# Chapter 1

# Concurrent Task Trees

## Recap

---

**Slide Context Toolkit:**

- Context Toolkit
  - Context Abstraction
  - Design Methodology

---

## 1.1 Task Models

---

**Slide HCI Lecture Summary:**

- Theories
  - Levels-of-analysis
  - Stages-of-action
  - GOMS
  - Widget-level
  - Context-of-use
  - Object Action Interface models

**Slide Describing user interaction:**

- Remember GOMS - Goals, Operators, Methods, Selection Rues
- The user wants to reach a Goal, he uses Operators and Methods that he selects via Selection Rules
- With GOMS, we can look at a sequence of Methods and analyze it.
- We can analyze a system using GOMS, but a GOMS model does not tell us how to implement a system
- Question: How can a GOMS-like system support development?
- A *Task Model* can be used to guide the implementation.

**Slide Task Model:**

- Task models indicate the logical activities that an application should support to reach users' goals.(Paterno, 1999)
- Goals are either state changes or inquiries
- Tasks can be highly abstract or very concrete
- Task models can be build for existing systems, future systems and for the user's view of the system
- Task models are formalized, other methods are often informal

**Slide What's the use of a Task Model?:**

- Understand the application domain
- Record the result of user discussions
- Support effective design
- Support usability evaluation
- Directly support the user in using the system
- Documentation

**Slide Task Model Representation:**

- GOMS can represent a task model
- GOMS is mainly textual
- GOMS cannot represent concurrency, interruption, order independence, optionality and iteration.
- Alternative: ConcurTaskTrees (Paterno, 1999)

## 1.2 ConcurTaskTrees

**Slide ConcurTaskTrees:**

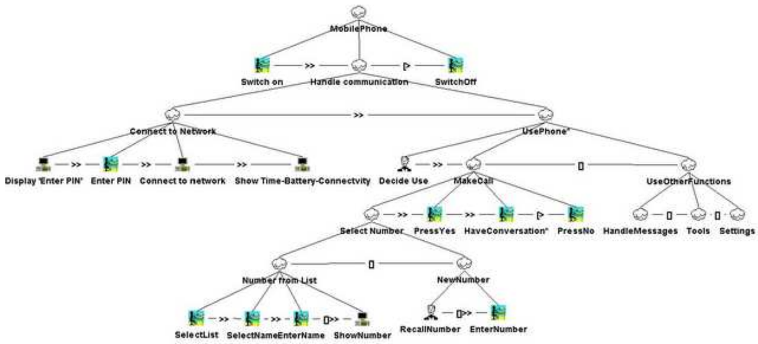



Image from Paterno, 1999

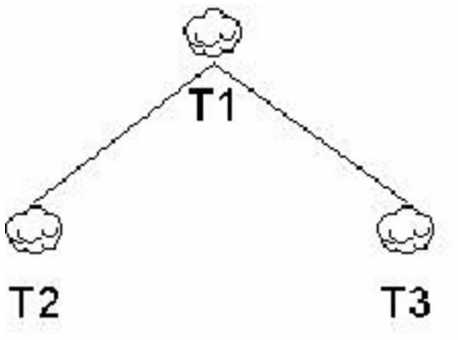
**Slide CTT: Features:**

- Hierarchical structure
- Graphical Syntax
- Many temporal operators
- Focus on activities

### 1.2.1 Temporal Operators

---

**Slide CTT: Temporal Operators:**

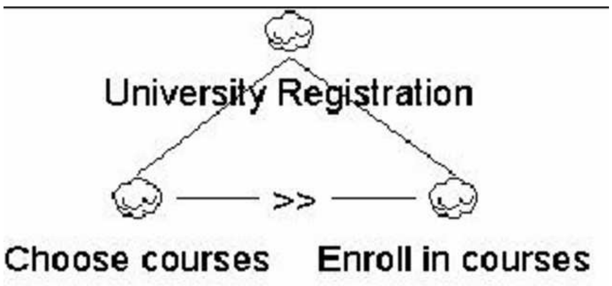



- Hierarchy

Image from Paterno, 1999

---

**Slide CTT: Temporal Operators:**

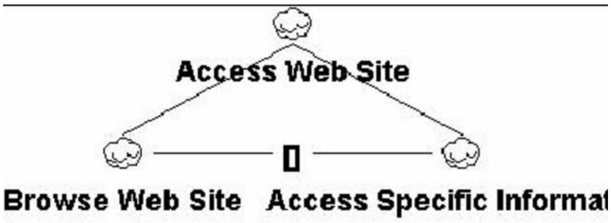



- Enabling

Image from Paterno, 1999

---

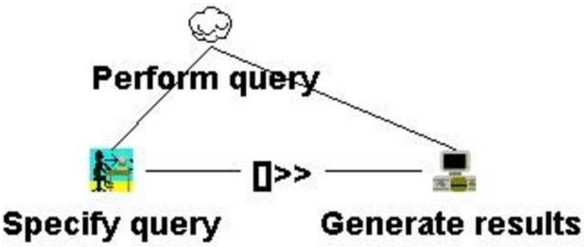
**Slide CTT: Temporal Operators:**




- Choice

Image from Paterno, 1999

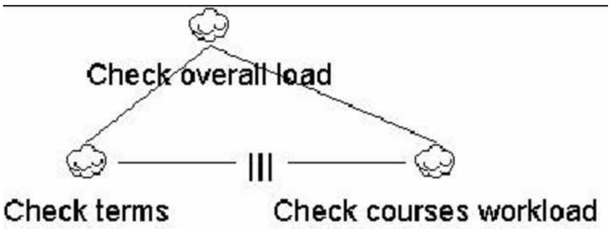
**Slide CTT: Temporal Operators:**




- Enabling with information passing

Image from Paterno, 1999

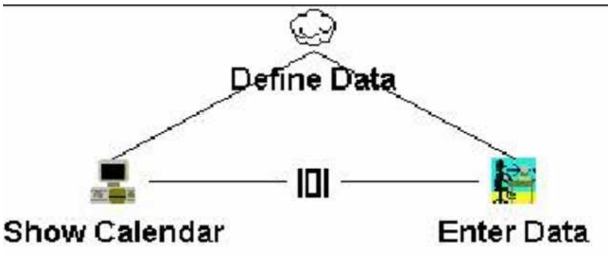
**Slide CTT: Temporal Operators:**




- Concurrent Tasks

Image from Paterno, 1999

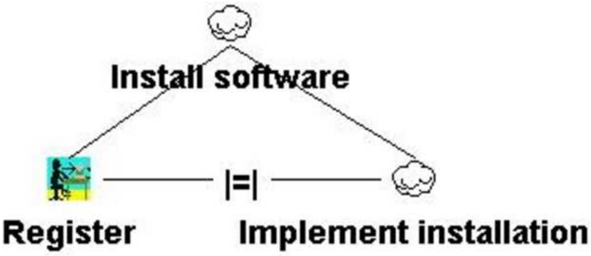
**Slide CTT: Temporal Operators:**




- Concurrent Communicating Tasks

Image from Paterno, 1999

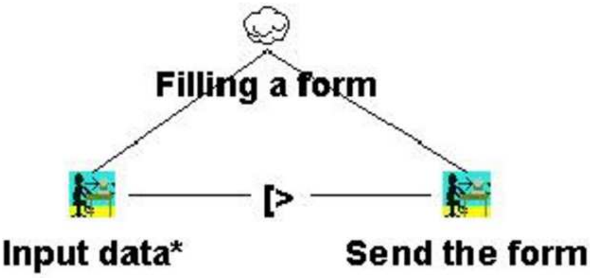
**Slide CTT: Temporal Operators:**




- Task Independence

Image from Paterno, 1999

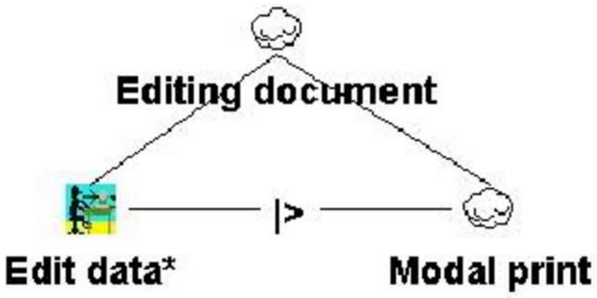
**Slide CTT: Temporal Operators:**




- Disabling

Image from Paterno, 1999

**Slide CTT: Temporal Operators:**

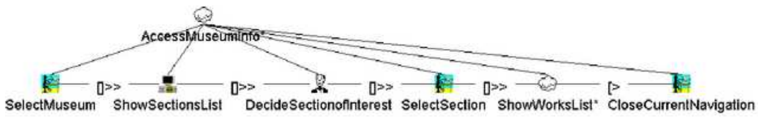



- Suspend-Resume

Image from Paterno, 1999

### 1.2.2 Examples

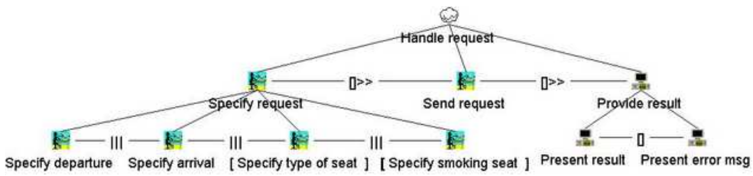
**Slide CTT: iterative task:**




- Task sequence with iteration: only the last transition ends the iteration

Image from Paterno, 1999

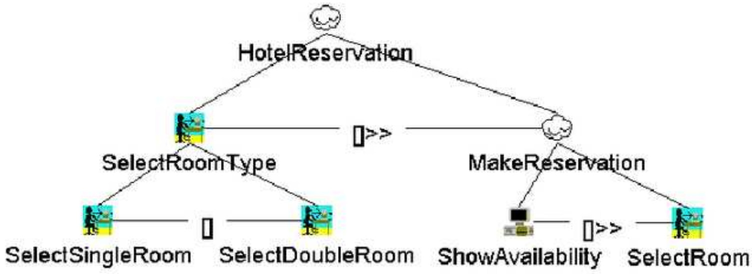
**Slide CTT: optional tasks:**




- Optional Tasks are marked with [ and ] brackets

Image from Paterno, 1999

## Slide CTT: inheritance of temporal constraint:




- ShowAvailability inherits the temporal constraint (executed after SelectRoomType) from its parent MakeReservation

Image from Paterno, 1999

# Chapter 2

# Abstract and Wearable UIs

---

**Slide Wearable UIs:**

- Supporting a primary task, i.e. UI driven by external task
- Context-dependent (primary task is one context source)
- Non-”point-and-click”, i.e. No WIMP-based UI
- Sometimes no graphical UI at all
- Rich set of in- and output devices
- Question: How to write (and reuse) code for “generic” wearable computer?

---

## 2.1 Abstract UIs

---

**Slide Characterizing Wearable UIs:**

- Displaying information and changing state (like CTTs)
- Additionally: Context information
  - Context-dependent presentation
  - context includes input and output modes and devices available
  - Context change triggers information display / state change
- Idea:

- specify abstract UI using CTTs
- use context change triggers like input in CTTs
- decide context-dependent presentation during runtime

---

**Slide Context-dependent presentation:**

- Example: a web browser with two presentation modes
  - Desktop mode: Like firefox
  - Mobile mode: like opera “small screen rendering”
- Specification of UI (= html document, links) the same
- “Rendering” of UI different:
  - Compress graphics, change positions, use different fonts
  - Change interaction: no mouse click, but chose links via cursor keys

---

**Slide Abstract Specification:**

- Simple Example: Write Aircraft Repair Report
  - Input text of repair report
  - Indicate that the repair report entered is complete
- i.e. use CTT to specify abstract model
- Web browser equivalent: Form
  - Text input field
  - “submit” button

---

[fragile] **Slide AWT implementation:**

- PDA: Java 1.2 (AWT)

```
Panel p = new Panel();
p.add(new Label ("Enter␣Report");
TextField tf = new TextField("Your␣Report␣Here",256);
p.add(tf);
Button b = new Button("Save");
p.add(b);
```

---

```
private void makeTextInput( Container c, TextInputItem i, int depth ) {
     Panel p = new Panel();
     p.setLayout( new FlowLayout( FlowLayout.LEFT ) );
     if( depth == 0 ) {
         c.add( p );
     } else {
         c.add( p, BorderLayout.NORTH );
     }

     p.add( new Label( i.getDescription().getText() ) );

     TextField tf = new TextField( i.getInput(), i.getExpectedLength() );
     TextInputListener l = new TextInputListener( this, i, tf );
     tf.addTextListener( l );
     mActions.add( l );
     p.add( tf );
 }
```

---

[fragile] **Slide Swing implementation:**

- Desktop: Java 5 (Swing)

```
JPanel p = new JPanel();
p.add(new JLabel ("Enter␣Report");
JTextField tf = new JTextField("Your␣Report␣Here",256);
p.add(tf);
JButton b = new JButton("Save");
p.add(b);
```

---

---

[fragile] **Slide QT implementation:**

- QT 4

```
QLabel *reportLabel = new QLabel(tr("Enter␣report"));
QTextEdit *reportEdit = new QTextEdit;
QPushButton *saveButton = new QPushButton(tr("Save"));
myLayout = new QHBoxLayout;
myLayout->addWidget(reportLabel);
```

```
myLayout->addWidget(reportEdit);
myLayout->addWidget(saveButton);
```

**Slide Abstract to concrete:**

- How to get from abstract to concrete?
- Idea 1: Use an expert programmer, give him the spec, let him program, use result
- How about different devices?
- Idea 1a: Use expert for every possible device, send to expert programmer, let them work together.
- How about different contexts?
- Idea 1b: Use domain expert to describe contexts, send to device expert to design context-dependent optimal display for specific device, send to programmer, program
- Only viable for small number of devices and huge sales. i.e. mobile phone games

**Slide Abstract to concrete (2):**

- Can we do without all these experts?
- Idea 2: Divide the application program in two parts: The abstract UI and the renderer
- How about different devices?
- The renderer can be device-specific: It knows best how to use UI elements of the target device
- How about different contexts?
- The renderer itself can use context information in a device-specific way
- The abstract UI can choose from a number of available renderers. This choice can be based on device availability, user preference, context.

[fragile] **Slide AbstractUI implementation:**

- AbstractUI

```
mSave = new TriggerItem2(
      new TextData( "Save" ), false, this );
mComment = new TextInputItem2(
      new TextData( "Comment" ),
      20, "Your␣text␣here",this );
mComment.setNext( mSave );
mRoot = new GroupItem2(
      new TextData( "Write␣Repair␣Report" ),
      this );
mRoot.setSub( mComment );
```

---

**Slide Open questions:**

- Fundamental question: What can the AbstractUI express?
  - Speech-driven UI?
  - How to deal with non-renderable objects? (picture on audio-UI)
- Technical question: How can we implement it?
  - How can we specify an AbstractUI Model? XML?
  - How can the renderer decide what subtree of the CTT it renders? on-demand query mechanism?

---

## 2.2 Wearable UIs

**Slide Wearable UI Methaphor:**

- Output Mechanism
  - Visual: HMD
  - Audio
- Input Mechanism
  - Keys: Keyboard, Twiddler
  - Hands: gestures, direct manipulation
  - Speech
- Interaction Methods

– menu selection, direct manipulation, form fillin
– command language, natural Speech

---

**Slide Winspect GUI:**

- Java Implementation
- Uses HMD and “hands-free interaction”
- GUI elements optimized for wearable use
  - Colors, font sizes, highlighting
- Interaction based on dataglove
  - Direct Manipulation: Motion, Turn
  - Gesture for selection

---

**Slide Winspect UI HMD:**


Image from T. Nicolai

---

**Slide Winspect Direct Manipulation:**

Image from T. Nicolai

**Slide WearableUI:**

- Renderer for AbstractUI
- Uses HMD and “hands-free interaction”
- GUI elements optimized for wearable use
  - Colors, font sizes, highlighting
  - Few elements displayed
  - shows in the area of visual focus
- Interaction based on dataglove
  - Hand gestures to navigate and select
  - Additional keyboard for text entry

**Slide Wearable UI Gesture:**

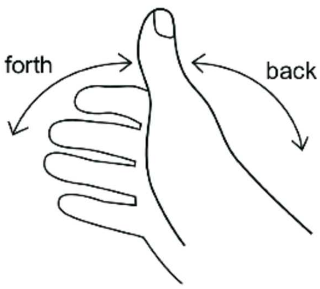


Image from H. Witt

**Slide Wearable UI Glove:**

Image from H. Witt

**Slide Wearable UI HMD:**

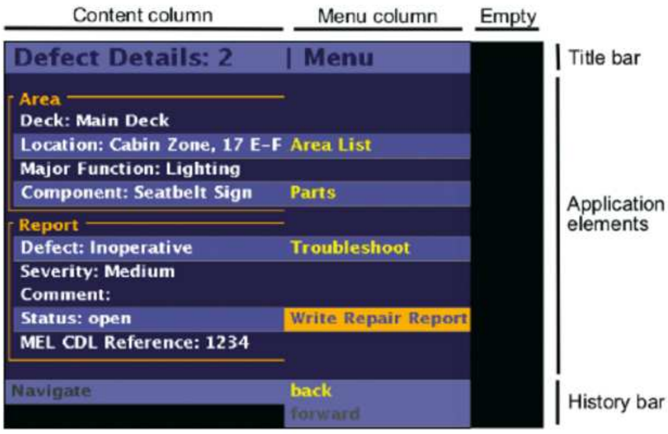



Image from H. Witt

# Chapter 3

# Wearable Evaluation

## Recap

---

**Slide Abstract/Wearable UI:**

- AbstractUI
  - Device-independent
  - Context-aware
- WearableUI
  - Uses AbstractUI
  - Wearable interaction mode

---

## 3.1 Adaptive UIs

---

**Slide WUI-Development:**

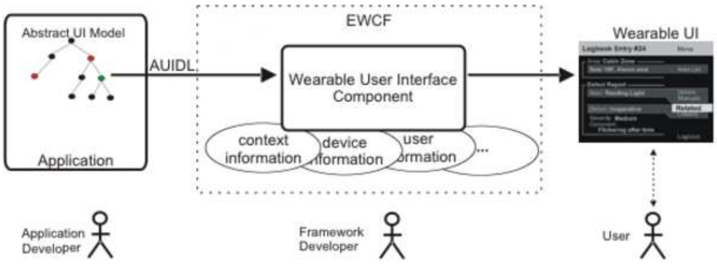



Image from H. Witt

---

**Slide WUI-Structure:**

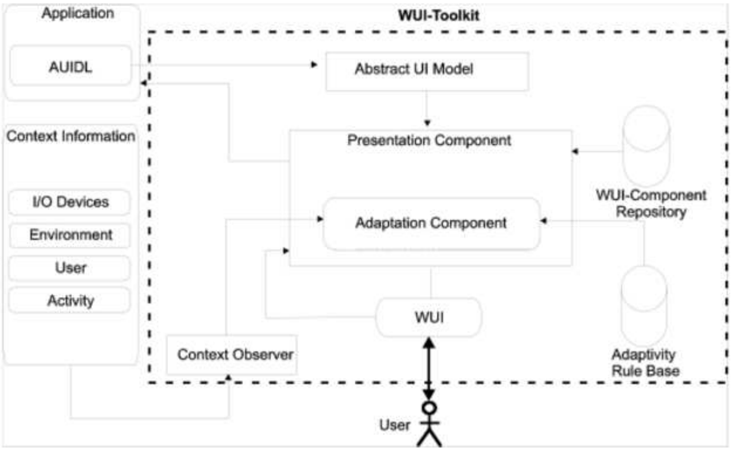



Image from H. Witt

---

**Slide Adaptive UIs:**

- Why adapt an UI?
- UI can be optimized due to changes in environmental context
  - Light conditions
  - User motion
  - Environmental noise
- UI cannot be controlled anymore under current context
  - affected by user activities
  - interaction device failure (e.g. low battery)

**Slide Layers of adaptation:**

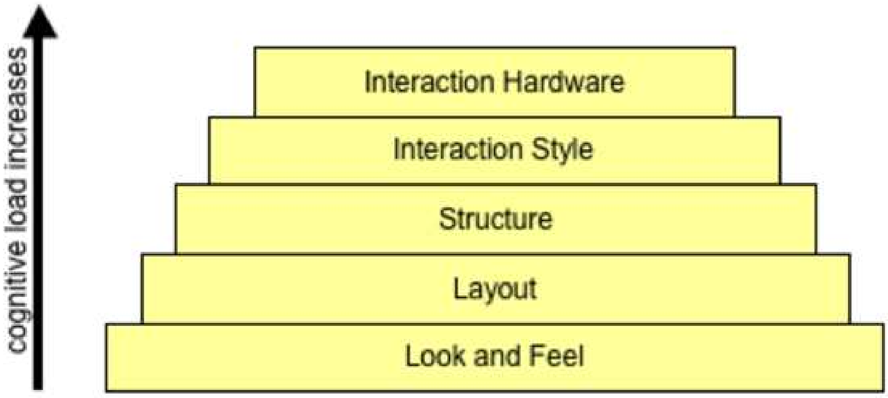



Image from H. Witt

**Slide Finding adaption rules:**

- How to find rules for adaptation?
- What's the user reaction on adaptation?

## 3.2 Wearable Evaluation

**Slide Wearable Evaluation:**

- How to measure the performance of a wearable system?
- Remember: Supporting a primary task
- Idea: measure the performance in the primary task.
- Example: Wearable Maintenance support

- Time
- Quality

---

---

**Slide Wearable Evaluation (2):**

- Drawbacks:
  - Long time needed
  - Variation in users/Tasks: Even more time needed
  - System has to be built and integrated to be evaluated
  - What if evaluation outcome is negative?
- Real-world evaluations are rare

---

---

**Slide Wearable Evaluation (3):**

- Idea: Implement parts of the system in a lab.
- “Living Lab” approache
- Question: How to simulate primary task in the lab?
- Aspects of the primary task:
  - Physical Task
  - Cognitive Task
  - Attention

---

---

**Slide Physical tasks:**

- Simple tasks: Walking, running, biking
- Strenuous tasks: running fast, carrying loads
- Manipulative tasks: push buttons, operate machines, use tools, select tools
- Precision tasks: handle tools carefully, avoid damage and spills

- Also physical tasks: input (e.g. gesture input)
- Body has physical limits: accuracy, force, energy limits

---

**Slide Cognitive tasks:**

- Simple tasks: Reading, Listening, Identify objects, following signs, “matching tasks”
- Complex tasks: calculations, translations, geometric tasks (see your favourite IQ test)
- Also cognitive tasks: input, understanding output
- Analog to physical limits: “cognitive load” limit
- cognitive load varies with age, familiarity with task, between persons

---

**Slide Matchingtask:**

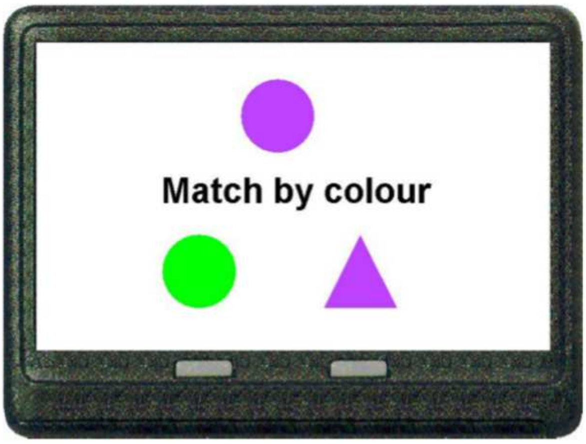



Image from H. Witt

---

**Slide Attention !:**

- Both physical and cognitive tasks need attention
- Attention is limited

- e.g.: you can only memorize a small (5-11) Number of things at the same time in your short time memory
- Some brain functions have limits: Humans only have one motor cortex
- Degrading attention leads to degraded performance: Precision lowers, reaction time rises, task execution takes longer
- Divided attention: affected by task similarity, task difference, practice

---

---

**Slide Measuring performance:**

- Idea: Use this information to craft artificial tasks to measure performance
- Cognitive taks: simple but measurable tasks, measure execution time and correctness
- Examples: Matching tasks, find repetitions in letter sequences, ...
- Physical tasks: Not too easy, but easy to measure
- Examples: Pushing buttons , “Hotwire experiment”
- Experiment:
  - Measure physical task w/o cognitive task
  - Measure cognitive task w/o physical task
  - Measure both together

---

---

**Slide The Hotwire experiment:**

- Origin: Children’s game, used to train hand-eye-coordination
- Conductive wire, bent in different shapes
- Conductive loop tool
- Task: move the loop tool over the wire without touching the wire

---

---

**Slide Hotwire:**

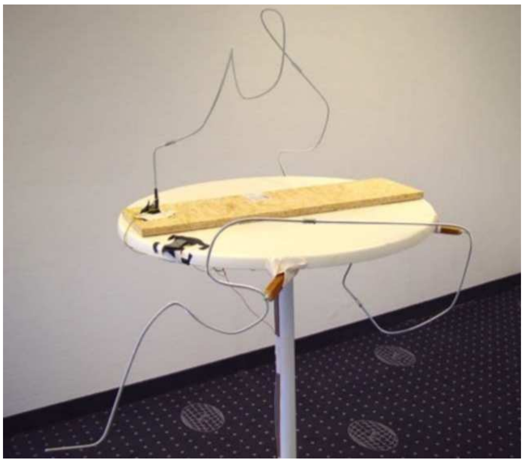


Image from H. Witt

**Slide Interruption by cognitive task:**

- Interruption studies: Well-known approach in HCI evaluation
- Matching task is presented to the user on a HMD
- Answer is given with gesture interface
- Different ways to present cognitive task
  - Immediate
  - Negociated
  - Scheduled
  - Mediated

**Slide Hotwire-Task:**

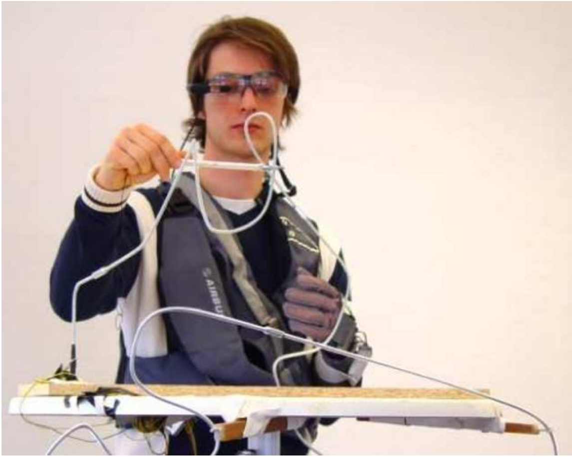


Image from H. Witt

---

**Slide Measuring Hotwire performance:**

- Time (to complete wire task)
- Contacts (tool-wire)
- Error rate (in matching task)
- Average age (Answer time for matching task)

---

**Slide Results:**

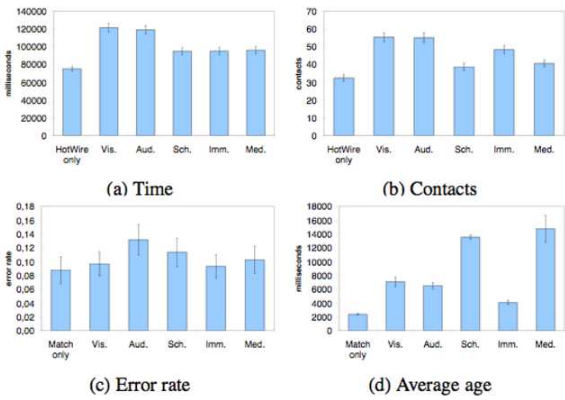



**Figure 5. Averages of user performance.**

Image from M. Drugge, H. Witt, ISWC06

**Slide Results:**

- Tasks have an influence to eachother
- Matching error rate almost unchanged
- Effect of the interruption methods
  - on Time: negotiated methods take longer
  - on Contacts: negociated methods have more errors (additional interaction)
  - on Error: nothing
  - on Average Age: unclear, side effects disturbe result

**Slide Larger Hotwire (on CeBit):**

Image from mrc

**Slide Summary:**

- Task Trees
  - Formal specification of user interaction
  - Can be used to support development

- ConcurTaskTrees
  - Temporal Operators
  - Examples

- AbstractUI
  - Device-independent
  - Context-aware

- WearableUI
  - Uses AbstractUI
  - Wearable interaction mode

- Evaluating wearable interfaces
  - simulate primary task
  - study effects of wearable use
  - use standardized experiments and measures for comparable results

---