Case-Based Stories for Traceability
Education and Training

Olly Gotel
olly@gotel.net

Stephen Morris
sjm@city.ac.uk
Motivation

• Learning from other disciplines
  – Real - accessible - see value
  – Proven techniques
  – Role of case studies

• Engaging students in software & systems
  traceability is hard
  – Lack of case studies
  – But, fun to write & tell stories
The REET paper

- Story about late night RE’12 munchies
- A context for tracing
- Epidemiological tracing process
- Food tracing process
- Analogy & discussion questions
- A model to emulate?
The REET session

1 - Detecting a possible outbreak
2 - Defining and finding cases
3 - Generating hypotheses
   * Hypothesis-generating interviews
   If cases continue
   Not finding associations between food & illness
   Finding associations between food & illness
   * Unsolved mystery
4 - Testing hypotheses
   * Analytic studies
   * Laboratory testing of samples
5 - Finding the point of contamination and source of the food
6 - Controlling an outbreak
   * Recall product(s)
   * Remove source of contamination
   * Revise production process
7 - Decide an outbreak is over

Cases stop

www.cdc.gov/outbreaknet/investigations
Back story
Analogy

Pilots and controllers can be located anywhere in the world.

Internet

C-FJDY

KNTS44

VATSIM Servers (FSX)

Boston Tower

Sydney Approach

Air Traffic Chaos

ERROR

www.physiczone.com - VS2003
Software systems ...

- Do we have a monitoring & early warning system for problems?
  - A way to recognize atypical situations?
  - An authority to notify?
- Of what does/should such a system comprise?
  - What real-time data is needed to establish trends?
  - Whose role would it be to coordinate this?
- Can we recognize when symptoms of a problem are requirements-borne?
- Any professional agencies or processes to guide how to investigate & act?
Detecting a possible outbreak

1. Detecting a possible outbreak
2. Defining and finding cases
3. Generating hypotheses: generating hypotheses
4. Testing hypotheses: analytic studies, laboratory testing of samples
5. Finding the point of contamination and source of the food
6. Controlling an outbreak
7. Decide an outbreak is over

Finding associations between food & illness

Not finding associations between food & illness

If cases continue

* Unsolved mystery

* Hypothesis-generating interviews

* Analytic studies
* Laboratory testing of samples

* Recall product(s)
* Remove source of contamination
* Revise production process

Cases stop
Software systems ...

• Do we routinely collect historical data & use statistical knowledge of “background activity” to inform analyses?
  – What historical data is useful to aid diagnosis?

• Are there repositories of known requirements-borne problems (i.e., well-defined characteristics that can be tested for, incubation periods, typical patterns of behavior & scope of impact)?

• What checks & balances can we employ prior to investing time & effort in any tracing investigation?

• How well do we understand what we trace?
  – Do we make an agreed diagnosis of any requirements-borne problems we investigate?
Defining & finding cases

1. Detecting a possible outbreak
2. Defining and finding cases
3. Generating hypotheses (interviews generating hypotheses)
4. Finding hypotheses
5. Analyzing hypotheses (analytic studies, laboratory testing of samples)
6. Testing hypotheses (analytic studies, laboratory testing of samples)
7. Finding associations (between food & illness)
8. Not finding associations (between food & illness)
9. If cases continue
10. If cases stop

- Unsolved mystery
- Generating hypotheses
- Hypothesis-generating interviews
- Defining and finding cases
- Detecting a possible outbreak
- Controlling an outbreak
- Recall product(s)
- Remove source of contamination
- Revise production process
- Finding the point of contamination and source of the food
- Testing hypotheses
- Analytic studies
- Laboratory testing of samples
- Controlling an outbreak
Software systems ...

• If a multidisciplinary team is responsible for creating software, what team should identify & address a requirements-borne problem within it?
  – What skills are required?
• What data needs to be collected to advance an investigation & how are these obtained?
• What tools are used to examine these data from multiple perspectives & support analyses?
• Do we characterize “cases” to identify, with certainty, those errors or anomalous behaviors caused by the same requirements-borne problem?
• Are temporary fixes for software monitored for effectiveness & analyzed for impact elsewhere?
Generating hypotheses (sources)
Software systems ...

• Do we form multiple hypotheses about a problem or tend to converge on a source?
• Do we collect sufficient data to reconstruct activities that led to a problem (i.e., use of software over time by numerous parties in different contexts)?
  – How do we account for evidence that is missing?
  – What could be done earlier to avoid a paucity of record?
• If the investigation leads to a function in a software component that is capable of generating new cases of the problem, what can we do to ensure its “withdrawal” is unproblematic?
Testing hypotheses
Software systems ...

- What kind of hypotheses could we form & test about a requirements-borne problem?
  - What added value is there to investigating the development processes employed & wider environments?
- What would be the equivalent of a case control study in the air traffic control story?
- Do we use statistics to confirm correlation between a problem & a candidate requirements source?
  - What do we do with noise data?
Finding point of contamination

1. Detecting a possible outbreak
2. Defining and focusing
3. Generating hypotheses
4. Finding the point of contamination and source of the food
5. Testing hypotheses
6. Generating interviews
7. Finding associations between food & illness

* Unresolved mysteries
* Hypothesis-generating interviews
* Defining and finding cases
* Detecting a possible outbreak
* Controlling an outbreak
* Decide an outbreak is over
* Recall product(s)
* Remove source of contamination
* Revise production process
* Generate hypotheses
* Define and find cases
* Detecting a possible outbreak

Cases stop
Software systems ...

- What could/should the requirements engineers have done to prepare for such an investigation?
- What are the roles & responsibilities of other stakeholders in the development chain?
- What agreement is there upon data that need to be collected, recorded & shared about traceable items (e.g., requirements) in different software domains?
  - How do we do this?
- If recreating a traceback later, & automated techniques are unavailable, how do we know we have located the correct starting point?
  - What guidance is there for undertaking tracing retrospectively & manually?
  - When would regulatory agencies get involved?
Controlling the outbreak

1. Detecting a possible outbreak
2. Defining and describing the outbreak
3. Generating hypotheses and identifying candidate food items
4. Finding associations between food and illness
5. If cases continue: Extend investigation
   - Finding associations between food and illness
       - Hypothesis-generating interviews
6. If cases stop: Decide an outbreak is over
   - Generating hypotheses
       - Analytic studies
       - Laboratory testing of samples
7. Testing hypotheses
   - Analytic studies
   - Laboratory testing of samples

- Recall product(s)
- Remove source of contamination
- Revise production process

Cases stop
Software systems ...

• How easy would it be to affect a requirements recall in the analogous story?
• Do we simply remove a requirement & all the components it touches, or do we alter it in some way & attempt to remedy the knock-on problems in all affected components?
  – Is our domain easier / harder / just different?
• What tools do we have to gain transparency into efforts & assess their effectiveness?
• Do software consumers have an equally vital role to play in the process?
Deciding the outbreak is over
Software systems ...

- Requirements, & their satisfaction in design & code, may be used in other systems, so a problem identified & addressed may re-emerge
  - Do we continue to gather data to trace & measure once a problem is “closed”?
- Do we share experiences of tracing to advance understanding & improve practice?
  - How?
  - What do we learn?
Epilogue
Meta-level observations

• Did you see the role & value of tracing?
• Did it stimulate broader discussion?
• What worked?
  – What contributed to making it work?
  – Detail & structure?
• What didn’t work?
  – Why was that?
  – Did the facilitator compensate?
• How could we make it better?
  – Video
  – Self-study versus group
Criteria for our story

• Reality (real case study data with detail)
• Traceability role explicit (interest, value & urgency)
• Set context, with steps & hooks (to discuss current & future practices)
Case-Based Stories for Traceability
Education and Training

Viable?  Effective?  Useful?
What’s the alternative?