Trader: An Industry-as-Laboratory Experiment in System Dependability

Ed Brinksma

Embedded Systems Institute

Second International Workshop on
Foundations of Component-based Design

Embedded Systems Week, Salzburg, September 30th, 2007

TRADER
System Dependability

NXP founded by Philips

Carrying Industrial Partner

Period: Sept. 2004 - Aug. 2009

20 fte/yr, 7 PhDs,

1 Postdoc, 10 Partners

TRADER

PHILIPS
Consumer Electronics

TU Delft
Delft University of Technology

TU/e technische universiteit eindhoven

Universiteit Leiden

DESIGN TECHNOLOGY INSTITUTE

imec

University of Twente
The Netherlands

Embedded Systems INSTITUTE

NXP founded by Philips
Research

TASS software professionals

Goal

Develop methods and tools to optimize dependability of high-volume products.

Issues

- Minimize product failures.
- Increase user satisfaction (user-centric design approach)

Dependability threats in TV domain

Increasing complexity

- ❑ **Functions/content increases rapidly**
  - – *Play music, view photos, search teletext, Electronic Programming Guide, child lock, sleep timer, Picture-in-Picture, TV ratings, emergency alerts, many image processing options and user settings, …*
- ❑ **External information sources multiply**
  - – *Connected planet strategy, downloadable applications*

➔ **Increase of SW (1KB in 1980 – 64MB in 2007)**
Increase of third party content (EPG, codec's)

Decreasing time-to-market

- ❑ **Fixed shipping gates to occupy reserved shelf space**

➔ **Faults in delivered products are a fact-of-life**

Business impact

- Not satisfying the high reliability expectations
  - Many returned products
  - Damages brand image
  - Reduces market share

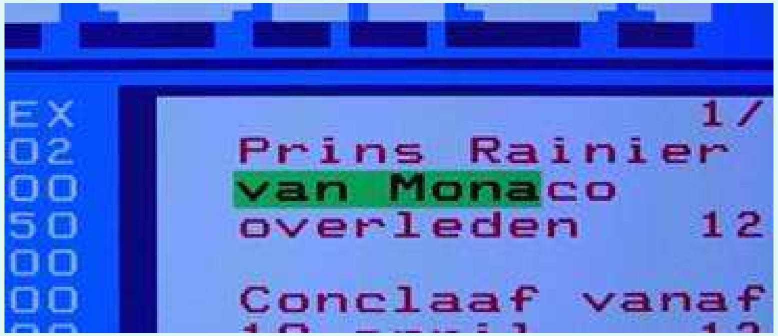
- Cost of non-quality (CoNQ)
  - 2-3% turnover (compare to research expenditures: 2.3%)
    Rudy Provoost (CEO Philips Consumer Electronics)

Challenge:

- Prevent product faults causing customer complaints constraints:
  - Low costs per item
  - Short time to market

Example

- Took TV that was on market for half a year
  - Installed latest upgrades
- Found 20 failures in 20 hours playing with TV
- 8 public upgrades in few months



EX
02
00
50
00
00

1/
Prins Rainier
van Monaco
overleden 12

Conclaaf vanaf

User sees problem immediately,

but TV seems unaware of it

Trader – Proposed Outcome

❑ Methods and techniques that

- **can expose,** *at development time*, **product** *weaknesses* **that could lead to erroneous behavior**
- **give the system** *awareness* **that its** *customer-perceived behaviour* **is or is likely to become erroneous**
- **provide the system with a** *strategy* **to** *correct* **itself in line with customer expectations**

❑ **Supported by**

- *Proof of concepts* **&** *publications* **that show the "how"**
- *Knowledge transfer* **to CIP and industry**

Trader – Proposed Outcome

❑ Methods and techniques that

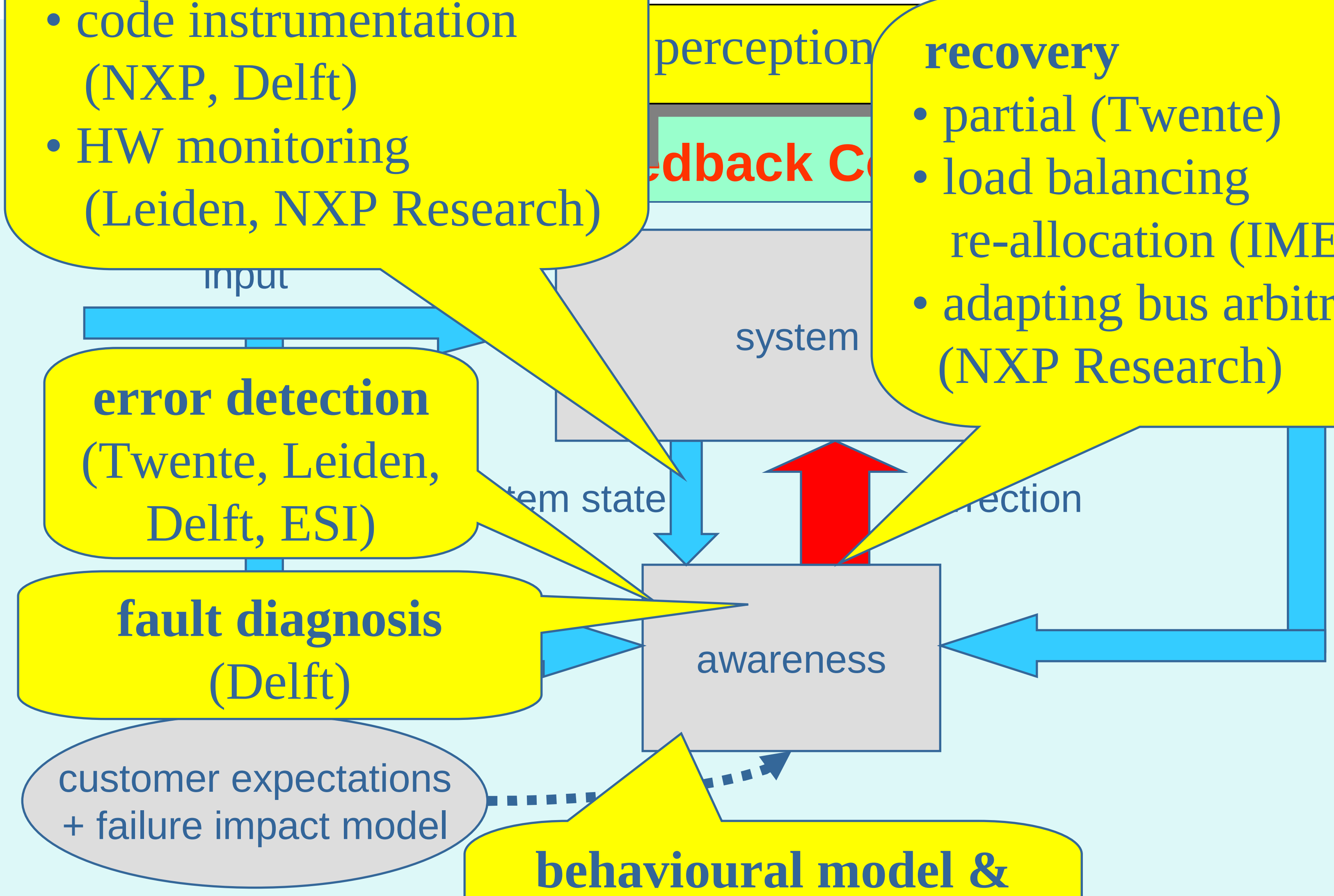
- **can expose,** *at development time*, **product** *weaknesses* **that could lead to erroneous behavior**
- **give the system** *awareness* **that its** *customer-perceived behavior* **is or is likely to become erroneous**
- **provide the system with a** *strategy* **to** *correct* **itself in line with customer expectations**

❑ **Supported by**

- *Proof of concepts* **&** *publications* **that show the "how"**
- *Knowledge transfer* **to CIP and industry**

observation
- code instrumentation (NXP, Delft)
- HW monitoring (Leiden, NXP Research)

recovery
- partial (Twente)
- load balancing re-allocation (IMEC)
- adapting bus arbitration (NXP Research)

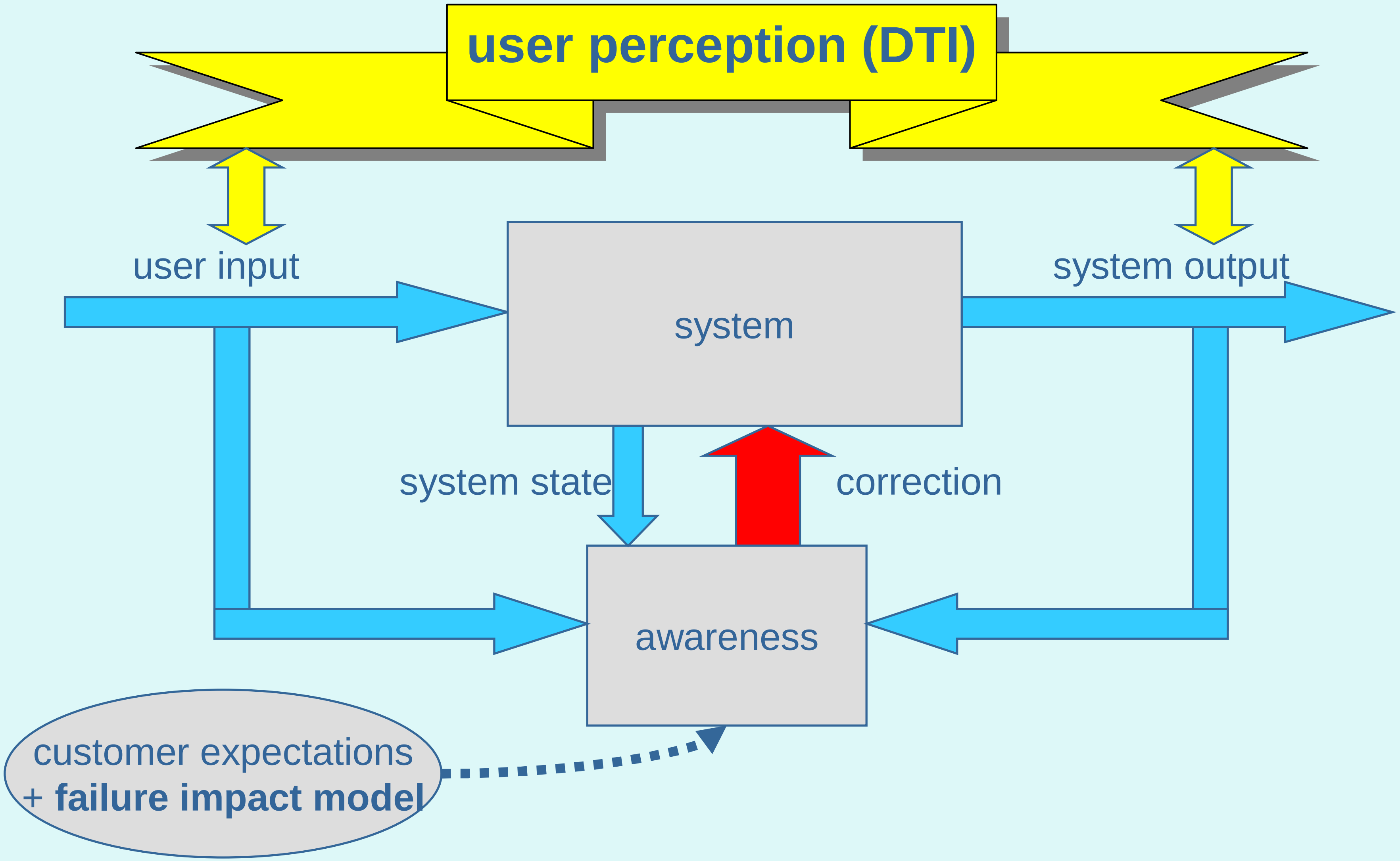


error detection
(Twente, Leiden, Delft, ESI)

fault diagnosis
(Delft)

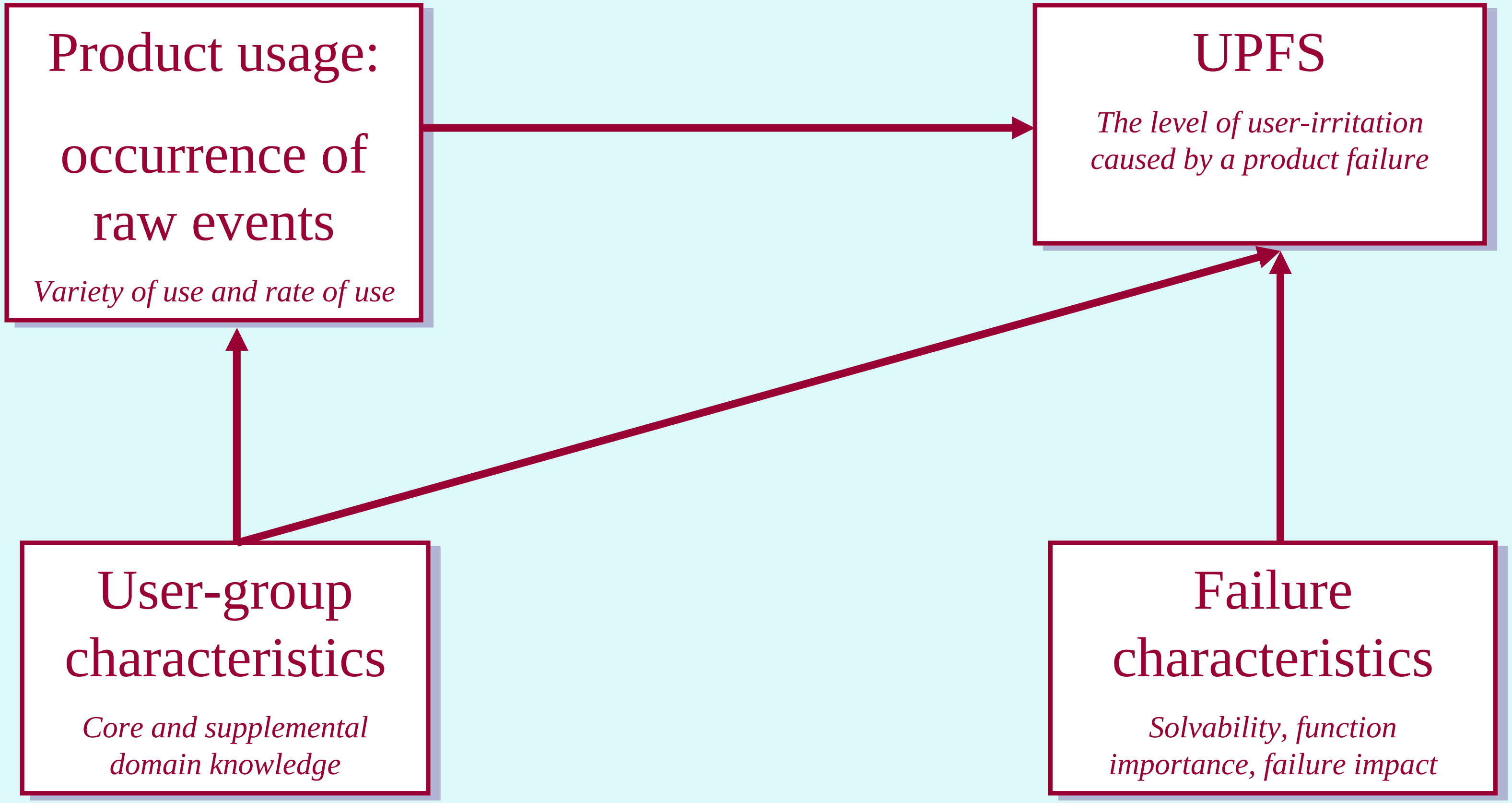
behavioural model & awareness framework
(ESI)

Awareness Inside



User preceived reliability

Aim is to capture User Perceived Failure Severity (UPFS)



Product usage:

occurrence of raw events

Variety of use and rate of use

UPFS

The level of user-irritation caused by a product failure

User-group characteristics

Core and supplemental domain knowledge

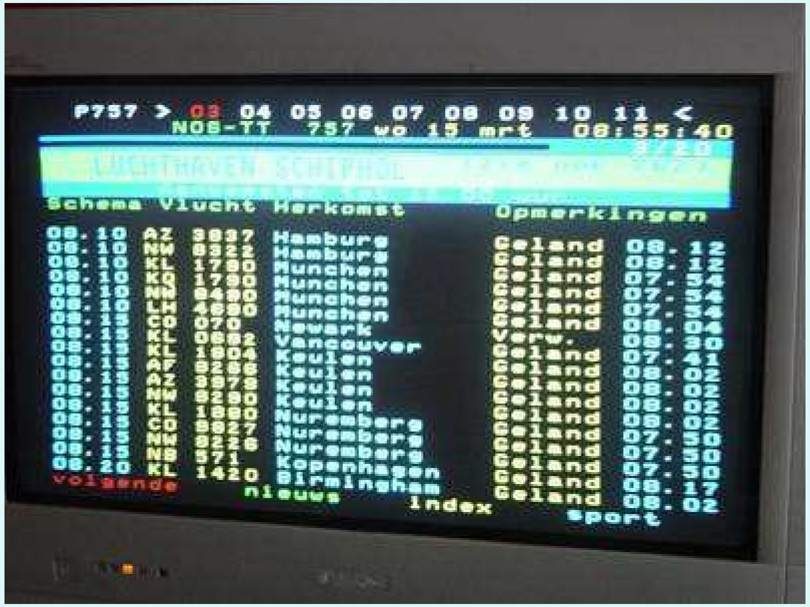
Failure characteristics

Solvability, function importance, failure impact

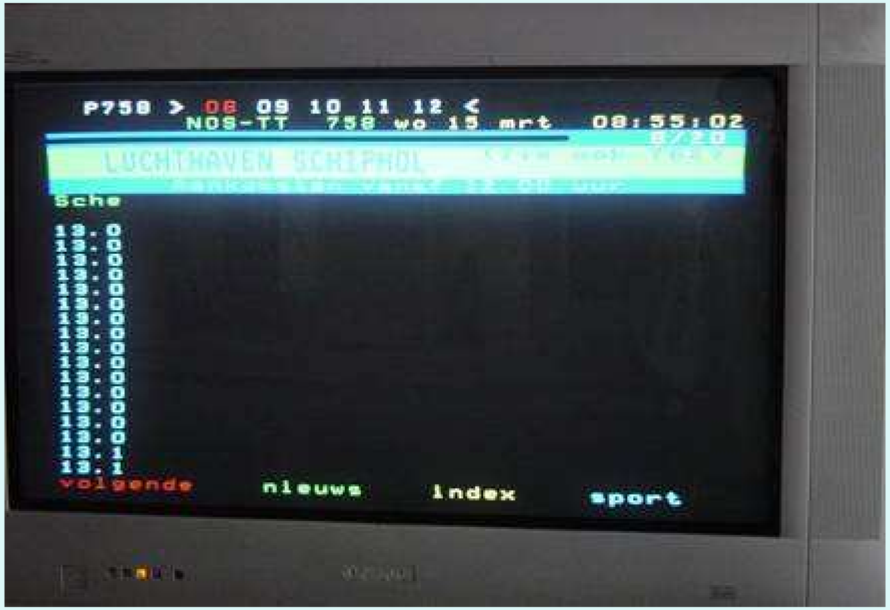
User perception

Example experiment

- Teletext experiment (29 subjects)
  - You have to pick up your wife/husband from Schiphol
  - Use Teletext to find arrival time for flight
- Failure was injected to hide part of this txt page

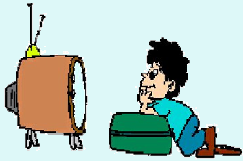


```
P757 > 03 04 05 06 07 08 09 10 11 <
         NOS-TT  757 wo 15 mrt   08:55:40
                                     9/20
   LUCHTHAVEN SCHIPHOL [illegible]
Schema Vlucht Herkomst          Opmerkingen
08.10  AZ 3837 Hamburg          Geland 08.12
08.10  NW 8322 Hamburg          Geland 08.12
08.10  KL 1790 Munchen          Geland 07.54
08.10  KQ 1790 Munchen          Geland 07.54
08.10  NW 8490 Munchen          Geland 07.54
08.10  LH 4890 Munchen          Geland 08.04
08.15  CO 070  Newark           Verw.  08.30
08.15  KL 0882 Vancouver        Geland 07.41
08.15  KL 1804 Keulen           Geland 08.02
08.15  AF 8288 Keulen           Geland 08.02
08.15  AZ 3978 Keulen           Geland 08.02
08.15  NW 8290 Keulen           Geland 08.02
08.15  KL 1880 Nuremberg        Geland 07.50
08.15  CO 8827 Nuremberg        Geland 07.50
08.15  NW 8226 Nuremberg        Geland 07.50
08.15  NB 571  Kopenhagen       Geland 08.17
08.20  KL 1420 Birmingham       Geland 08.02
volgende    nieuws    index    sport
```

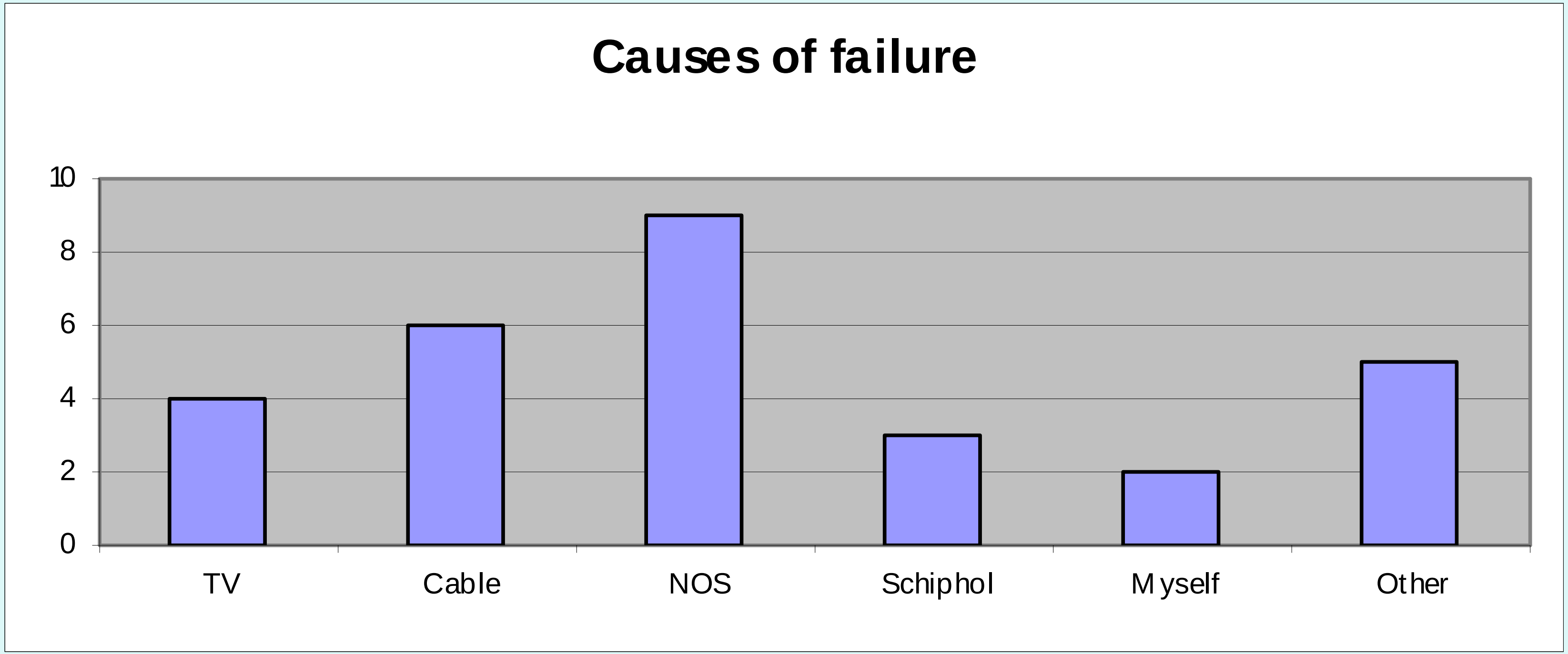


```
P758 > 08 09 10 11 12 <
         NOS-TT  758 wo 15 mrt   08:55:02
                                     8/20
   LUCHTHAVEN SCHIPHOL [illegible]
Sche
13.0
13.0
13.0
13.0
13.0
13.0
13.0
13.0
13.0
13.0
13.0
13.0
13.0
13.0
13.0
13.1
13.1
volgende    nieuws    index    sport
```

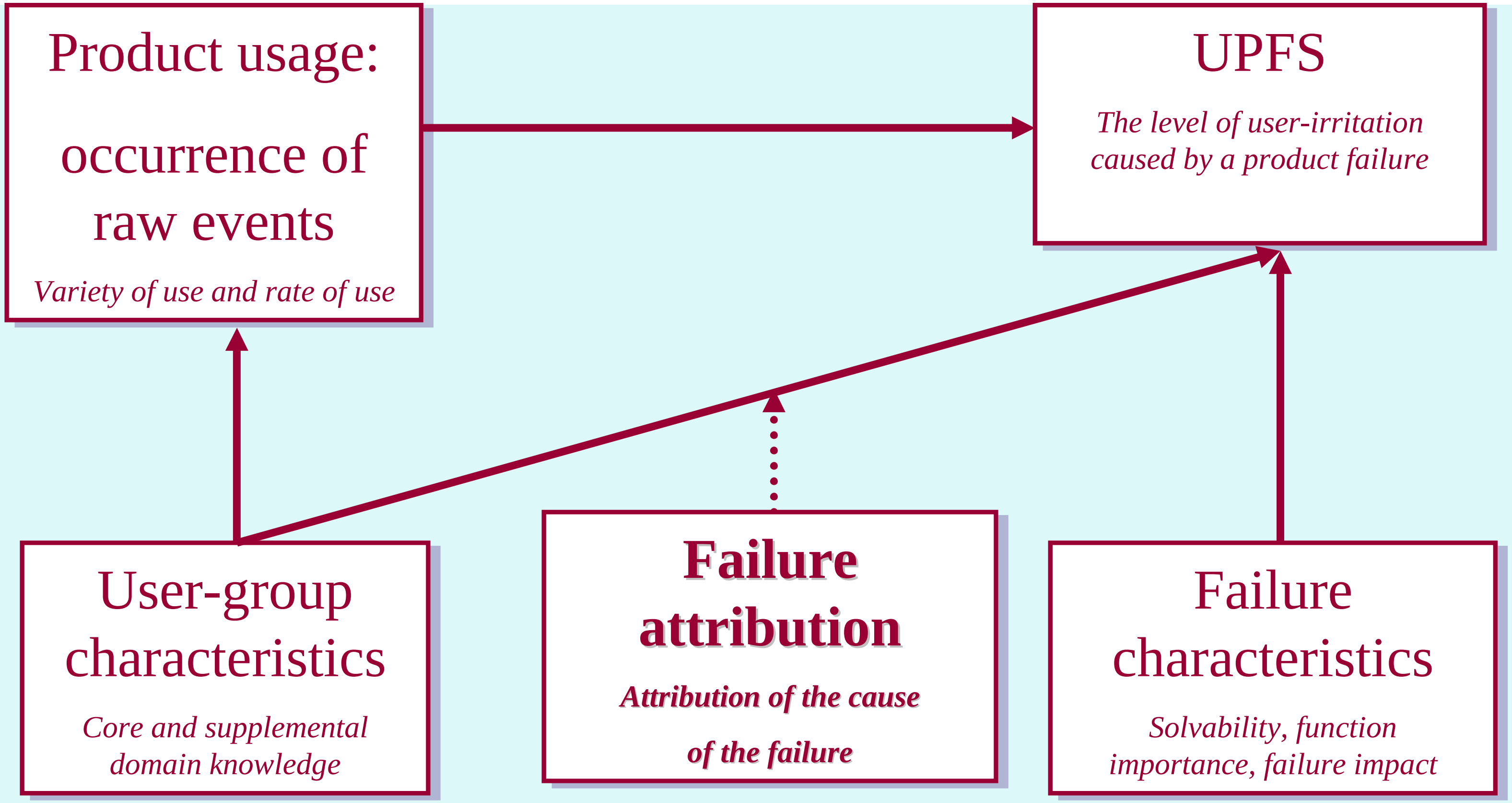
Conclusions from experiment



- ❑ Takes time/help to recognize there is a problem
- ❑ User did not recognize this problem as a TV problem



Adapted UPFS model



Product usage: occurrence of raw events

Variety of use and rate of use

UPFS

The level of user-irritation caused by a product failure

User-group characteristics

Core and supplemental domain knowledge

Failure attribution

Attribution of the cause of the failure

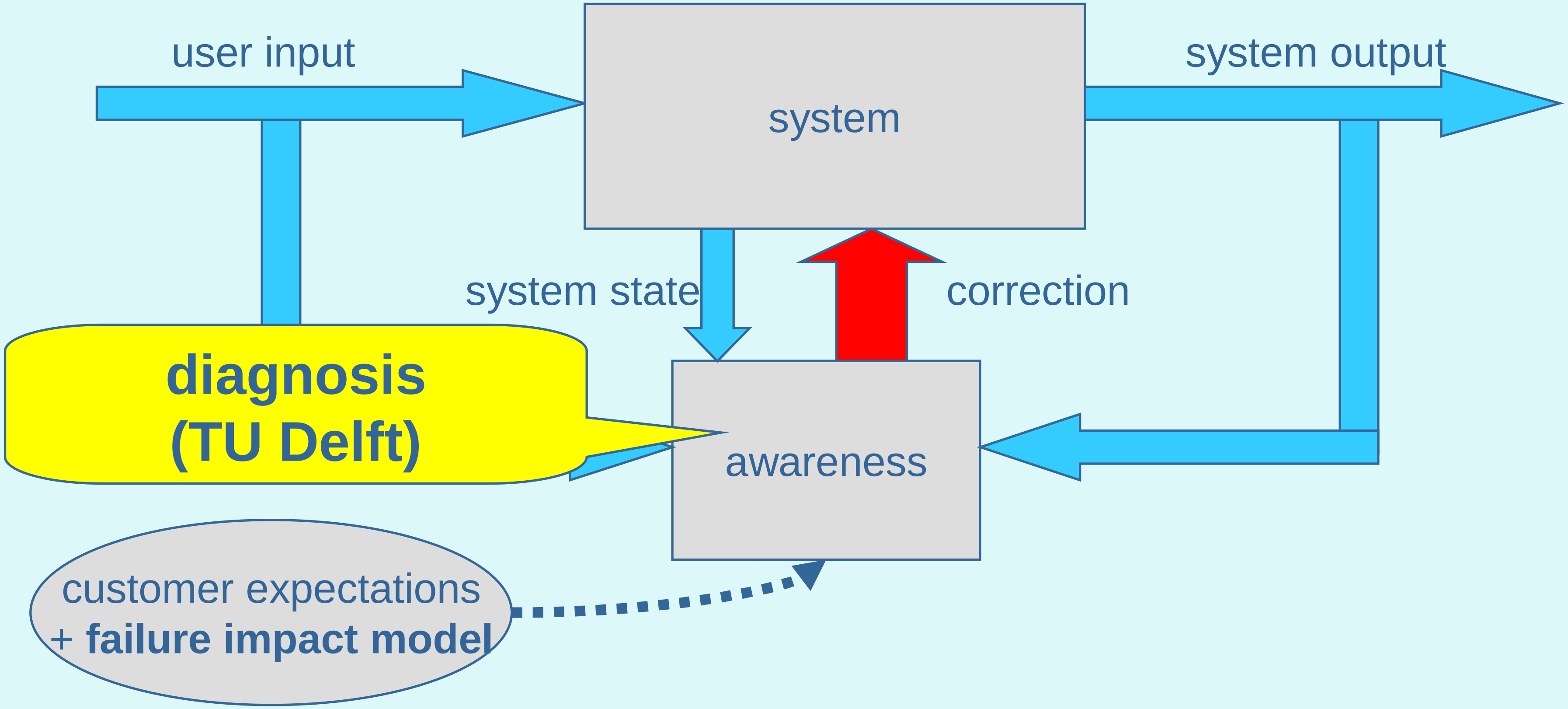
Failure characteristics

Solvability, function importance, failure impact

Recent experiment: compare failures of

- ❑ **motorized swivel (ranked as unimportant) and**
- ❑ **image quality (ranked as important)**

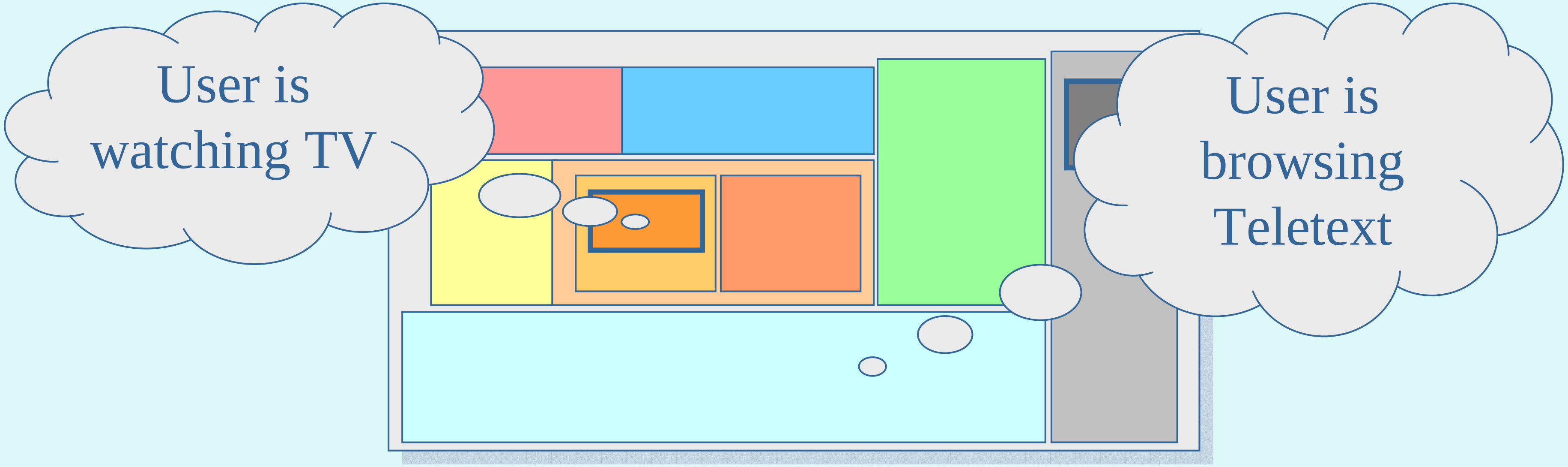
Awareness Inside



Case study: Teletext Lock-up

TV prepared with teletext faults

- ❑ Certain key sequences lead to failures (e.g. frozen or black screen)
- ❑ Each sub-system seems to work fine
- ❑ Synchronization is lost, but system is unaware (problem is not detected by current mechanisms)



diagnosis based on spectra

Case study: teletext lock-up

- ❑ Assume given: error detection [done by UTwente]

  Detection based on:
  - Explicit (high-level) behavioral information
  - Modeling the states of sub-systems
  - Check for consistency between states at run-time

- ❑ Aim of the diagnosis:
  - find block in C code that introduces the inconsistency

Fault diagnosis for teletext lock-up

1. **Add observations to C code to record which blocks are executed**

Exp: ≈ 60 000 blocks

Code instrumentation

```
Bool mgkey__rkeyntf_OnUp (KeySource source, KeySystem system, KeyCommand command)
{
  hook_log (20345);
  if ((1) && Enabled) {
    Bool translated=0;
    hook_log (20346);
    hook_EndTransaction ();
    ...
    if ( !translated) {
      hook_log (20349);
      Translate (source, system,  &command);
    }
    if (command >= 1000 && command <= 1009) {
      hook_log (20350);
      seq[0] = seq[1];
      seq[1] = seq[2];
      seq[2] = seq[3];
      seq[3] = command - 1000;
      if ( !triggered) {
        hook_log (20351);
        if (seq[0] == 1 && seq[1] == 2) {
          hook_log (20353);
          triggered = 1;
          switch (seq[3]) {
            case 1:
              hook_log (20354);
              tmode = 6;
              break;
            case 2:
              hook_log (20355);
              ...
```

start a new spectrum

- Transaction: time between two key presses
- Instrumentation using Front
- Small Koala component for caching / downloading spectra

log use of the block in the current spectrum

planted inconsistency

Fault in block 20354

Fault diagnosis for teletext lock-up

1. Add observations to C code to record which blocks are executed

   Exp: ≈ 60 000 blocks

2. For a sequence of key presses (scenario),

   collect for each block whether it has been executed or not between the presses; leads to vector (spectrum) for each block

   Exp: 2 scenarios with 24 and 27 key presses, where 13 451 and 13 796 blocks were executed

3. Record for each key press whether it leads to error or not

   Exp: 2 error vectors of length 24 and 27

Embedded Systems
INSTITUTE

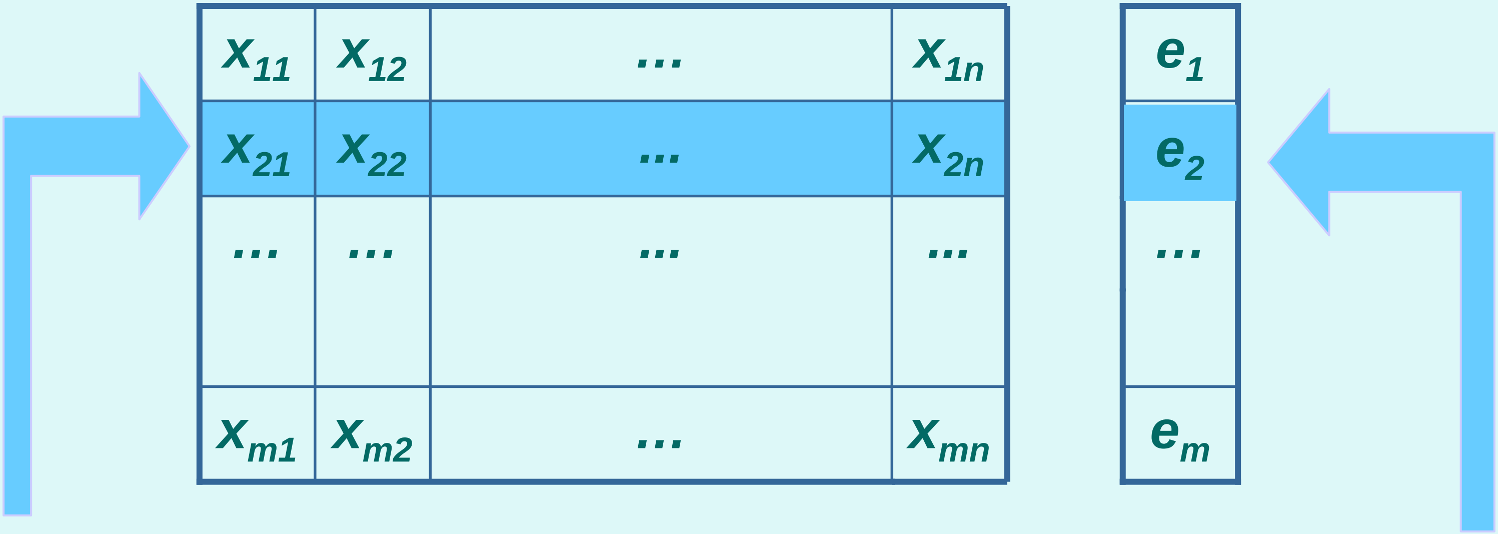
Fault Diagnosis

Spectra for *m* runs/transactions and n blocks

1: block executed
0: block not exec.

block 1

block n



block 1			block n
x_{11}	x_{12}	…	x_{1n}
x_{21}	x_{22}	…	x_{2n}
…	…	…	…
x_{m1}	x_{m2}	…	x_{mn}

e_1
e_2
…
e_m

Row: the blocks that are executed between 2 keys presses (transaction)

e_i=*1* : error in transaction i
e_i=*0* : no error in transaction i

Fault diagnosis for teletext lock-up

1. Add observations to C code to record which blocks are executed

   Exp: ≈ 60 000 blocks

2. For a sequence of key presses (scenario),

   collect for each block whether it has been executed or not between the presses; leads to vector (spectrum) for each block

   Exp: 2 scenarios with 24 and 27 key presses, where 13 451 and 13 796 blocks were executed

3. Record for each key press whether it leads to error or not

   Exp: 2 error vectors of length 24 and 27

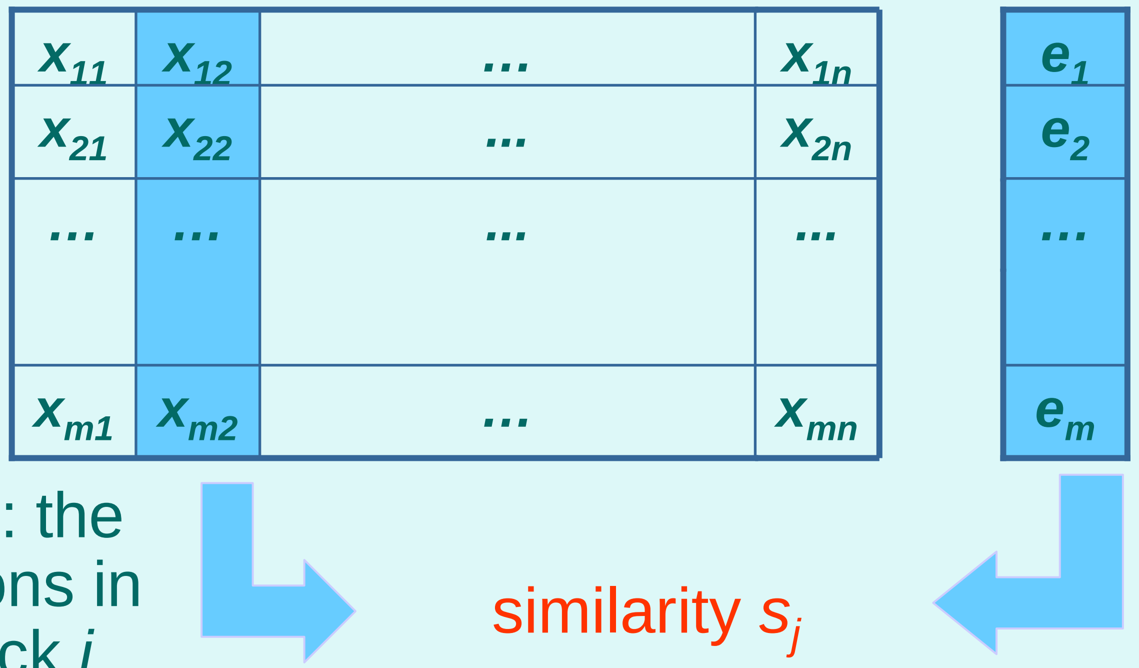
4. Compute simularity between error vector and spectra

Fault Diagnosis

Compare every column vector with the error vector.

block *j*

error vector



x_{11}	x_{12}	…	x_{1n}
x_{21}	x_{22}	…	x_{2n}
…	…	…	…
x_{m1}	x_{m2}	…	x_{mn}

e_1
e_2
…
e_m

Column *j* : the transactions in which block *j* was executed

similarity s_j

E.g. Jaccard similarity coefficient

Fault diagnosis for teletext lock-up

1. Add observations to C code to record which blocks are executed

   Exp: ≈ 60 000 blocks

2. For a sequence of key presses (scenario),

   collect for each block whether it has been executed or not between the presses; leads to vector (spectrum) for each block

   Exp: 2 scenarios with 24 and 27 key presses, where 13 451 and 13 796 blocks were executed

3. Record for each key press whether it leads to error or not

   Exp: 2 error vectors of length 24 and 27

4. Compute simularity between error vector and spectra

5. Rank blocks according to their simularity

Diagnosis

Scenario 1: Block ranking:

```
20353 (1/1)  20354 (1/1)  58890 (1/4)  3134 (1/5)  3664 (1/6)  3135 (1/6)
58889 (1/7)  59839 (1/8)  29569 (1/9)  1256 (1/9)15755 (1/10)  20351 (1/10)
15781 (1/11)  15777 (1/11)  15778 (1/11)  15779 (1/11)  15782 (1/11)
15823 (1/11)  20432 (1/11)  15727 (1/11) ...
```

Block 20354 is right at the top of the diagnosis

… but it shares the first position with block 20353

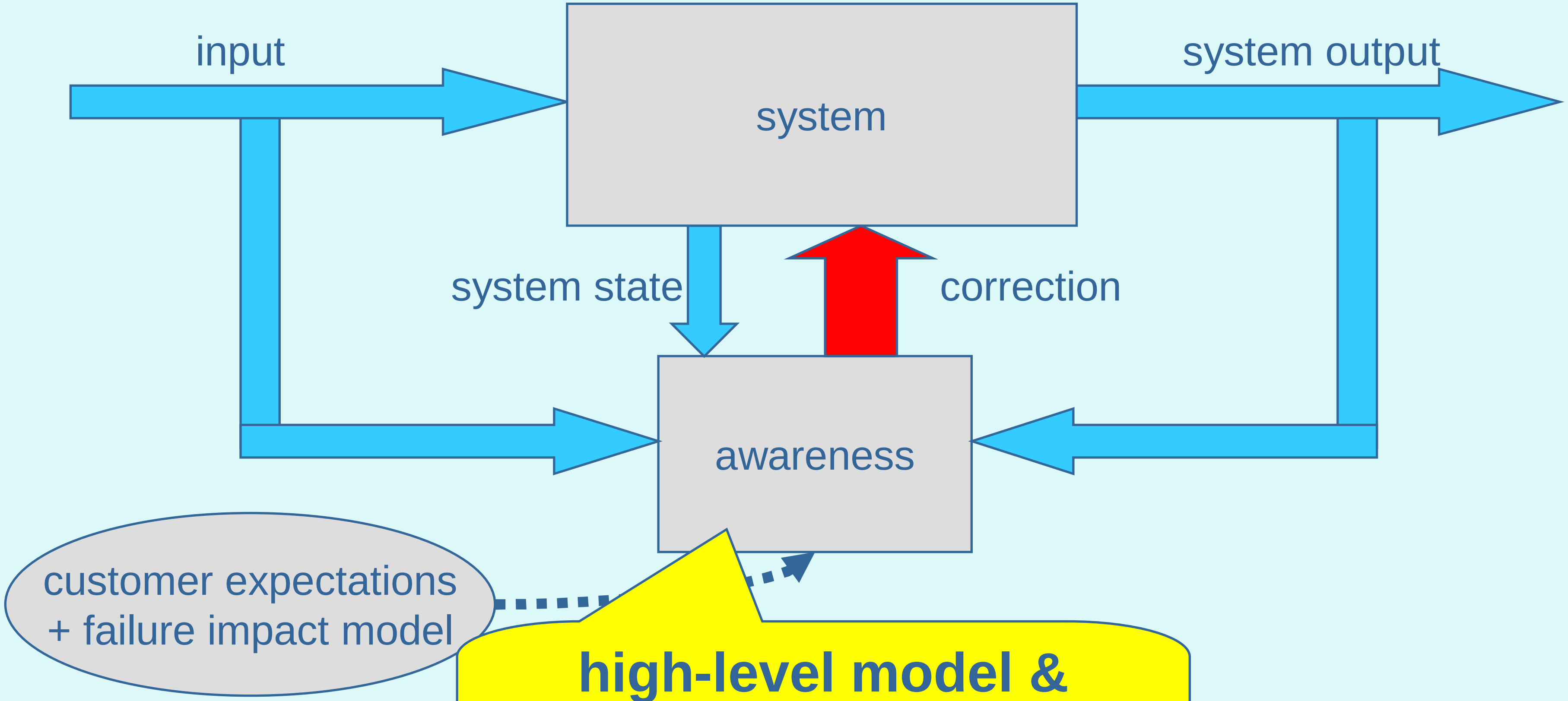
Scenario 2: Block ranking:

```
 20354 (1/1)  20353 (1/2)  3134 (1/5)  50466 (1/11)  20432 (1/11)
 15755 (1/11)  58208 (1/12)  58207 (1/12)  59816 (1/12)  50439 (1/12)
 50436 (1/12)  14817 (1/12)  50432 (1/12)  50437 (1/12)  50288 (1/12)
 50428 (1/12)  14814 (1/12)  14816 (1/12)  50422 (1/12)  …
```

Block 20354 is diagnosed correctly

Current this approach is tried at NXP to support debugging

Awareness Inside



high-level model & awareness framework (ESI)

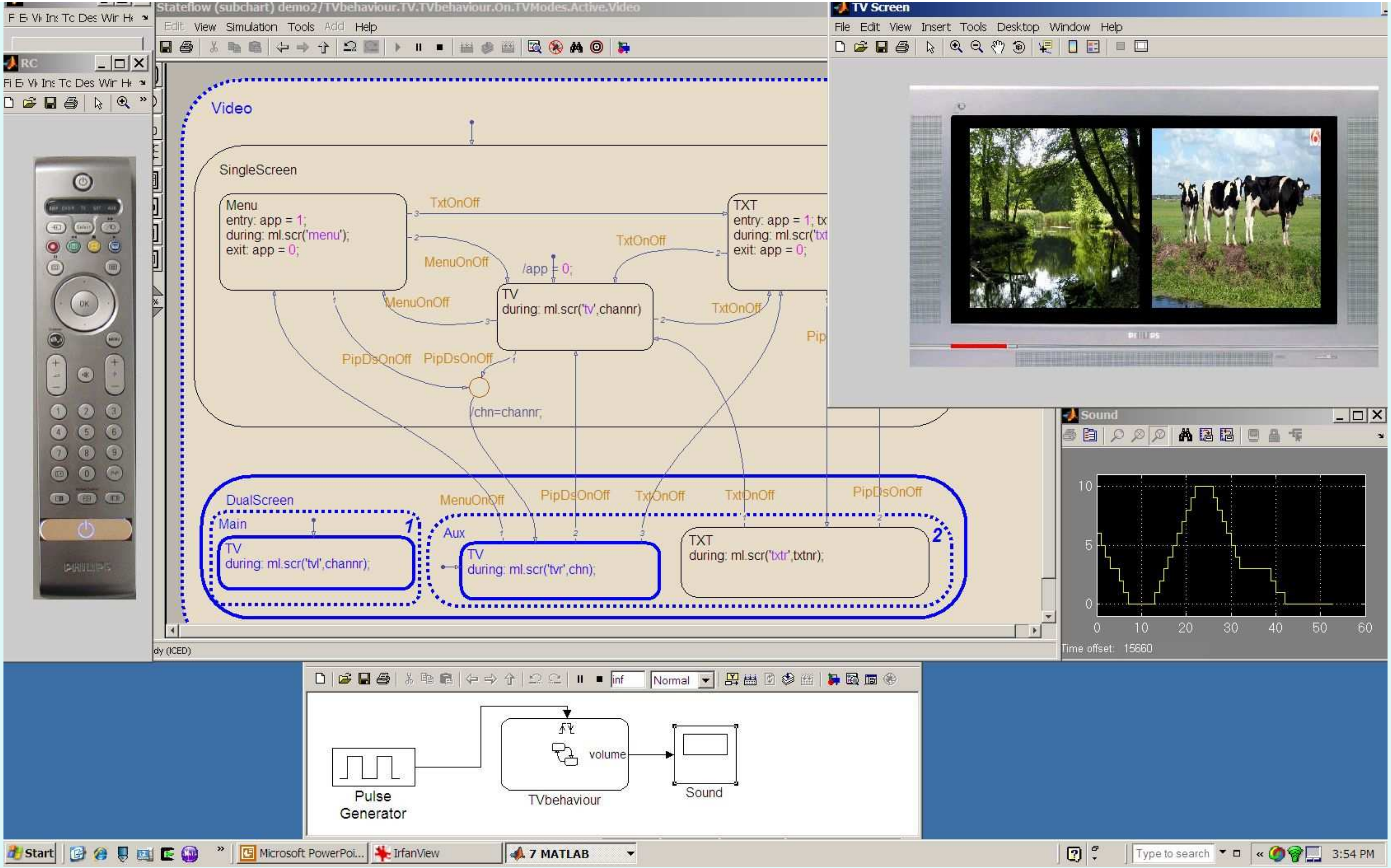
Model behaviour

Model user perceived behaviour mainly by executable state diagrams with hierarchy and concurrency to deal with complexity

Current tool support: Matlab/Simulink, mainly using Stateflow toolbox

Approach is rather tool-independent; diagrams similar to state machines in UML-tools

Modeling TV Behaviour in Stateflow



```
Menu
entry: app = 1;
during: ml.scr('menu');
exit: app = 0;

TXT
entry: app = 1; tx
during: ml.scr('txt
exit: app = 0;

TV
during: ml.scr('tv',channr)

TV
during: ml.scr('tvl',channr);

TV
during: ml.scr('tvr',chn);

TXT
during: ml.scr('txtr',txtnr);
```

Using the behavioural model (1)

improve@development:

- ❑ to obtain **concise, visual specification;**
  currently spec is distributed over many documents
- ❑ to enable **early detection of faults**
  (e.g. ambiguities, omissions, inconsistencies, interference between features)
- ❑ to get **quick feedback** on product variations
- ❑ to **generate test cases**, e.g., to test implementations

➔ Transfer to NXP in preparation

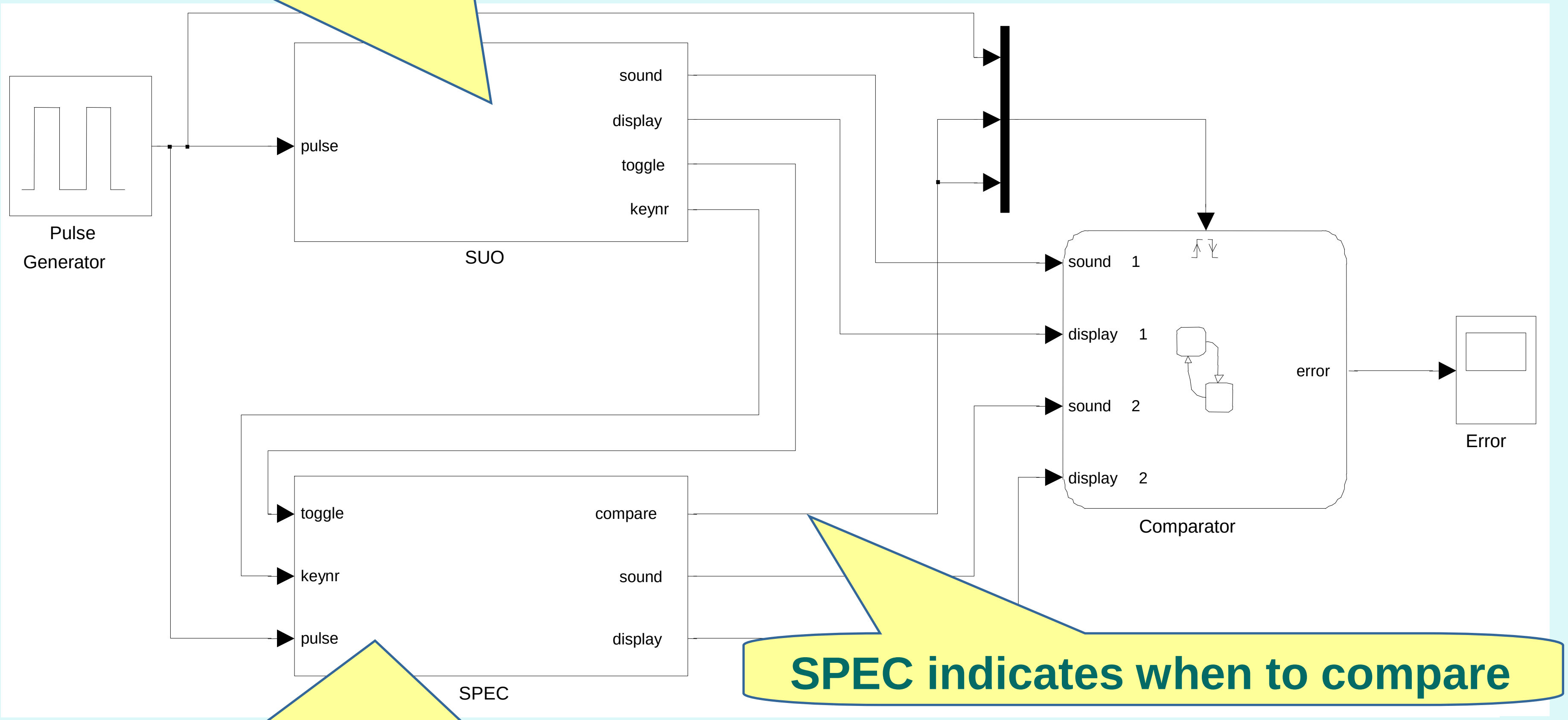
Using the behavioural model (2)

improve@run-time

Experiment with awareness concept:

- ❑ **Linux-based awareness framework in which System Under Observation (SUO) and SPEC can be inserted easily and we can try different error detection strategies**
- ❑ **Open source media player MPlayer as first case study, followed by experiments in TV domain**
- ❑ **Model awareness concepts in Stateflow**

Awareness in Stateflow

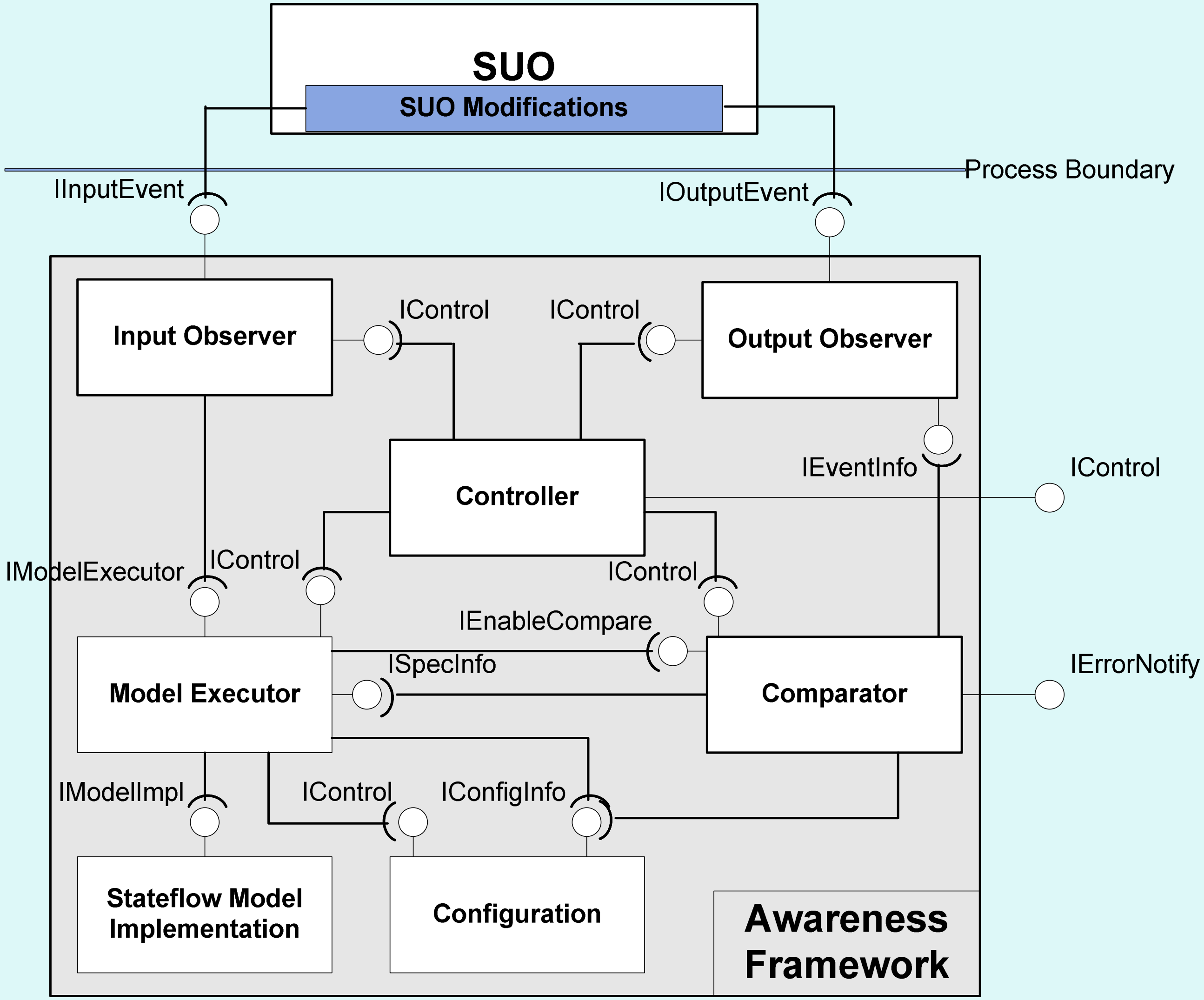


System Under Observation (SUO)

SPEC indicates when to compare

Behavioural specification model (SPEC)

Design of Framework in Linux



Concluding remarks

- System level feedback is a powerful generic concept for the design of dependable systems
  - paradigm shift:

    systems shall be correct ⇒ errors must be contained
- High-level system models become a component of the system itself
  - system features emerge from interaction basic system and model components
  - further experimentation needed to determine trade-off between model complexity and effectiveness
- Effective compositional instrumentation using aspect-oriented programming
- Industry-as-laboratory very useful research instrument for system engineering

Thank you for your attention!