Scenario Synthesis from Imprecise Requirements

Bill Mitchell, Robert Thomson, Paul Bristow
Enterprise Development Process

Groups
- Customers
- Technical Marketing
- Feature Teams
- Box Teams
- Integration Teams
- Testing Teams

Feature
- Technical Marketing Requirements
- High Level Designs
- Functional Requirements
- Detailed Designs

System
- System Requirements
- Component Requirements
- Architecture Requirements
- Standards
Telecoms Example

• Network provider deploying 3G.

• Placing order for handsets.

• One of the many features included will be access to network Java game repository.
Initial Customer Requirements

User

Handset

Network

Menu Key

Options

Select

Confirmation

Fetch Resource

Java Game
Customer Scenario broken down into sequence of atomic events, which change interface functionality.

Each event modifies functionality and UI configuration.

Java Games
- Doom 8
- Quake 9
- etc......
Functional Requirements

User -> Phone

Java App Download

key_press(java_menu)

Phone -> Browser

Idle -> Inactive

select(option)

Browser

Download

load(URL)

Load File

Active

Download File

Download

Display Notification

download_OK

Inactive
Technical Marketing Scenarios

Normative scenarios are very focused on isolated behaviour of feature in these requirements:

• What if voice or data call received during download?
• If memory is expandable (as with some PIM-phone hybrids) how should the mem-full error be handled if the user could add extra memory with, say, a USB flash memory stick?
• What if during the download the network service provider tries to update the phone configuration via the air interface for enhanced game play?

Need to synthesise model of system from all MSC requirements scenarios for simulation and analysis.

Problem:
• Practitioners use states imprecisely
• Different engineering groups define scenarios differently
• Legacy requirements
Deadlock example from TETRA PPT

ruthless pre-empt

agreed pre-empt

FSA for A

FSA for B
Example Deadlock Avoided

Composite States
- Anonymous internal states
- Multiple entry/exit states

Too Weak to ever give any interactions!
Example, Call Waiting from paper in FIW 2000
Example, RBWF, from paper in FIW 2000
Example, FI from paper in FIW 2000

Whenever in these composite states CW can happen
Trace semantics for states

State $x$ is $(\text{In}, \text{Out})$, where $\text{In}$ and $\text{Out}$ are sets of traces.

For every trace $t_1$ of $\text{In}$ there is a path

\[ u \xrightarrow{t_1} x \]

some initial state $u$

For every trace $t_2$ of $\text{Out}$ there is a path

\[ x \xrightarrow{t_2} y \]

some accepting state $y$
Deterministic trace semantics

For any $t_1$ of $\text{In}$ if there is a path $u \xrightarrow{t_1} x$ for some initial state $u$ then for every trace $t_2$ of $\text{Out}$ there is a path $x \xrightarrow{t_2} y$ for some accepting state $y$. 
MSC trace semantics for exit/entry states

Every MSC trace $t$ can be split into pairs $(t_1, t_2)$ where $t_1$ leads to exit state.

For any $t_1$ if there is a path

\[ u \xrightarrow{t_1} x \]

for any state $u$ then there is a path

\[ x \xrightarrow{t_2} y \]

for some state $y$
State semantics
Overlapping Processes, continued

Scenario 1, machine for A

Scenario 2, machine for A
Overlapping Composition of Processes

P trace simulates Q when:
given any (state annotated) execution traces t1 and t2:

\[ P \xrightarrow{t1} P1 \]
\[ Q \xrightarrow{t2} Q1 \]

where t1 matches t2, then P1 must be able to simulate Q1
Livelock from naive composite state semantics

DFSA for A
Exit State transition matching

P trace simulates Q when:

given any (state annotated) execution traces $t_1$ and $t_2$:

$$
P \xrightarrow{t_1} P_1$$

$$
Q \xrightarrow{t_2} Q_1$$

where $t_1$ matches $t_2$,
and $t_1$, $t_2$ have reached exit states
then $P_1$ must be able to simulate $Q_1$.

where $t_1$ matches $t_2$,
and $t_1$, $t_2$ have reached entry states
then $P_1$ must be able to simulate $Q_1$. 
Temporal contexts for defining matching traces

$\mathcal{X} \vdash (\parallel t \parallel \Rightarrow \lozenge \parallel t' \parallel)$

$\square([\text{load} (\text{URL})](\text{Active} \Rightarrow (\text{‘Load File’ } \cup \text{ Inactive})))$
Download File with Browser

Phone
- Download
  - load(URL)
    - Active
      - Resolve URL
        - Connect(URL)
          - Data
            - send(file)
      - Inactive
  - Load File
    - download OK
      - Display Notification

Browser
- Inactive
  - Download
    - Channel

Air Interface
- Channel
Overlap of Java Game and Browser Download
Error Check

Will have universal scope over exit states
Overlap Java App + Browser + Error Check

User → Phone
- Java App Download
  → File
  → key_press(java_menu)
  → Java Menu
  → select(option)
  → Download
  → load(URL)
  → Load File
  → Active
  → Resolve URL
  → Connect(URL)
  → Data
  → send(file)
  → Check File
  → Check for Errors

alt
- Error Found
  → corrupt_file(file)
  → Display Failure Notice
  → Inactive
  → Load File
  → download CK
  → Display Notification
  → Inactive

Browser → Air Interface
Questions