18th IEEE Requirements Engineering Conference
27-Sep to 1-Oct, 2010
Requirements Engineering @ Intel

- few dedicated requirements engineers
- central training / coaching department
- still using Word / Excel for many projects
- using PLANGUAGE to specify quality requirements
Requirements vs. Design

Many products carry the majority of their specifications forward from previous versions; Therefore,

It’s not whether a statement is a “requirement” or a “design” that matters, but whether the statement places appropriate constraints on the people that will read it

If the system must be or act a certain way, say so… If not, leave the people downstream as much freedom to do their jobs as possible

Copyright © 2010 Intel Corporation. All rights reserved.
## Writing Functional Requirements: EARS

A recent refinement of the generic syntax is the *Easy Approach to Requirements Syntax (EARS)*, created at Rolls Royce PLC*

EARS contains patterns for specific types of functional requirements

<table>
<thead>
<tr>
<th>Pattern Name</th>
<th>Pattern</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ubiquitous</td>
<td>The &lt;system name&gt; shall &lt;system response&gt;</td>
</tr>
<tr>
<td>Event-Driven</td>
<td>WHEN &lt;trigger&gt; &lt;optional precondition&gt; the &lt;system name&gt; shall &lt;system response&gt;</td>
</tr>
<tr>
<td>Unwanted Behavior</td>
<td>IF &lt;unwanted condition or event&gt;, THEN the &lt;system name&gt; shall &lt;system response&gt;</td>
</tr>
<tr>
<td>State-Driven</td>
<td>WHILE &lt;system state&gt;, the &lt;system name&gt; shall &lt;system response&gt;</td>
</tr>
<tr>
<td>Optional Feature</td>
<td>WHERE &lt;feature is included&gt;, the &lt;system name&gt; shall &lt;system response&gt;</td>
</tr>
<tr>
<td>Complex</td>
<td>(combinations of the above patterns)</td>
</tr>
</tbody>
</table>

*3rd party brands and names may be claimed as the property of others.*
Additionally

1. always challenge ubiquitous (generic) requirements (as they normally aren’t as generic as the author originally thought)

2. requirements pairing – aids documentation of Wanted vs. Unwanted behaviours
Nokia has introduced Agile methods in many projects
no 2 teams agree on what “agile” means
has been very successful in some areas
has been disastrous in other areas: one group had no output for 18 months

benefits: … see Agile literature …
risks: take away control over who implements what
Agile development – the Nokia experience 2/2

- **works well where**
  - project has many independent functions ("user stories"), which can be nicely separated (e.g. web application with many independent pages)
  - an existing, fixed architecture is used (e.g. Ruby on Rails)
  - a single product is developed
  - every developer is competent to change any piece of code – i.e. all in one technical domain (e.g. all UI stuff)

- **works badly where**
  - cross-cutting concerns are important (e.g. battery life)
  - an architecture needs to be built or redesigned
  - many technical domains are involved (UI, DSP, GSM, …)
  - a product line is developed
  - cost per unit is an issue
1. Problems came when people used agile as an excuse to “short cut”.

2. Large Projects either:
   1. use a “dictator model” (one architect decides) or
   2. use something like Scrum of Scrum

3. Scale:
   1. Large architectural changes need waterfall
   2. Small incremental changes work well with Agile (once the architecture has taken care of critical requirements)

4. Method:
   1. Take a standard method and use it.
   2. Ensure development streams share the responsibility for the overall product
Kano categories

- relates degree of implementation to customer satisfaction
- **must-have**: dissatisfied if absent, neutral if present
- **desired**: dissatisfied if absent, satisfied if present (more is better)
- **differentiator**: neutral if absent, satisfied / excited if present

- can be used to describe / rate
  - usage scenarios
  - features
  - requirements
Requirements-aware systems

- “Classic” systems are built to (static) requirements, which describe a (static) world
- Adaptive systems can function in a changing environment
- A representation of requirements or (conflicting) user goals at runtime allows machine reasoning about e.g. tradeoffs
Physics of Notations: A theory for visual notation design

1. A repeat of a one day tutorial Bernd reported on last year
   1. a rapid summary..
   2. the focus is on how we can design notations that are understandable with the minimum mental load
Physics of Notations: A theory for visual notation design

1. Quantifies how shape, size, colour, orientation, texture & horizontal and vertical position should be used for optimum use of perceptual processing

2. The result is that the reader
   1. needs to do less cognitive processing (less mental load) to interpret diagrams
   2. and is better able to think about the content (a good thing)

3. Most notations in use are in many respects very poor at this
   1. including UML / BPMN
   2. “If we had this work available in the early 90’s UML would look very different today” - Grady Booch
Physics of Notations: A theory for visual notation design

1. we can learn a lot from diagramming experts
   1. cartographers have practiced their craft for many years!!

2. The “Moody Dream Team” for new notations would include a Cartographer, a Human Computer Interaction person, a Psychophysicist and ….

3. Moody’s paper is available on paper and as a PDF via the IEEE
   1. unusually they allowed an author to exceed their 10 page limit (this is 22)
   2. but they published it in monochrome (so they hadn’t read the paper)
   3. … and they also went a long way out of their way to publish the PDF in monochrome!
   4. we have a colour PDF if you are interested and if you do print it, colour is good.
Successful Deployment of Requirements Traceability in a Commercial Engineering Organization…Really

1. Reported the results of a survey of engineers most involved in traceability at Teradyne

2. The majority of engineers see traceability as adding value (68%)

3. However…. 

Benefits of Traceability

- Prevents missed lower-level requirements
- Helps in understanding rationale
- Increases confidence in requirements
- CMMI
- Change Management
- Prevents implementing what is not required
- Root cause analysis when requirement missed

*Expected vs. Not Expected*
1. How do engineers use traceability

---

**Trace Matrices**

- Doesn’t scale well
- Correctness of Trace??

---

**Trace Tables**

- Missing traces
- Named requirements somewhat convey correctness
- Automatically generated
- Part of Requirements Document (at end)
Selected conclusions and assertions. Traceability:

- is perceived as providing value
- ... however, not always in the ways expected
- is a tool to improve requirements
- ... however, it is not a deliverable in and of itself
- is inherently more ambiguous than the traced requirements
- can be a valuable tool
- ... however you cannot deliver perfectly correct traceability

Conclusions:

- traceability should be used as part of engineers daily work
- recognize that different groups will use / value traceability differently