FOMHCI 2015 notes

Interventions

Benjamin Weyers

Classical layered architecture

- Physical representation
- Interaction logic
- System interface
- System logic

Workflow for HCI

- Modeling phase
- Use phase
- Adaptation phase

FILL - formal interaction logic language

- Operation nodes
  - System operation
  - Interaction-logic operation
  - Channel operation
- BPMN nodes
- Edges

Maps to reference nets (Petri net with annotations for types)

Double pushout graph rewriting on petri nets

Judy Bowen

Combining models at different levels of abstraction Different stakeholders, designers, programmers...

Do with what we have and combine those

The specification contains different artifacts

- General behaviours
  - Z specification language

- Simulation interface
- Defined startup procedure
- Defined shutdown procedure

4 languages

Z, microcharts, PIM, PRM

Guillaume Maudoux

Concept of *safe mental model*

HMI LTS : Labelled Transition System with in and outs

Bad situations

- A command missing on the system model
- A command the user is not aware of
- A variable is not observable for the user

Control property vs *Full control* property

Extract elementary tasks to teach to the user, from the LTS of the full system

Johannes Pfeffer

Use a subset of BPMN to orchestrate apps workflows

RDF to store all data on chemical plants

Orchestration engine running on servers and on client devices, and interacts with apps

Philippe Palanque

User centered design

Norman's interaction theory

Camille Fayolas

Anik E-1 & E-2 1994-01-20 failure with a cost of 60M$

José Campos

- Theorem proving with PVS Based on higher order logic
- Model checking with IVY and SMV, models as finite state machine

Layered specification

- First layer constant and types related to devices and entities in the system
- Second layer the underlying process ofen reused on families of devices
- Third layer interfaces for devices
- Fourth layer information resources

Represent the user in the formal language

How to proove that both representation (PVS and SMV) are simlar ? They have the same concepts, so the translation is mostly syntactic

Bart Meyers

Meta model used to model DSLs

Use spin and promela Properties in LTL are complicated

4 different DSL: Properties, system, configuration, state

DSLs are generated from the annotated meta model using UML stereotypes

Everything is represented internally as a graph

AtomPM a tool for multi paradigm modeling

Genetic algorithms for test case generation

Discussions

In the past

2 books on formal methods in HCI

- Harrisson 1991 Patchwork
- Patterno 1997 After CHI 1996

The book

Deadline : 2015-12 Size : 400 Pages Case study or not ? Yes, a chapter to generally describe each case, and then chapters can refer to them.

Structured along topics

- Introduction
- Analysis and Verification and Valdiation
- Modeling
- Execution
- Future work

Put in presentation

LIDL

```
interface Reactor is
{
  command:{
    sv1: Boolean,
    sv2: Boolean,
    wv1: Boolean,
    wv2: Boolean,
    wp1: Number,
    wp2: Number,
    cp: Nubmer,
    rodPosition: Number
  } in,
  status:{
    sv1: Boolean,
    sv2: Boolean,
    wv1: Boolean,
    wv2: Boolean,
    cpUmin: Number,
    wp1Umin:Number,
    wp2Umin:Number,
    rodPosition:Number,
    outputPower:Number,
    reactorWaterLevel:Number,
    reactorPressure:Number,
    condenserWaterLevel:Number,
    condenserPressure:Number
  } out
}
```

```
interface ReactorUser is
{
  powerDisplay: Label,
  reactor: DualGaugeWidget,
  condenser: DualGaugeWidget,
  controlRods: ComplexSlider,
  wp1: ComplexSlider,
  wp2: ComplexSlider,
  cp: ComplexSlider,
  sv1: ValveWidget,
  sv2: ValveWidget,
  wv1: ValveWidget,
  wv1: ValveWidget,
  leds: MultipleLedWidget
}

interaction
  (human machine interface connecting (user:ReactorUser) to (reactor:Reactor)):Activation in
is
  ((user)=({
    powerDisplay:(label (active) displaying (reactor.status.outputPower))
    reactor: (dual gauge (active) with water level (reactor.status.reactorWaterLevel))

    }))

interface Task is
  {
    start: Activation in,
    abort: Activation in,
    progress: Number out,
    running: Activation out,
    finished: Activation out
  }

interaction
  (reactor user manual (reactor:co(ReactorUser))):Task
is
  ( sequentially
    (start up (reactor))
    (operate (reactor))
    (shut down (reactor))
  )

interaction
```

```
  (start up (reactor:co(ReactorUser))):Task
is
  ( sequentially
    ( all
      (open valve (reactor.sv1))
      (open valve (reactor.wv1))
    )
    ( all
      ( set slider (reactor.wp1) to (1000))
      ( set slider (reactor.cp) to (1000))
    )
    ( set slider (reactor.controlRods) to (90))
  )
```