirements Evolution

Classified Requirements Evolution

![Chart showing requirements evolution over software releases]

- **Red** bars represent requirements added, deleted, and modified.
- **Green** bars represent requirements added, deleted, and modified with the note: (No Change to delivered code permitted).

The chart illustrates the number of requirements changes across different software releases.
Requirements Evolution

Classified Requirements Evolution

Changes having permission to change delivered code

Changes not having permission to change delivered code
...hence the RMI:
- is too sensitive to changes introduced in a single release
- does not take into account historical information about changes (e.g., age)
Ageing Requirements Evolution

Measuring Requirements Evolution

\[ AR_C = \frac{CR_C}{n} \]

\[ RSI = \frac{R_T - CR_C}{R_T} \]

\[ HRMI = \frac{R_T - AR_C}{R_T} \]

Average Number of Requirements Changes

Cumulative Number of Requirements

Requirements Stability Index

Number of Software Releases

Historical Requirements Maturity Index

Total Number of Requirements
Ageing Requirements Evolution

Measuring Requirements Evolution

Requirements Changes

RMI

RSI

HRMI

Total Number of Requirements
Ageing Requirements Evolution

- Average Number of Requirements Changes
- Requirements Stability Index
- Historical Requirements Maturity Index
Measuring Requirements Evolution

Functional Viewpoint

![Graph showing cumulative requirements changes against number of requirements]
The functional requirements have different evolutions... this may be due to requirements' dependencies. Intuitive notion of stability based on the proportion between changes and requirements.
Requirements Dependencies

Functional Viewpoint

<table>
<thead>
<tr>
<th></th>
<th>F1</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>F2</td>
<td>2</td>
<td>F2</td>
</tr>
<tr>
<td>F3</td>
<td>3</td>
<td>F3</td>
</tr>
<tr>
<td>F4</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>F5</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>F6</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>F7</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>F8</td>
<td>1</td>
<td>4</td>
</tr>
</tbody>
</table>
Visualizing Requirements Evolution

F1 - Software Architecture

Add[n]  
Del[n]  
Mod[n]
Conclusions & Further Work...

Design...
...for Evolution...

Evolve...
...for Dependability...