Wearable Computing

Holger Kenn

Universität Bremen

WS 05/06

Overview of the course

- What is wearable computing?
- Building blocks of wearable computing
    1. Hard- and Software
    2. Wearable Human Computer Interfaces
    3. Context-Aware Computing
- Wearable computing research
- Wearable computer engineering

The Rules

1. Good scientific practice:
   1.1 Be precise and clear in statements. This is no literature class!
   1.2 Use scientific sources. (Journalistic articles aren't. Wikipedia isn't.)
   1.3 Cite! Plagiarism of ANY source is forbidden!
   1.4 Cooperate within limits: Talking OK, Copying not!
   1.5 Always use your own words.
2. Homework
   2.1 in groups, group size ≤ 3, no exceptions
   2.2 Fixed groups

The Grading

1. Homework + Fachgespräch
    1.1 At least 30% of the points of each problem sheet
    1.2 Homework grades serve as initial group grade
    1.3 Individual short oral exam to verify individual contribution.
2. Modulprüfung
    2.1 Homework recommended but points have no influence
    2.2 Longer oral exam

Wearable Computing?

- Evolution of computer hardware

1950s	Central computing facilities, Batch processing
1960-1980	Timesharing, "‘mini"’-computers
1980s	Personal Computers
1989	Nintendo Gameboy
1990	GSM Mobile Phones
1992	Apple Newton PDA

Wearable Computing?

- Evolution of computer hardware

1992	IBM Smartphone (sold by BellSouth in 1993)
1996	Digital Camcorder (Sony Digital8, MiniDV)
1998	MP3 Player (MPMan, Diamond Rio,...)
1999	Bluetooth SIG: Personal Area Network

Wearable Computing?

- Evolution of computer users

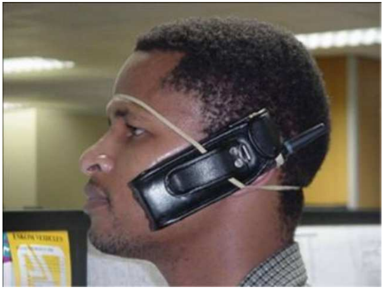
1950s	Experts build computers and use them
1960s	Computer companies: Commercial use
1980s	"‘Geek"’ user: Desktop, Home use (communication, games)
1990s	"‘Everyone"’ user: Mass Medium (Internet, Games)
2000s	"‘Everywhere"’ User (Notebook, PDA, Mobile Phone,...)

Wearable Computing?

- Evolution of computer uses

1950s	Special tasks, special computers
1960s	Business Support (Accounting)
1980s	Business, Information (BTX), Entertainment (video games)
1990s	Information (internet), Communication (e-mail, chat)
2000s	Mobile information and communication, casual use (SMS, WAP)

Is this wearable computing?!?



"‘Wearable"’ mobile phone?

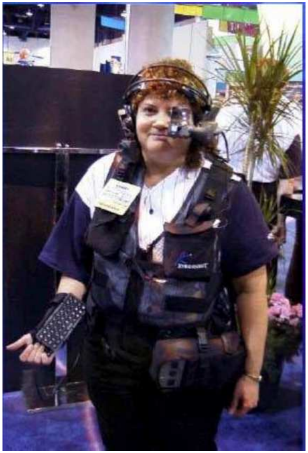
Is this wearable computing?!?

Can you imagine hauling around a large, light-tight wooden trunk containing a co-worker or an assistant whom you take out only for occasional, brief interaction. For each session, you would have to open the box, wake up (boot) the assistant, and afterward seal him back in the box. Human dynamics aside, wouldn't that person seem like more of a burden than a help? In some ways, today's multimedia portables are just as burdensome.

Steve Mann

Is this wearable computing?!?

Socially acceptable?



So what's Wearable Computing?

- Support the user (during a primary task)
  - Don't *disturb* the user
  - Provide useful functions *all the time*
- Seamless integration
  - Into existing processes and tasks
  - Into environment and social context

An Example: Symbol Wearable

> *In January of 1995, a major customer, the United Parcel Service (UPS), challenged Symbol Technologies to create a Wearable Data Collection device for their package loaders to use.*

(Stein et al: Development of a Commercially Successful Wearable Data Collection System, ISWC'98)

Symbol Wearable: Requirements

- Package loaders scan barcodes on packages
- Existing solution: Symbol APS3395 (from 1992): Three components (Display/Keyboard, CPU, Scanner) interconnected by wires
- fatigue-free scanning, 5.5h operation time
- improve hygiene, operation, reliability

Symbol Wearable: Challenges

- Ergonomics and Hygiene: safe, comfortable, unobtrusive
- Miniaturisation of computer and scanner
- Power management: 5.5h with single battery
- Ruggedization - Operation in hostile environment

Symbol Ws 1000

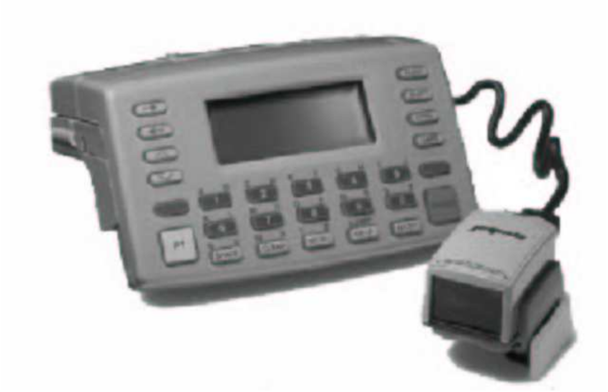


Image from Symbol Technology Inc.

Design Process

- "‘interactive design process"’: user testing, feedback
- Observation, Interviews, Scientific Literature
- concept scetches
- creating mockups
- user tests with mockups

User testing and feedback

- Interviews with many users, male and female
- Main concerns: comfort for wide range of body sizes, ruggedness, user safety, hygiene/cleanliness
- Design challenge: One Device for different hand and arm sizes, connecting "‘soft goods"’ to hard plastic

Scientific input

- Physiological studies: arm, hand, fingers in neural position most of the time
- Study of potential disease transmission: synthetic cloth materials for soft goods
- Ergonomic Evaluation in Lab: six subjects, simulation of UPS tasks

Steve Mann

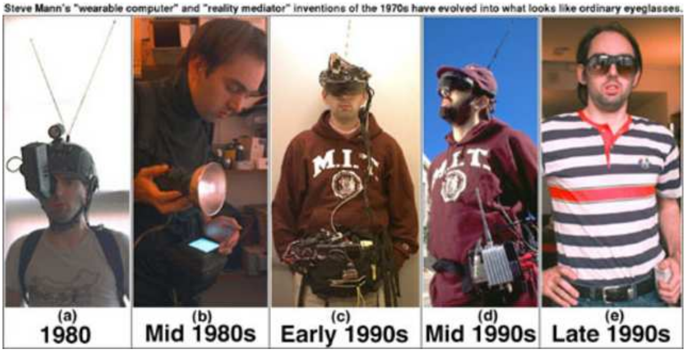

Steve Mann's "wearable computer" and "reality mediator" inventions of the 1970s have evolved into what looks like ordinary eyeglasses.
(a) 1980
(b) Mid 1980s
(c) Early 1990s
(d) Mid 1990s
(e) Late 1990s

Image from eyetab.org website

Humanistic Intelligence

HI is a new information processing framework in which the processing apparatus is inextricably intertwined with the natural capabilities of our human body and intelligence. [. . .] Devices that embody HI are worn continously during all facets of ordinary day-to-day living. Thorugh long-term adaptation thex begin to function as a true extension of the mind and body

Steve Mann, Intelligent Image Processing

Features of HI

- Relies on the existence of the human user
- operational constancy (always on)
- interactional constancy (inputs and outputs potentially always on)
- HI does not necessarily mean user-friendly, user learns from the device
- WearComp as means of realizing HI

WearComp Constancy

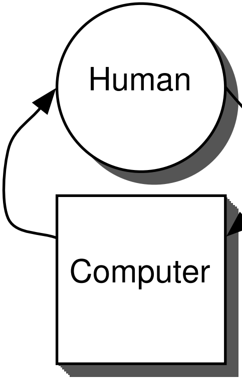


Image from Steve Mann: Intelligent Image Processing, Pg 5. Fig 1.1

WearComp Augmented Intelligence and Augmented Reality

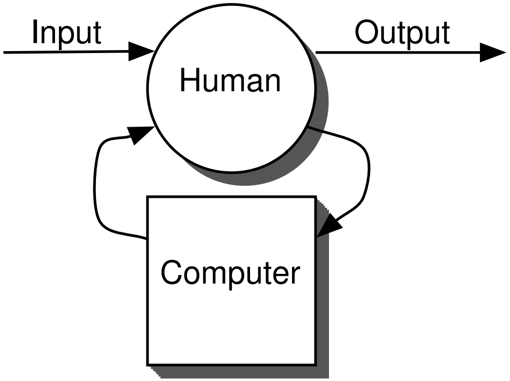


Image from Steve Mann: Intelligent Image Processing, Pg 5. Fig 1.1

WearComp encapsulating the user

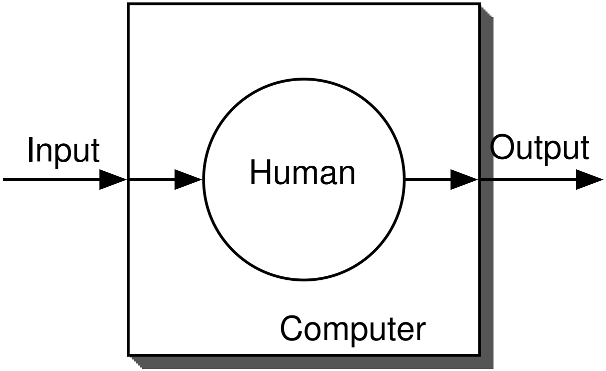


Image from Steve Mann: Intelligent Image Processing, Pg 5. Fig 1.1

WearComp encapsulation with Constancy and AR

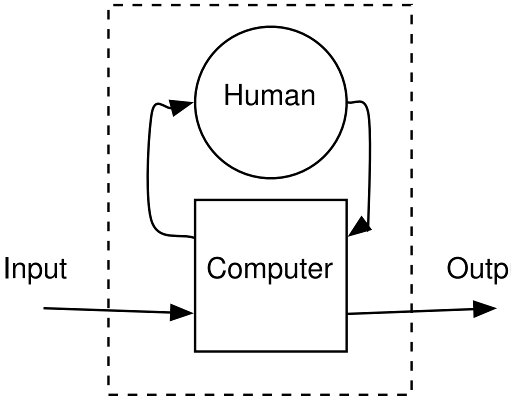


Image from Steve Mann: Intelligent Image Processing, Pg 5. Fig 1.1

Personal Imaging

Human beings optain their main sensory information from their visual sytem. [. . .] A computationally mediated visual reality is a natural extension of the next-generation computing machines. [. . .] This will not be done by implanting devices into the brain [. . .] , but rather by noninvasingly "tapping" the highest bandwidth pipe into the brain, namely the eye.

Steve Mann, Intelligent Image Processing Image from Steve Mann's eyetap.org website

Research on Wearable Computing

- Hardware
- Software
- Human Computer Interaction
- Context Recognition
- Cooperation (human-human and human-machine)

Research on Hardware

- Size, weight, shape

Image from ETH Zürich

- Energy source and management, heat dissipation

Research on Wearable Software

- Embedded Systems: Programming Language, Operating System
- Software Engineering: Software Structure, Engineering Method
- Distributed Systems: Service Architecture, Unreliability
- AI Techniques: Adaptivity, Knowledge Representation, Learning

Research on Wearable Human Computer Interaction

- User Studies: Interaction, Interruption, Physical Tasks
- Adaptivity: I/O Devices, Multimodality, Context-awareness
- Evaluation: Guidelines, Standards

Research on Wearable Context Awareness

- Sensor Signal Analysis: from Sensor to low-level context
- Localization: outdoor, indoor, sensor fusion, mapping
- High-level Context recognition: combination of low-level contexts
- Task Recognition: temporal sequences
- Learning Contexts: Supervised, Unsupervised machine learning

Research on Wearables for cooperation

- Human-Human: Situational Awareness
- Human-Machine: Common “world model”
- Cooperation: Working towards a comon goal
- Example: Robocup Rescue

Research Problems

- System engineering: How to build it? (so that it's useful?)
- Human Machine Interaction: How can it be used?
- Context-awareness: How does it know what's going on?
- Augmenting human capactiy: Building better cyborgs?
- ...

Wearable Computing!

- Properties of wearable computing
  - Unobtrusive
    - mobile, small, lightweight, no wires
    - body-wearable (sometimes in clothing)
  - Supporting a primary (work) task
    - Don’t disturb, be useful all the time
  - casual use, context-aware, "‘smart"’
- Engineering discipline or cyborg philosophy?

Timeline I

1268	Eyeglasses mentioned by Roger Bacon
1510	Pocket Watch invented by Peter Henlein in Nürnberg (“Nürnberger Ei”)
1665	Robert Hooke calls for augmented senses

Nürnberger Ei

Timeline I

1268	Eyeglasses mentioned by Roger Bacon
1510	Pocket Watch invented by Peter Henlein in Nürnberg (“Nürnberger Ei”)
1665	Robert Hooke calls for augmented senses

Timeline II

1907 Aviator Alberto Santo-Dumont commissions the creation of the first wrist watch by Louis Cartier

1945 Vannevar Bush (MIT) proposes the idea of a “memex”

1960 Manfred Clynes coins the word "Cyborg"

1966 First wearable computer: Ed Thorp and Claude Shannon: Analog computer for roulette wheel prediction.

Timeline III

1967 Hubert Upton invents analogue wearable computer with eyeglass-mounted display to aid lipreading

1972 Alan Lewis invents a digital camera-case computer to predict roulette wheels

1977 CC Collins develops wearable camera-to-tactile vest for the blind

1977 (C): HP releases the HP 01 algebraic calculator watch

HP-01

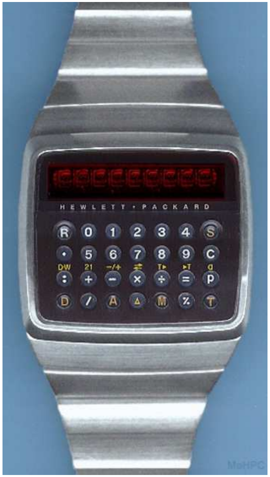


Image from www.hpmuseum.org

Timeline IV

1978	Eudaemonic Enterprises invents a digital wearable computer in a shoe to predict roulette wheels
1979	Sony introduces the Walkman
1981	Steve Mann designs backpack-mounted computer to control photographic equipment
1984	William Gibson writes Neuromancer
1987	The movie Terminator is released

Timeline V

- 1991 Doug Platt debuts his 286-based "Hip-PC"
- 1991 CMU team develops VuMan 1 for viewing and browsing blueprint data
- 1991 Mark Weiser proposes idea of Ubiquitous Computing in Scientific American
- 1993 Thad Starner starts constantly wearing his computer, based on Doug Platt's design and writes the first version of the Remembrance Agent
- 1993 BBN finishes the Pathfinder system, a wearable computer with GPS and radiation detection system

Timeline VI

1993	Feiner, MacIntyre, and Seligmann develop the KARMA augmented reality system
1994	Mik Lamming and Mike Flynn develop "Forget-Me-Not," a continuous personal recording system
1994	Edgar Matias debuts a "wrist computer" with half-QWERTY keyboard
1994	Steve Mann starts transmitting images from a head-mounted camera to the Web
1997	Creapôle Ecole de Création and Alex Pentland produce Smart Clothes Fashion Show

Wearable Computing Fashion

Image from http://www.media.mit.edu/wearables/lizzy/out-in-the-world/beauty/intro.html

Scientific Sources

Conferences	Cutting-edge research
Journals	Established Research
Books	“Textbook” knowledge
Research Groups	Lots of information, unreviewd

The first conferences

1996	DARPA sponsors “Wearables in 2005” workshop http://www.darpa.mil/MTO/Displays/Wear2005/index.html (no longer online, seen web archive link on course website)
1996	Boeing hosts a small wearables conference in seattle
1997	CMU, MIT and Georgia Tech co-host the ISWC’97

ISWC

- Since ’97 (Cambridge,Boston,MA)
- ISWC 07 back in Boston

IFAWC

- Since 2004
- IFAWC07 in Tel Aviv
- IFAWC08 not yet fixed

Other conferences

- Pervasive
- Percom
- Ubicom
- several workshops on specific topics: IWSAWC, NWUI

Journals

- IEEE Pervasive Computing
- Elsevier Pervasive and Mobile Computing
- Springer Personal and Ubiquitous Computing

MIT

Image from http://www.media.mit.edu/wearables/history.html

MIT

- MIT “Borglab”
- Part of the Media Lab
- http://www.media.mit.edu/wearables/
- Home of MIThril http://www.media.mit.edu/wearables/mithril/

MIThril

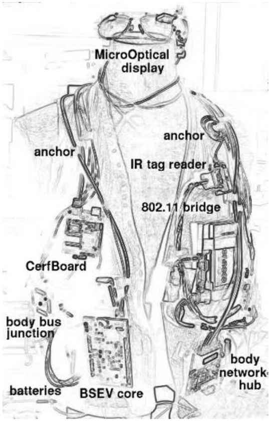


Image from http://www.media.mit.edu/wearables/mithril/photos.html

CMU

- Dan Siewiorek, Asim Smailagic
- http://www.ce.cmu.edu/ wearables/
- Home of VU-Man http://www.ices.cmu.edu/design/VuMan.html
- Project AURA http://www.cs.cmu.edu/ãura/

Toronto

- Steve Mann
- Philosophy of Wearable Computing
- Cooperation with artists
- http://www.eyetap.org/

Georgia Tech

- Thad Starner
- Focus on mobile HID

ETH

- http://www.wearable.ethz.ch/
- Gerhard Tröster
- QBIC Wearable Computer

Other places

- IBM LinuxWatch
  http://www.research.ibm.com/WearableComputing/
- Univ. Passau: Paul Lukowicz
- Univ of South Australia: Bruce Thomas
  http://people.unisa.edu.au/Bruce.Thomas
- Lancaster University: Hans Gellersen, Embedded Interactive Systems http://eis.comp.lancs.ac.uk/index.php
- TU-Darmstadt: Bernd Schiele, Multimodal Interactive Systems
  http://www.mis.informatik.tu-darmstadt.de/
- TZI Bremen

Output Device Classes

1. Optical
   - Body-mounted, Head-mounted, projection, ambient
2. Audible
3. Tactile

Wearable Displays

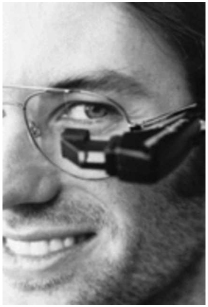


Image from TZI, T. Nicolai

Human Vision

- Spectral response: 400 to 700 nm, changes with age
- Adaptive resolution, 120 Megapixel (rods for greyscale)
- High resolution visual center (fovea), color receptors (6-7 million cones)
- 180 Degree low resolution with motion detection, greyscale
- High sensitivity about 15-20 degrees off the optical axis, single photon detection
- Integrated signal preprocessing for motion, edges, noise filtering

The Eye

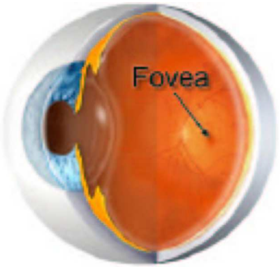


Image from Vorlesung Mensch-Maschine-Interaktion, LMU, Andreas Butz und Albrecht Schmidt

Human Vision: Accomodation and Sensitivity

- Adjustable lens system
- Focal range: 20 cm - ∞
- Dynamic range: 1 : 10^6 for dark environments, at least 1:10000
- Chemical (rhodopsin) and mechanical (iris) adaption of sensitivity

Human Vision: Resolution

- Angular resolution: Visual Acuity
- Measured with optometrician charts
- 20/20 (100%) visual acuity: Person can recognize a letter that spans less than a 5 minutes of arc visual angle
- Effective Resolution: about 1 arc minute

Effective resolution on a sheet of paper

- Viewing distance: 30 cm
- Paper Size: about 30x20 cm
- Viewing angle $2 \arctan \frac{1}{2} \approx 53^\circ$
- 53*60=3360 pixel
- 30cm =11.8 inch. $\frac{3360}{11.8} \approx 284 DPI$
- Why do people buy 1200 DPI printers?

Display Technology

- reflective, transflective, back-illumination, front-illumination
- B/W, greyscale, color
- CRT, LCD, TFT, OLED, DLP,...

Body-worn displays

- Wearable Computer displays
- re-used PDAs
- body-worn projection devices

Wrist-Displays

- Symbol
- Xybernaut
- IBM linux watch
- Fossil Wristwatch Palm

Symbol

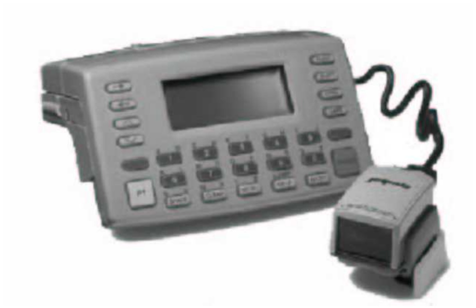


Image from Symbol Technology Inc.

Xybernaut

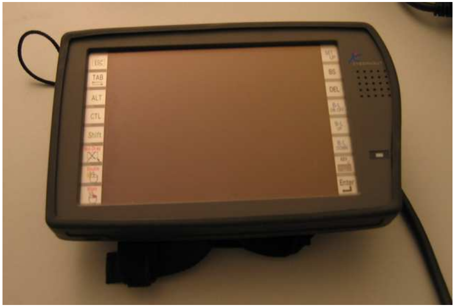
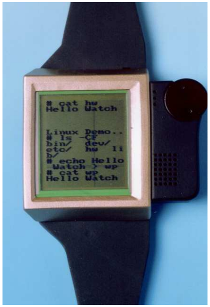


Image from TZI H. Kenn

IBM



```
# cat hw
Hello Watch


Linux Demo..
# ls -CF
bin/   dev/
etc/   hw   li
b/
# echo Hello
 Watch > wp
# cat wp
Hello Watch
```

Image from IBM

Fossil

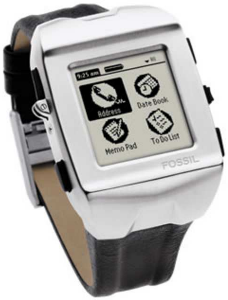


Image from fossil website

HMDs

- HMD = Head-Mounted Display
- Monocular vs. Binocular
- See-Through vs. See-Around
- Various resolutions, color and B/W

How do HMDs work?

- Eye minimum focal distance = 20 cm

Image from TZI H. Kenn

Focal Distance for HMDs

- Simulate aparent focal distance
- Additional optics
- calculation of resolution uses aparent focal distance

Lumus HMD

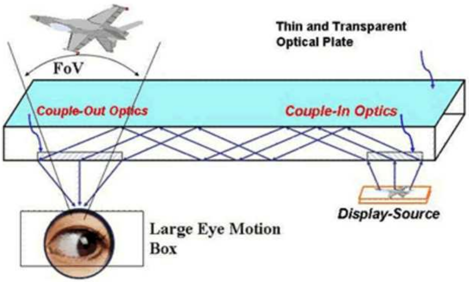


Image from lumusvision.com website

Xybernaut

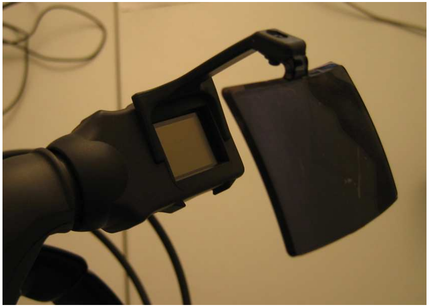


Image from TZI H. Kenn

Microoptical

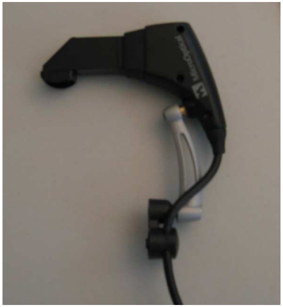


Image from TZI H. Kenn

Hearing

- Audible frequencies: 20-20kHz (for really young people)
- Dynamic range: 3dB-130dB (logarithmic scale! +3 dB = Energy $\times 2$)
- Equal loudness is frequency-dependent
- Hearing threshold is age-dependent

FletcherMunson Equal Loudness Contours

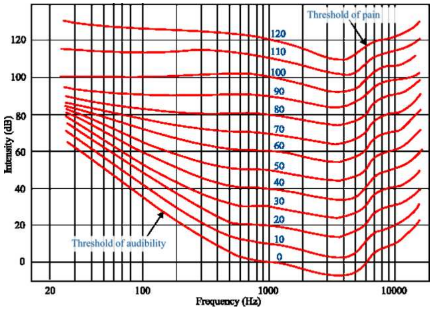


Image from wikipedia.org

Noise

- undesired disturbance affecting a signal. Here: acoustic noise
- Measured like sound
- Signal-to-noise ratio: ratio of signal levels of wanted (signal) and unwanted (noise) sound

Headphones

- Open vs closed
- Closed: high attenuation of noise, bulky, separation from environment
- open: low attenuation of noise, interaction with the environment possible

In Ear

- “earplug” style, found in many mobile devices
- exist in combination with body microphone technology

Active noise compensation

- Problem: environment noise
- But: interaction with the environment necessary
- Idea: record external noise through microphones, invert, play back through headphones
- possible: frequency-dependent noise compensation (only low frequencies)
- Implementations: Bose, Sennheiser
- Effect: -15dB noise reduction (Sennheiser PXC 250)

Sennheiser NoiseGard Headphone

Image from Sennheiser website

Sennheiser NoiseGard Controller

Image from Sennheiser website

Excenter-Vibration

- Needed if “output” needs to be unobtrusive (see roulette wheel prediction)
- Simple technologies: Motors, Solenoids
- Motor with excenter disk: Mobile phone “silent” alarm
- Solenoid: electromagnetic, delivers small “punch”, can be used for morse code

Force Feedback

- Part of input devices
- Simulates feedback force from a mechanical device
- simple implementations: Joystick, Racing Game Steering Wheel (simulate spring behaviour)
- Professional application: steering wheel feedback through “lane assistant”
- Professional application: telemedicine operation system, chirurgic training

Braille Displays

- Output of standard Braille letters
- Screen emulation
- Drivers for many operating systems
- Preinstalled in some linux distributions (Knoppix)

Braillex 40 char display

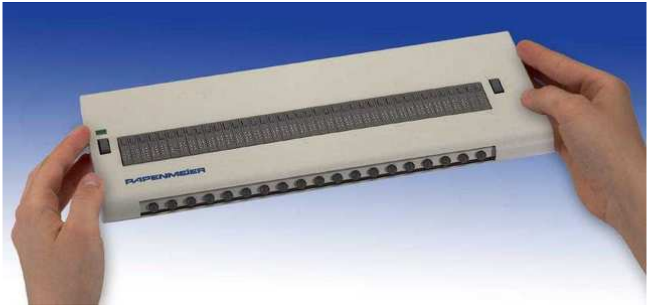


Image from papenmeier.de website

Braille PDA

- Linux PDA with keyboard and Braille display
- Normal PDA functions incl. e-mail and web access
- build-in ethernet and WLAN

Braillex Elba

Image from papenmeier.de website

Requirements for Wearables

- Wearable computing: support primary task
- Use computer while doing other things
- Goal: hands-free interaction
- Hands-free definition: interaction while using hands for primary task.
- Data-glove is sometimes considered “hands-free”

Text Input

- Typing
- Word Selection
- Pen Input
- Voice Input

Key Input

- Input Keys
- Command Keys (Backspace, Del, Cursor, Enter,...)
- Modifier Keys (Shift, Alt, Ctrl, Command)
- Keyboard Mode (Shift Lock, Num Lock)

Standard Keyboards

- Full-size (102-105 Keys, localized): >50 wpm, trained users much faster
- built-in (Notebooks, PDAs, Push Clients, . . .)
- wrist-mounted
- flexible

small PS2 keyboard

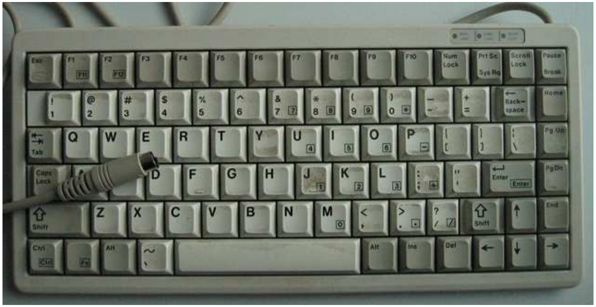


Image from H. Kenn

xybernaut arm keyboard

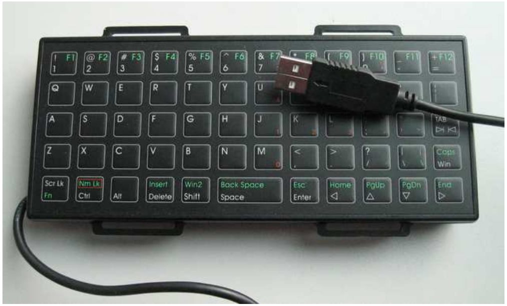


Image from H. Kenn

Indestructibe Keyboard

Image from H. Kenn

OQO PDA w. keyboard

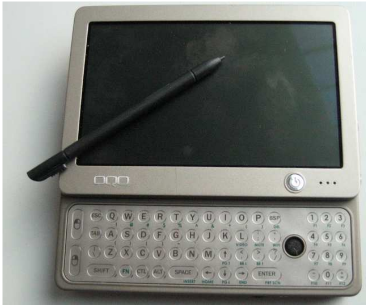


Image from H. Kenn

SK65 keyboard

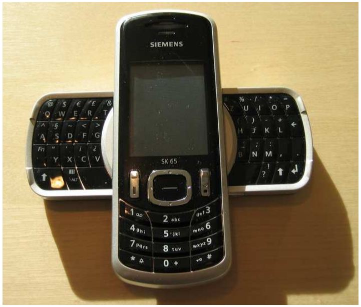


Image from H. Kenn

Wireless Standard Keyboards

- proprietary Infrared (Multimedia Remote Control)
- proprietary RF (“Wireless Desktop”)
- Bluetooth (HID-Profile)
- GSM Phones with HID Profile (e.g. K600i)

Stowaway Bluetooth keyboard

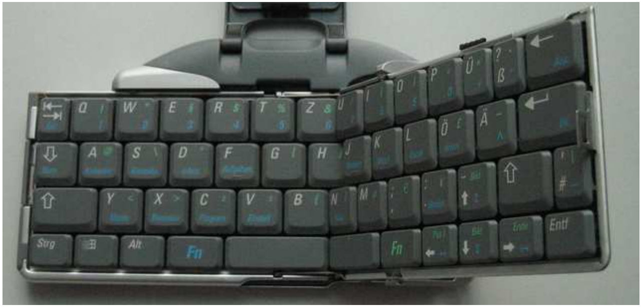


Image from H. Kenn

Custom Keyboards

- wired
- wireless
- textile-integrated

titv textile keyboard

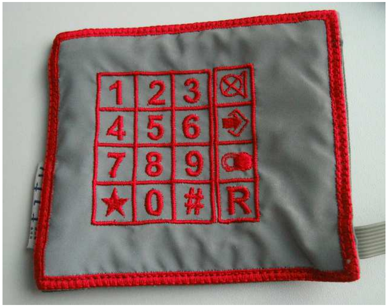


Image from H. Kenn

Chording Keyboards

- Idea: Multiple Keys pressed together create a single key event
- Result: Less keys
- one-hand blind typing (for trained users)
- Training needed, Impractical for untrained users

Twiddler

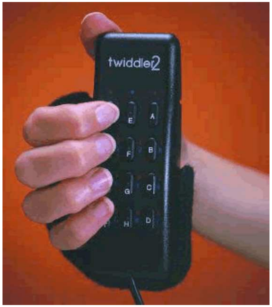


Image from handkey.com website

Frogpad

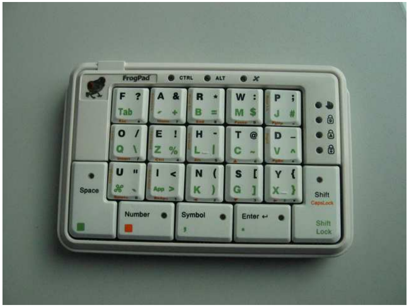


Image from H. Kenn

Phone Keyboard

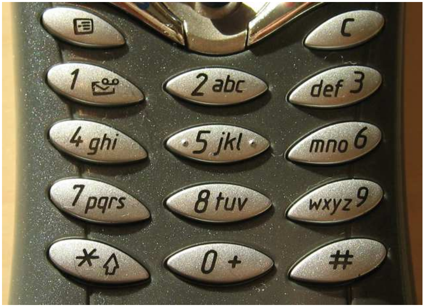


Image from H. Kenn

Multitap

- Origin: American “vanity number” letter codes
- Problem: Multiple letters on keys
- Solution: Select letter by tapping the key multiple times
- Timeout needed, Alternative: two-key (letter + index) or timeout key
- Maximum speed: 25-27wpm (w. timeout key), untrained users about 7 wpm

T9

- Predicting text input method
- invented by tegic communications, now owned by AOL
- Idea: type vanity keys without selecting the letter, use a dictionary to find a list of possible words
- Language-specific dictionaries, input language must be configured
- Shorthands for common words
- Timeout, selection keys and/or enter key needed
- Speed up to 46 wpm

Morse Key

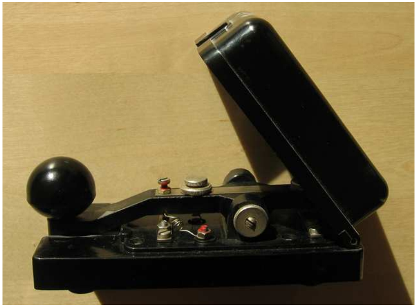


Image from H. Kenn

Morse Code

- Single Key, four symbols (dash, dot, short break, long break)
- Training required
- short codes (Q-code, Z-Code)
- 1939 speed record: 75.2 wpm (McElroy)
- still used in HAM radio
- QRQ Clubs (>40 wpm)

Pen Input

- Input devices: touch screen, tracking pen
- Touch Screen: Pressure sensitive (Palm) vs special pen (OQO)
- graphic only: UPS “electronic signature”
- tracking pen: optical (Anoto pen), motion sensor

Logitech IO Anoto Pen

Image from logitech.com website

Handwriting Recognition

- Hard problem
- Block Letters: easier
- smooth handwriting: tough
- Various standard products: PocketPC, Windows XP Tablet PC Edition

Graffiti

- As handwriting recognition is a hard problem, use a simplified set of strokes to ease recognition
- Palm Graffiti: single stroke letters
- Palm Graffiti2: multiple stroke letters, more similar to block letters

Graffiti

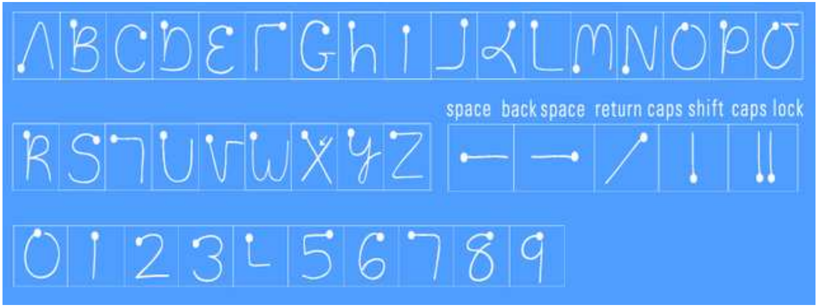


Image from palm.com website

Graffiti 2

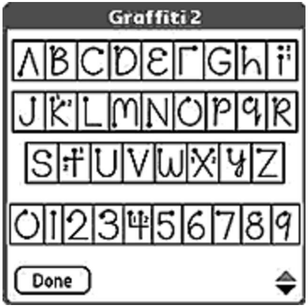


Image from palm.com website

Voice Input

- Goal: Computer “understands” “spoken language”
- General voice recognition unsolved, speech ambiguity $\rightarrow$ strong AI problem
- Several approaches: Speaker-dependent vs. Speaker-independent, large vs. small dictionary

Input Devices

- Microphones
- Problem: Signal/Noise ratio
- Solution 1: Move microphone closer
- Headsets, invisio
- Solution 2: Ignore noise
- directional microphones
- Multiple microphones, beamforming (used in speakerphones)

Command based

- Problem: when is information relevant for the computer
- Solution: Magic Word
- Scifi example: Star Treck: Commands start with “computer!”
- Commercial implementations: Sony Ericsson phones voice dial
- Alternative: Push-to-talk

Few words, speaker independent

- Typical application: automated phone services
- Typical words: Yes, No, numbers
- Sometimes larger dictionaries: Automatic timetable service
- Try it yourself: Deutsche Bahn Toll-free 0800 1 50 70 90

Many Words, few speakers

- Training required
- uses machine learning and dictionaries
- specialized professional dictionaries: medicine, law
- Example: IBM ViaVoice

Pointing, Selection, Gesture

- Complementing keyboard
- Often more efficient
- In many application, a text entry system is still needed.

WIMP Methaphor

- Windows, Icons, Menus, Pointer
- Standard for desktop
- Comparable interfaces exist for PDA: Pen controls Pointer
- not really suited for wearable use

Finger Trackball

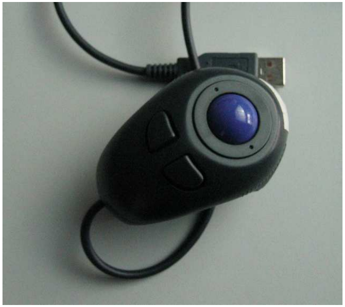


Image from H. Kenn

Twiddler Trackpoint

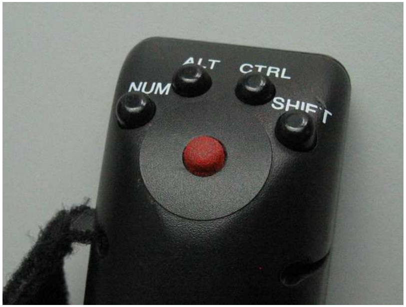


Image from H. Kenn

Ultrasound 3d Mice

- Uses body-mounted ultrasound transmitters and receivers
- Tracks hand motion in 3D

Image Processing-based

- Using a camera to recognize gestures
- hard problem: find hand, track hand, recognize gesture
- even harder in wearable environment
- Implementation: Fingermouse from ETH Zuerich (Patrick de la Hamette)

Finger Mouse

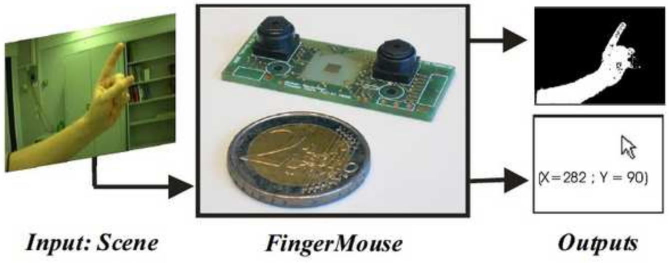
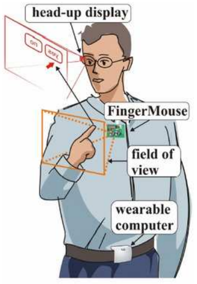


Image from P. de la Hamette, ISWC2006 Poster

Finger Mouse



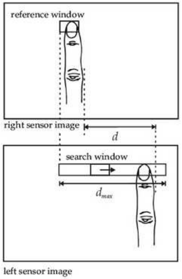


Image from P. de la Hamette, ISWC2006 Poster

Gesture

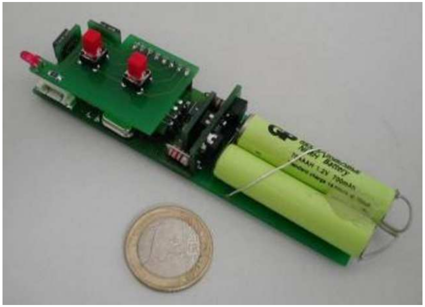


Image from beecon.de website

Glove

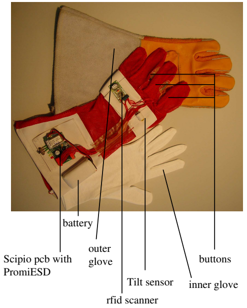


Image from H. Kenn

WInspect Glove

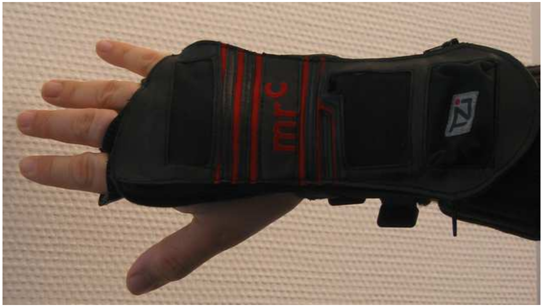


Image from H. Kenn

GestureBand

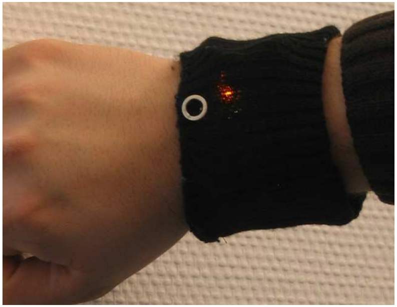


Image from H. Kenn

Human Computer Interaction

- Research Topic: Understand Human Computer Interaction
- Engineering Topic: Build interactive systems
- “Cognitive” Ergonomics: Physiology, Psychology (and Sociology)
- Business impact: HCI design important for product success

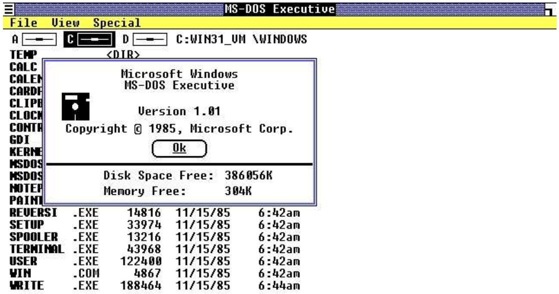
HCI History and People

Ivan Sutherland	“Sketchpad: A Man-Machine Graphical Communications System” First GUI, light pen device, 1963
Doug Englebart	Mouse
Ted Nelson	Hypertext, 1970
Alan Kay	Smalltalk: OO-Programming language + operating system + user interface
1982-	GUI Systems: Xerox Star, Apple Lisa, Apple Macintosh
1985	Windows (birthday 20.11.1985, yesterday!)

MS Windows 1.0

Image from heise.de website

MS Windows 1.0 screenshot



```
C:WIN31_VM \WINDOWS
TEMP            <DIR>
CALC
CALEN
CARDF
CLIPB
CLOCK
CONTR
GDI
KERNE
MSDOS
MSDOS
NOTEP
PAINT
REVERSI  .EXE   14816  11/15/85   6:42am
SETUP    .EXE   33974  11/15/85   6:42am
SPOOLER  .EXE   13216  11/15/85   6:42am
TERMINAL .EXE   43968  11/15/85   6:42am
USER     .EXE  122400  11/15/85   6:42am
WIN      .COM    4867  11/15/85   6:42am
WRITE    .EXE  188464  11/15/85   6:44am
```

```
Microsoft Windows
MS-DOS Executive

Version 1.01
Copyright © 1985, Microsoft Corp.

Disk Space Free:  386056K
Memory Free:      304K
```

Image from heise.de website

What HCI is about...

- People
- Activities
- Contexts
- Technologies

People

- Physical Differences
- Psychological Differences
- Usage Differences

Activities

- Temporal Aspects
- Cooperation
- Complexity
- Safety-Critical
- Content

Context

- Physical Environment
- Social Context
- Organizational Context

Technology

- Input
- Output
- Communication
- Content

PACT Framework

- PACT Analysis
- Development of personas
- Example: Sales Clerk
- Example: Technical Inspector

Principles and Practice

- Accessibility
- Usability
- Acceptability
- Engagement

Accessibility

- Don’t Exclude Users!
- Physically
- Conceptually
- Economically
- Cultural Exclusion
- Social Exclusion

Usability

- efficient
- effective
- easy to learn
- safe to operate
- high utility

Acceptability

- Legal
- Political
- Convenience
- Cultural and social habits
- Usefulness
- Economic

Engagement

- Is it a “Killer App”?
- Identity
- Adaptivity
- Narrative
- Immersion
- Flow

Design Principles I

- Visibility
- Consistency
- Familiarity
- Affordance

Design Principles II

- Navigation
- Control
- Feedback
- Recovery
- Constraints

Design Principles III

- Flexibility
- Style
- Convivality

Examples

- Design Windowed Applications
- Website Design
- Other things (like Wearables)

Theories of HCI

- Is PACT a Theory?
- PACT is best practice approach for requrement analysis, but can't say if a system built performs well
- Lack of predictive power: PACT is an approach for requirement analysis
- Low-level theories: Input, Output
- . . . cannot predict the performance of a complete system
- HCI-Theroies needed

Levels of analysis theory

- Four levels of analysis: conceptual, Semantic, Syntactic, lexical
- conceptual: describes the user's mental model. (Text Processing with Word/Latex/Page Maker)
- semantic: meanings of user actions: delete a paragraph
- syntactic: select paragraph with mouse, select “delete” from menu
- lexical: move mouse cursor, click, press function key,. . .
- Clean top-down-approach: good for designers
- . . . but less relevant today, as systems are very complex

Stages of action theory

- Explanatory thesis of HCI, Norman (1988)
- 7 Stages (“executed” in a cyclic way by the user):
    1. Forming the goal
    2. Forming the intention
    3. Specifying the action
    4. Executing the action
    5. Perceiving the system state
    6. Interpreting the system state
    7. Evaluating the outcome

Stages of action theory

- Norman suggests four principles of good design:
  1. State and action alternatives should be visible
  2. Good conceptual model with consistent system image
  3. The interface should include good mappingm that reveal the relationships between the stages
  4. Users should receive continuous feedback
- Question: is this applicable to wearable computing?

GOMS

- Originated from CMU: Decompose user actions into small measurable steps
- GOMS: Goals, Operators, Methods, Selection rules
  1. Goals and subgoals: Edit text, delete paragraph
  2. Operators: Move mouse, press mouse button, check if mouse cursor is at the end of a paragraph but also: recall file name, search for menu option
  3. Methods (to reach goal): Move mouse, click button, press delete to delete a paragraph
  4. Selection rules (select one of many methods): Delete Paragraph with “delete” key, use “delete” menu entry, use multiple “backspace” to delete paragrpah...

keystroke-level models

- Also from CMU, same idea as GOMS, but simplified
- Predict (error-free) task time by summing up time for elementary actions
- keystrokes, mouse moves, thinking, waiting, ...
- uses a simplified “human processor” model
- good for modeling error-free tasks performed by experts
- does not model errors, learning, problem solving ...
- Other GOMS-Derivatives: NGOMSL (Kieras, 1988), CPM-GOMS (used to predict performance of extremely skilled users) ...

Consistency

- Idea: Make consistency checkable
- Use a grammar to describe the user interaction
- Reisner (1981) action grammar: UI with simpler grammar is easier to learn
- Payne and Green (1986) Task Action Grammars: multiple levels: (lexical, syntactical, semantic consistency), Completeness check

Widget-level theories

- Instead of decomposing along elementary tasks, use decomposition of high-level UI toolkits
- Create model based on widgets and predict user performance based on widgets used
- Interface model emerges from implementation task, estimates of perceptual complexity and motoric skills needed emerges as well
- Goal: develop well-established UI patterns (with predictive model of user performance attached)

Context-of-use theories

- Problem with previous models: based on “lab” experiments
- The real world has context, not only HCI
- Suchman(1987) Plans and Situated Action
- Mobile (and wearable!) computing: physical space becomes relevant
- (Dourish, 2002) social/psychological space also has to be considered

Object Action Interface Model

- descriptive and explanatory model
- can also be used to guide design
- Observation: syntax becomes simplerin modern GUI systems
- Object Action Design: Decompose Objects and Actions
- Objects may include “real world objects”, Tasks may include “common activities”

Examples

- Design Windowed Applications
- Website Design
- Other things (like Wearables)

Project WINSPECT

- TZI & Stahlwerke Bremen (Steelmill)
- Topic: Wearable Solution for inspection of industrial cranes

Winspect

Image from T. Nicolai

Winspect

Image from T. Nicolai

Winspect

Image from T. Nicolai

Sensors and Wearables

- User
- Place
- Task
- Environment

Sensing the User

- Presence of a user
- Actions and Tasks
- Body and Mind

Simple example: Presence of the user

- Useful for the wearable computer: Save energy
- Useful for other systems: Communicate the presence of a user
- Useful for the user: No “on”-Switch...

Detecting User Presence

- Tactile: Design switches that detect

  Pro: Easy to implement (binary input)
  Con: Can be fooled.
- Temperature: Design sensors that detect body heat

  Pro: Harder to fool (but still possible), easier to integrate
  Con: harder to interface, computation needed
- Motion: Detect body motion

  Pro: Even harder to fool
  Con: even harder to interface, computation and signal analysis needed

The details . . .

- Task: Detecting users of a body-worn wearable computer
- Mechanics: User's Body presses switch when user is present
- Textile: pressure-sensitive textiles
- Electronics: Schmitt trigger circuit
- Computer I/O: Binary input
- System Software: event-driven, similar to ACPI switches
- Application Software: Event API

Sensors and signals

- A *sensor* is a transducer that is used for measurement
- (A *transducer* is a device that converts one form of energy to another)
- Sensors measure a property of the physical world . . .
- . . . and produce a corresponding signal that can be processed.
- Evaluating the signal at one fixed point in time is called a *measurement*
- A number of measurements taken at different times is called a *time series*
- Typically, these measurements are taken *sequencially at fixed equal time intervals*, i.e. once every second

Signals and Computers

- Typically, sensors produce electrical signals
- A property of the electrical signal (current, voltage, frequency, pulse width, phase) corresponds to the property measured by the sensor.
- In order to use the signal in a computer, the relevant property of the signal needs to be converted into a binary representation.
- This process is called A/D-conversion. It is performed by an A/D-converter.
- Typical A/D-converters convert the voltage of a signal into a binary representation.

Sampling

- Performing the digitalization of a signal at a fixed point in time is called sampling. Sampling is a form of measurement.
- Signals are continous, a binary representation produced by an A/D-Converter is discrete.
- Multiple signal values reproduce the same binary representation. This phenomenon is called *quanitzation*
- Through quantization, information is lost, resulting in the so-called *quantization error* for a single measurement and *quantization noise* for the whole signal.

How often do we need to sample

- Sampling in regular, fixed time intervals T produces a time series of binary values.
- The frequency $\frac{1}{T}$ is called sampling frequency.
- Question: How often do we have to sample? Answer: Depends on the signal
- The faster and the more often the signal changes, the more often we have to sample.
- Rule of thumb: Sample slightly more than twice as often as the period of the highest frequency component in the signal.
- Example: CD-Player Signals $\leq$ 20kHz, sample rate 44.1kHz

Classification of sensors

- Measured property
- Type of measurement (absolute vs. relative)
- Dimensionality
- limiting factors: precision, noise, frequency response,...

Measurable properties

- Distance and Position
- Motion
- Light
- Temperature

Distance and Position

- Signal Attenuation
  - Propagation of a signal in a medium
  - Assumption: propagation path attenuates signal
  - Original signal strength s_1, received signal strength s_2
  - Attenuation depends on signal properties and path properties
- Time-Of-Flight measurement
  - Propagation of a signal in a medium
  - Assumption: constant propagation speed v
  - Signal transmitted at time t_1, received at time t_2.
  - $d = (t_2 - t_1)v$

Position from distance and angle

- Triangulation
  - Known distances from three known points
  - Known distance differences from four points
  - Example: GPS
- Angle-of-arrival
  - Position from three angles to known points
  - Two angles sufficient for 2D positioning
  - Example: Ships

Motion

- Differential position
- Doppler
  - Frequency of received signal changes with relative motion
  - Self-transmitted or remote signal
- Inertial Measurement
  - Motion can be measured through accelleration
  - Accelleration can be measured
  - linear accelleration and rotation

Sensors and Wearables

- User
- Place
- Task
- Environment

Sensing the User

- Presence of a user
- Actions and Tasks
- Body and Mind

Simple example: Presence of the user

- Useful for the wearable computer: Save energy
- Useful for other systems: Communicate the presence of a user
- Useful for the user: No “on”-Switch...

Detecting User Presence

- Tactile: Design switches that detect

  Pro: Easy to implement (binary input)
  Con: Can be fooled.
- Temperature: Design sensors that detect body heat

  Pro: Harder to fool (but still possible), easier to integrate
  Con: harder to interface, computation needed
- Motion: Detect body motion

  Pro: Even harder to fool
  Con: even harder to interface, computation and signal analysis needed

The details . . .

- Task: Detecting users of a body-worn wearable computer
- Mechanics: User's Body presses switch when user is present
- Textile: pressure-sensitive textiles
- Electronics: Schmitt trigger circuit
- Computer I/O: Binary input
- System Software: event-driven, similar to ACPI switches
- Application Software: Event API

Sensors and signals

- A *sensor* is a transducer that is used for measurement
- (A *transducer* is a device that converts one form of energy to another)
- Sensors measure a property of the physical world . . .
- . . . and produce a corresponding signal that can be processed.
- Evaluating the signal at one fixed point in time is called a *measurement*
- A number of measurements taken at different times is called a *time series*
- Typically, these measurements are taken *sequencially at fixed equal time intervals*, i.e. once every second

Signals and Computers

- Typically, sensors produce electrical signals
- A property of the electrical signal (current, voltage, frequency, pulse width, phase) corresponds to the property measured by the sensor.
- In order to use the signal in a computer, the relevant property of the signal needs to be converted into a binary representation.
- This process is called A/D-conversion. It is performed by an A/D-converter.
- Typical A/D-converters convert the voltage of a signal into a binary representation.

Sampling

- Performing the digitalization of a signal at a fixed point in time is called sampling. Sampling is a form of measurement.
- Signals are continous, a binary representation produced by an A/D-Converter is discrete.
- Multiple signal values reproduce the same binary representation. This phenomenon is called *quanitzation*
- Through quantization, information is lost, resulting in the so-called *quantization error* for a single measurement and *quantization noise* for the whole signal.

How often do we need to sample

- Sampling in regular, fixed time intervals T produces a time series of binary values.
- The frequency $\frac{1}{T}$ is called sampling frequency.
- Question: How often do we have to sample? Answer: Depends on the signal
- The faster and the more often the signal changes, the more often we have to sample.
- Rule of thumb: Sample slightly more than twice as often as the period of the highest frequency component in the signal.
- Example: CD-Player Signals $\leq$ 20kHz, sample rate 44.1kHz

Classification of sensors

- Measured property
- Type of measurement (absolute vs. relative)
- Dimensionality
- limiting factors: precision, noise, frequency response,. . .

Measurable properties

- Distance and Position
- Motion
- Light
- Temperature

Distance and Position

- Signal Attenuation
  - Propagation of a signal in a medium
  - Assumption: propagation path attenuates signal
  - Original signal strength s_1, received signal strength s_2
  - Attenuation depends on signal properties and path properties
- Time-Of-Flight measurement
  - Propagation of a signal in a medium
  - Assumption: constant propagation speed v
  - Signal transmitted at time t_1, received at time t_2.
  - $d = (t_2 - t_1)v$

Position from distance and angle

- Triangulation
  - Known distances from three known points
  - Known distance differences from four points
  - Example: GPS
- Angle-of-arrival
  - Position from three angles to known points
  - Two angles sufficient for 2D positioning
  - Example: Ships

Motion

- Differential position
- Doppler
  - Frequency of received signal changes with relative motion
  - Self-transmitted or remote signal
- Inertial Measurement
  - Motion can be measured through accelleration
  - Accelleration can be measured
  - linear accelleration and rotation

From Sensors to Context

- Sensors measure aspects of the real world
- Sometimes, multiple sensor measurements are combined to measure another aspect
- Example: Position derived from time difference, angles, proximity to beacons
- Question: How do we determine the “right” thing for the wearable to do when we have (combined) sensor information
- Example: Location-based services, supporting mobile workers
- In general: How can we combine all the sensor measurements to a concept that we can use to write an application program for the wearable?

Context?!?

- Activity recognition and task context: What is the user doing?
- Physiological monitoring and user context: How is the user doing?
- Environment and situational context: Where is he and what is going on around him?

How to identify the context?

- Prior knowledge: User model, configuration, external data sources
  Disadvantage: tedious to aquire, outdated, wrong, expensive
- Explicit interaction: Asking the user (“Press OK to continue”)
  Disadvantage: impact on the user, lowers user acceptance
- Sensors
  Disadvantage: open research question

Dealing with Context

- Context has both a qualitative and quantitative nature
- Sensors give only quantitative information (“The temperature at sensor position 3 is 29.4 degrees celsius”)
- Problem: Making qualitative from quantitative information (“It’s too hot.”)
- Solution: Explicit programming

```
if (position(3)->TempSensor()->curentValue() > 29)
    setTemperatureContext(CONTEXT_TEMP_TOO_HOT);
}
```

- Question: Implementation complexity? Application Domain Expertise?
- Better Solution: Supervised machine learning (see problem sheet)
- Question: Reusability?

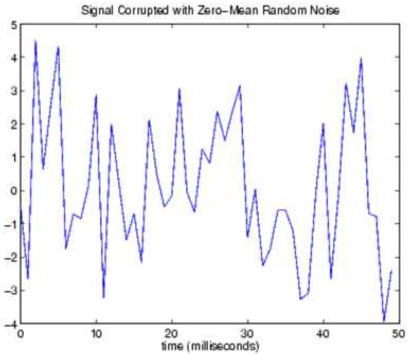
Processing context

1. Reading sensors
2. Combining sensors
3. Detecting simple context
4. Combining contexts
5. Using context information

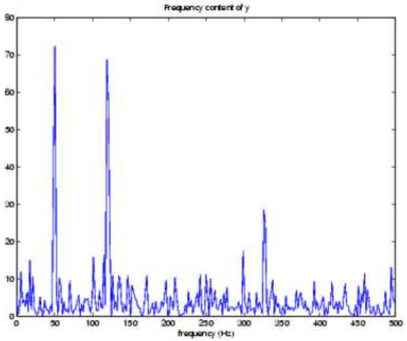
Detecting simple contexts

1. Assign “meaning” to sensor values
2. Most simple approach: Tresholds (as seen before)
3. Often needed: filtering
   - temporal filtering: combine several measurements from different times
   - simple example: low-pass-filter (limits high-frequency noise)
   - Integration, Derivatives
   - Time domain → Frequency Domain: Fourier Transform
4. Signal analysis: Pattern recognition, state machines, Hidden Markov Models, Artificial Neural Networks, . . .
5. Combining contexts: first order logic, temporal logic, spatial logic

FFT example


Signal Corrupted with Zero−Mean Random Noise
5
4
3
2
1
0
−1
−2
−3
−4
0
10
20
30
40
50
time (milliseconds)

FFT example


Frequency content of y
0
10
20
30
40
50
60
70
80
0
50
100
150
200
250
300
350
400
450
500
frequency (Hz)

Using Context in an application

- Query-based

```
if (getContext(CONTEXT_TEMP) == CONTEXT_TEMP_TOO_HO
    {...}
```

- Event-based (Publisher-Subscriber-Pattern)

```
RegisterContextSubscriber(CONTEXT_TEMP,
          CONTEXT_TEMP_TOO_HOT,
          TempContextSubscriberCallback);
...
void TempContextSubscriberCallback(context current)
{
 cout << "Too hot!" << endl;
}
```

Context processing frameworks

- Question: How to integrate all steps of context processing
- Context Toolkit (Salber, Dey and Abowd)
  - object-oriented toolkit
  - uses widget and event concept from GUIs
  - organized context detection, but does not define specific mechanism
- Context Recognition Network (CRN) Toolbox: Banach, Kunze, Lukowicz, Amft
  - organizes filters and classifiers for sensor data
  - Supports reuse: filters and classifiers can be parameterized and combined by application programmers
  - Separation of concerns: Experts program and parameterize filters, application programmers use combined filters
  - no learning, experts conduct experiments and build filters

The Context Toolkit

- Middleware for context processing
- Designed by Dey and Abowd
- Idea: Instead of redesigning context-aware applications from scratch, build them with a reusable middleware
- Uses GUI-Like metaphor: Context Widgets

Context according to Dey & Abowd

> *Context: any information that can be used to characterize the situation of entities (i.e. whether a person, place or object) that are considered relevant to the interaction between a user and an application, including the user and the application themselves. Context is typically the location, identity and state of people, groups and computational and physical objects.*

Dey & Abowd, Towards a better understanding of context and context-awareness, 2000

Context

- Entities: Places, People and Things
- Categories: Identity, location, status (or activity), time
- Application uses *context-aware functions* related to
  1. Presentation of information
  2. Execution of services
  3. Storage of context-information (attached to other stored items)

Examples for functions presenting information

- Map display of surroundings and points of interest (aka Location-based systems)
- in/out information of a group of users
- ambient information displays
- remote awareness of others (IMs, Skype)

Examples for functions executing services

- Teleport system: User's Desktop follows from Workstation to Workstation (Want et al, 1992) (commercial example: Sun Ray)
- Car navigation systems that recompute the path automatically on traffic information, wrong turns etc.
- Recording whiteboard senses ad-hoc meeting begin (Brotherton, Abowd and Truong, 1999)
- Mobile devices change settings according to context change
- Location-aware reminders

Examples for functions storing context information

- storing time of recording (digital cameras)
- storing location of recording (zoology application, Pascoe, Ryan and Morse)
- storing meeting information (people present, when, where meeting took place)
- Memory augmentation (Forget-Me-Not, Lamming & Flynn, 1994, Remembrance Agent, Rhodes, 1997)

Why handling Context ist difficult...

Day, Abowd and Salber claim:

Handling context is difficult for at least three reasons:

1. *there are no guiding principles to support good software engineering practices*
2. *designers lack abstractions to think about context*
3. *context sensing is very often distributed and leads to complex distributed designs*

Requirements for Dealing with Context

Day, Abowd and Salber claim:

> *. . . we have identified a number of requirements that the framework must fulfill to enable designers to more easily deal with context. These requirements are*
>
> 1. *Separation of concerns*
> 2. *Context interpretation*
> 3. *Transparent, distributed communications*
> 4. *Constant availability of context acquisition*
> 5. *Context storage*
> 6. *Resource discovery*

Separation of concerns...

- Previous application hardcoded sensor data aquisition, signal analysis, context detection and processing. This requires the application programmer to know a lot. (See last problem sheet. . .)
- Can we handle Context like we handle User Input? Applications don't care how a letter is typed (or a button is clicked). . .
- Separation of concerns: Let the toolkit deal with the acquisition of context, let the application programmer deal with using context in his application.
- UI-Widgets are a mechanism for this: All widgets have a common external interface, query and callback mechanisms, an application can treat all widgets (i.e. contexts) the same.

Separation of Concerns

Day, Abowd and Salber claim:

> *By separating how context is acquired from how it is used, applications can now use contextual information without worrying about the details of a sensor and how to acquire context from it. These details are not completely hidden and can be obtained if needed.*

Context Interpretation

- Applications need high-level context information (i.e. User is in a meeting)
- Sensors provide low-level context information (User is in room x, sound level is moderate, there are 6 other users present, the user is sitting etc.)
- The low level context must be interpreted to form high-level context.
- This can be done in the application, but it would not be reusable
- Or it can be done in the toolkit, so that other applications can reuse the context interpretation.

Transport, Discovery

- Context is often acquired by different computer systems
- . . . and may be used by several applications at the same time
- A network-transparent mechanism for context transport is needed.
- The application programmer may not know which context sources are available
- Therefore, a discovery mechanism is needed.

Availability, Storage and History

- An event determining the current context may happen before the application asks for it
- The context acquisition system has to be running all the time
- Applications may need previous contexts for interpretation
- Old context information has to be stored
- Some interpretation is only possible by looking at a previous sequence of contexts
- Context History has to be stored, i.e. a sequence of old contexts
- This may need a lot of storage, so a good selection mechanism is needed.

Context Abstractions

The Context Toolkit abstracts context in the following way:

- Context Widgets : Provides Context, interface to a sensor
- Interpreters : Raises the level of abstraction
- Aggregators : Optimize the flow of data by combining local context sources
- Services : Execute actions, counterpart to widgets
- Discoverers : register capabilities available

Context Toolkit for ActiveBadge example

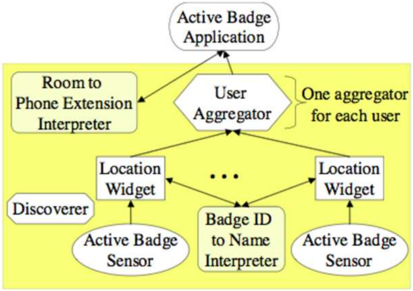


Image from Day, Abowd and Salber, J. of HCI

Context Toolkit for Intercom

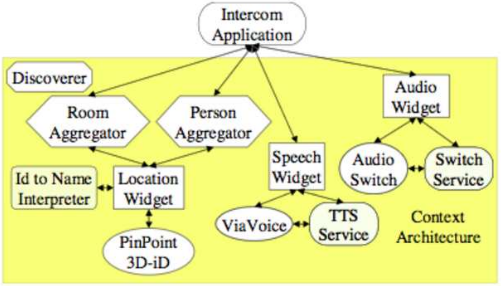


Image from Day, Abowd and Salber, J. of HCI

Design Methodology

The Context Toolkit authors suggests the following design methodology

- Identify Enitites
- Identify Context Attributes
- Itentify Quality of Service Requirements
- Choose Sensors
- Derive a Software Design

Context Recognition Network Toolbox

- Written by David Bannach (ETH/UMIT/Uni Passau)
- Implements a network of readers, filters, classifiers, recorders, writers
- Distributed: can pass data over network links
- Java editor, Interface to Matlab

CRN Components

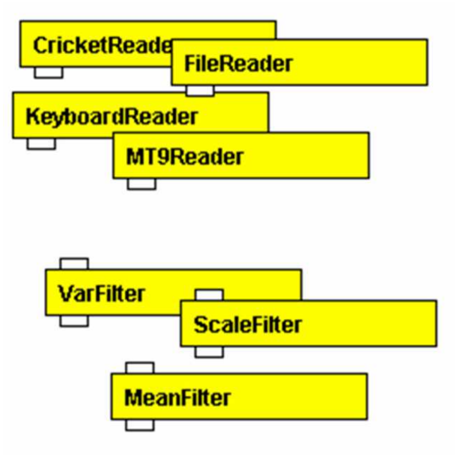


Image from D. Bannach, Uni Passau

CRN Components

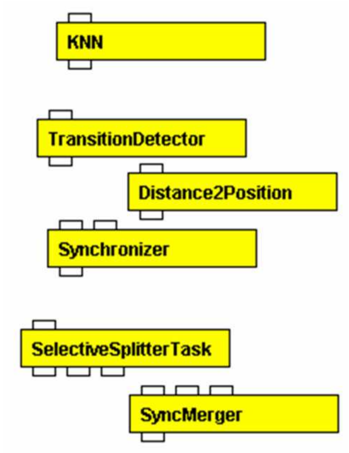


Image from D. Bannach, Uni Passau

CRN Components

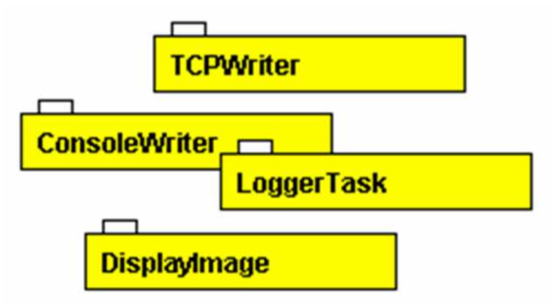


Image from D. Bannach, Uni Passau

CRN Toolbox Chain

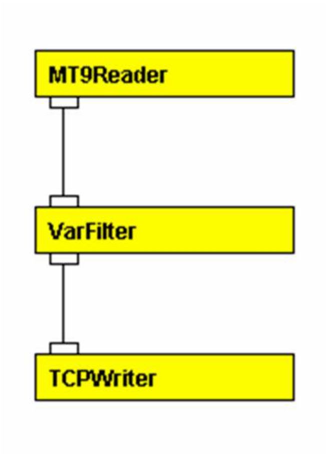


Image from D. Bannach, Uni Passau

CRN Editor

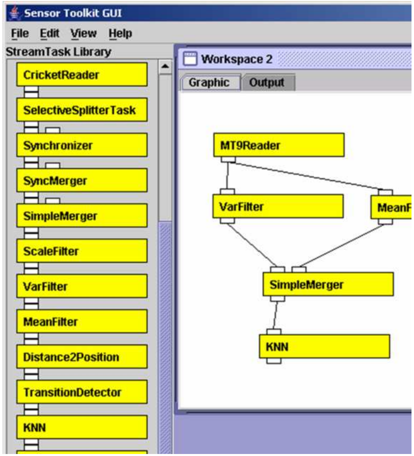


Image from D. Bannach, Uni Passau

CRN Matlab Interface

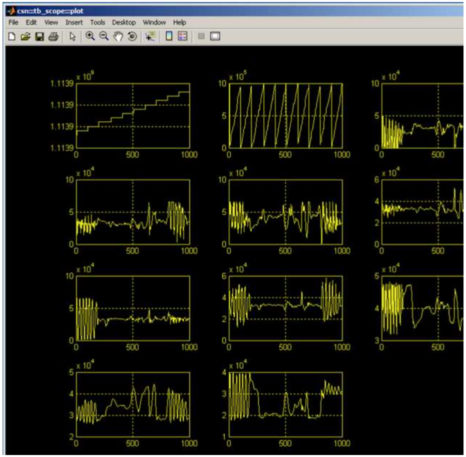


Image from D. Bannach, Uni Passau

HCI Lecture Summary

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

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.

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

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

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)

ConcurTaskTrees

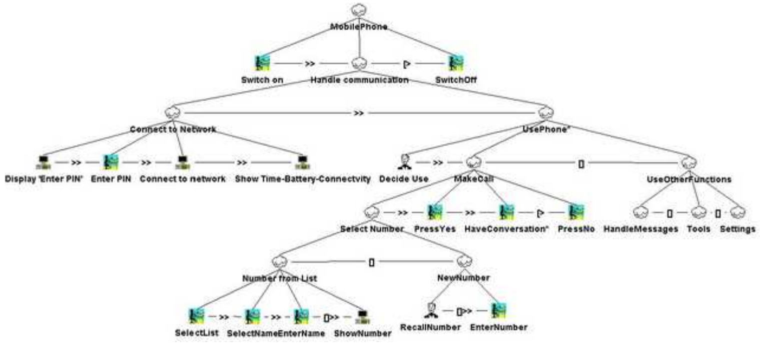
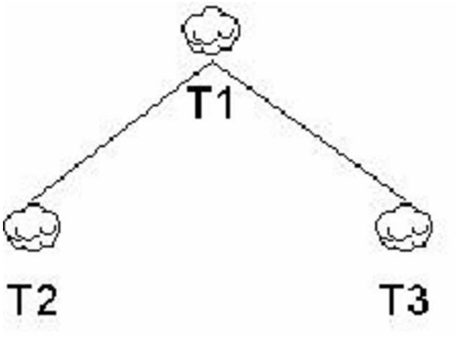


Image from Paterno, 1999

CTT: Features

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

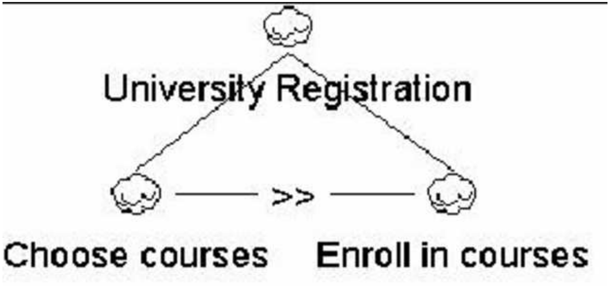
CTT: Temporal Operators



- Hierarchy

Image from Paterno, 1999

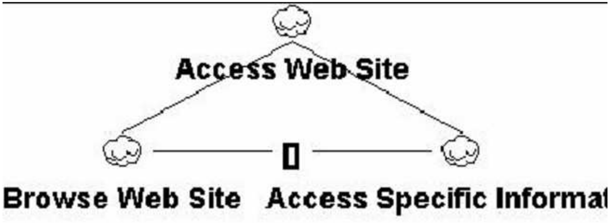
CTT: Temporal Operators



- Enabling

Image from Paterno, 1999

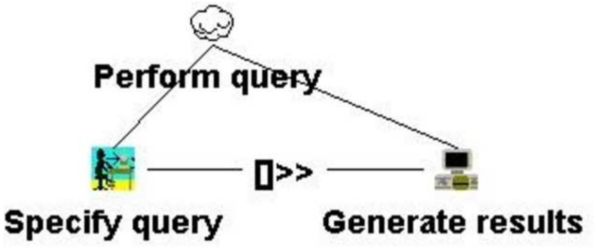
CTT: Temporal Operators



- Choice

Image from Paterno, 1999

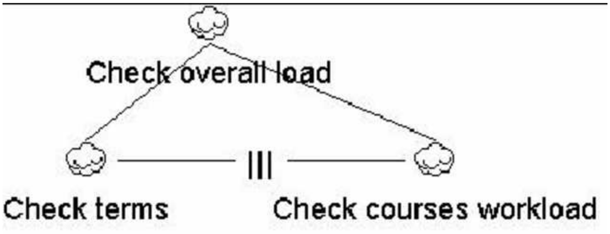
CTT: Temporal Operators



- Enabling with information passing

Image from Paterno, 1999

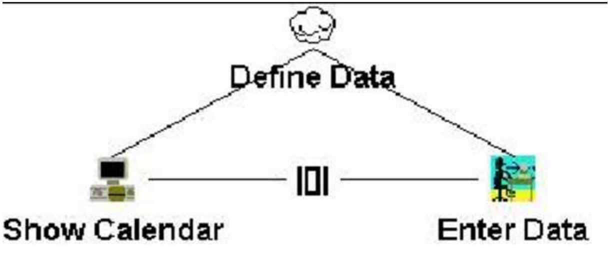
CTT: Temporal Operators



- Concurrent Tasks

Image from Paterno, 1999

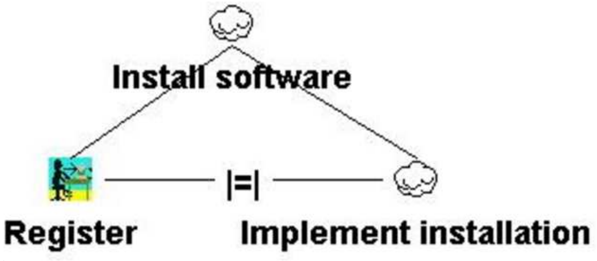
CTT: Temporal Operators



- Concurrent Communicating Tasks

Image from Paterno, 1999

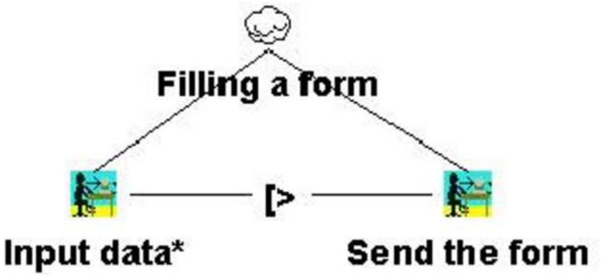
CTT: Temporal Operators



- Task Independence

Image from Paterno, 1999

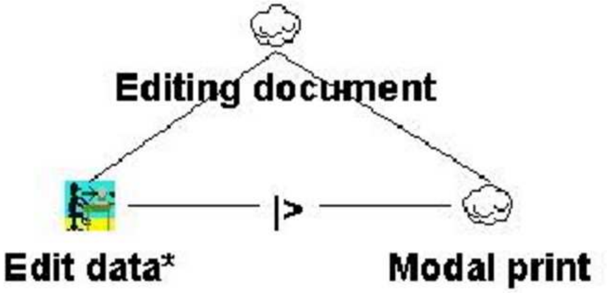
CTT: Temporal Operators



- Disabling

Image from Paterno, 1999

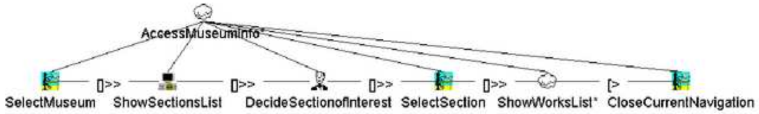
CTT: Temporal Operators



- Suspend-Resume

Image from Paterno, 1999

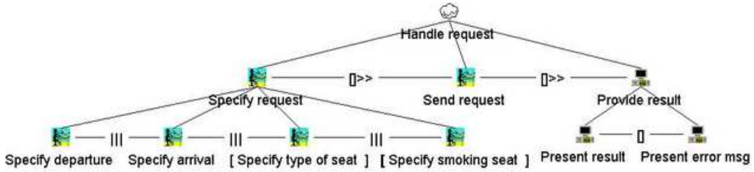
CTT: iterative task



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

Image from Paterno, 1999

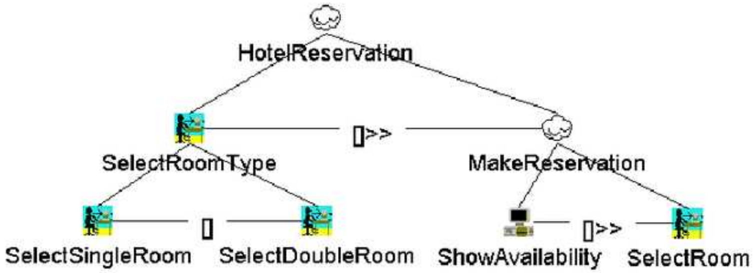
CTT: optional tasks



- Optional Tasks are marked with [and] brackets

Image from Paterno, 1999

CTT: inheritance of temporal constraint



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

Image from Paterno, 1999

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?

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

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

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

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);
```

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);
```

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);
```

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

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.

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 );
```

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?

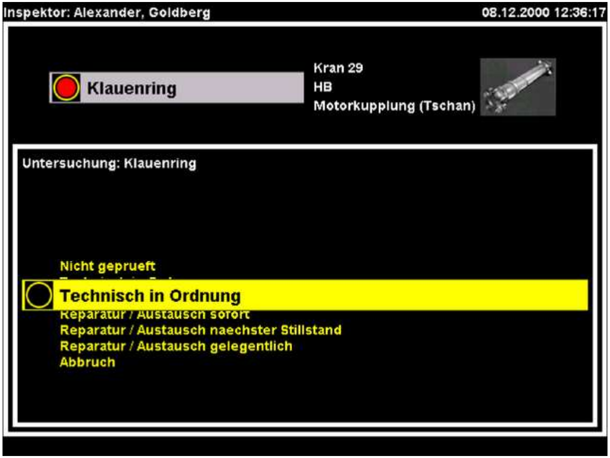
Wearable UI Methaphor

- 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

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

Winspect UI HMD



Inspektor: Alexander, Goldberg 08.12.2000 12:36:17

Klauenring

Kran 29
HB
Motorkupplung (Tschan)

Untersuchung: Klauenring

Nicht geprueft
Technisch in Ordnung
Reparatur / Austausch sofort
Reparatur / Austausch naechster Stillstand
Reparatur / Austausch gelegentlich
Abbruch

Image from T. Nicolai

Winspect Direct Manipulation

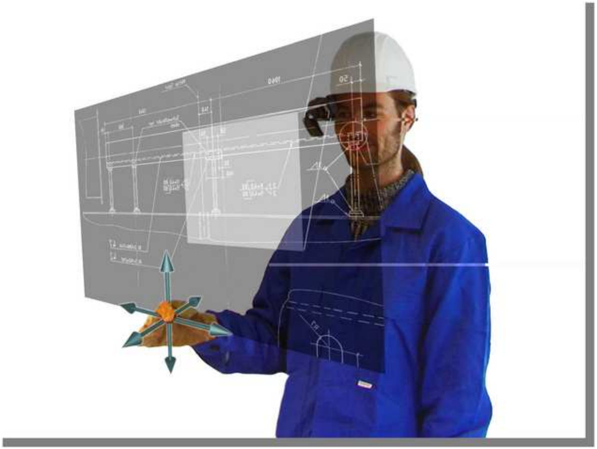


Image from T. Nicolai

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

Wearable UI Gesture

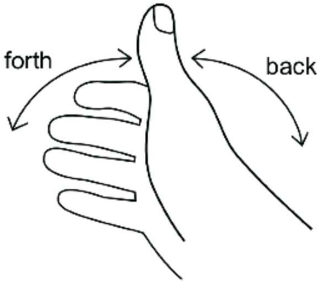


Image from H. Witt

Wearable UI Glove

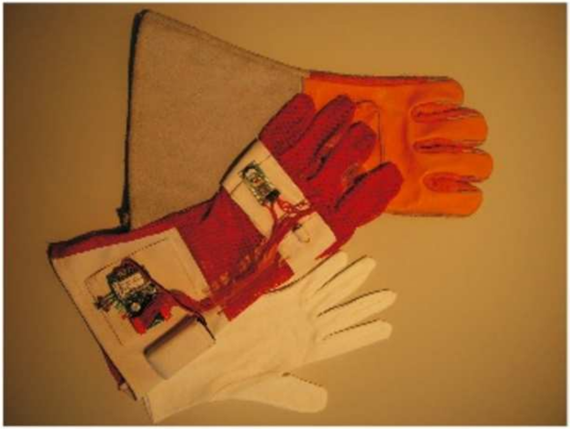
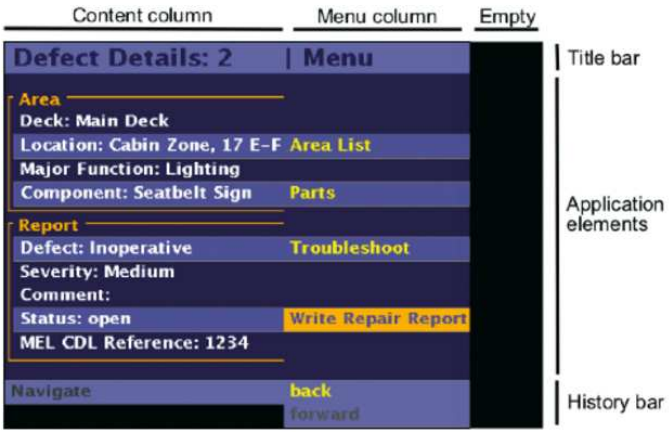


Image from H. Witt

Wearable UI HMD



Defect Details: 2 | Menu

Area

Deck: Main Deck

Location: Cabin Zone, 17 E-F Area List

Major Function: Lighting

Component: Seatbelt Sign Parts

Report

Defect: Inoperative Troubleshoot

Severity: Medium

Comment:

Status: open Write Repair Report

MEL CDL Reference: 1234

Navigate back

forward

Image from H. Witt

WUI-Development

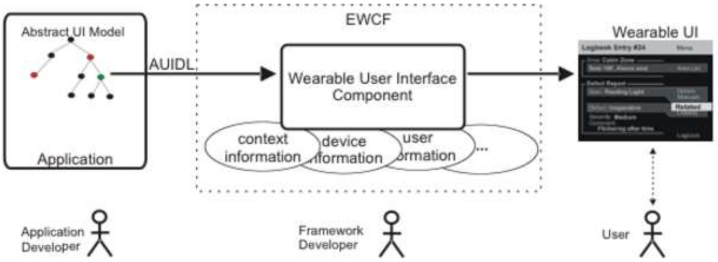


Image from H. Witt

WUI-Structure

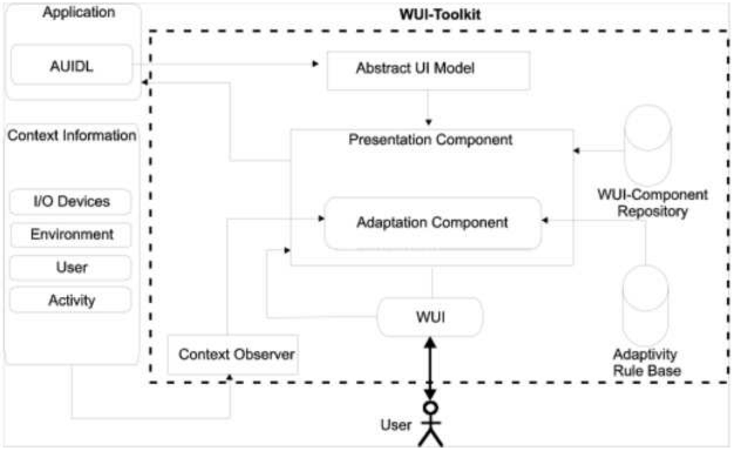


Image from H. Witt

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)

Layers of adaptation

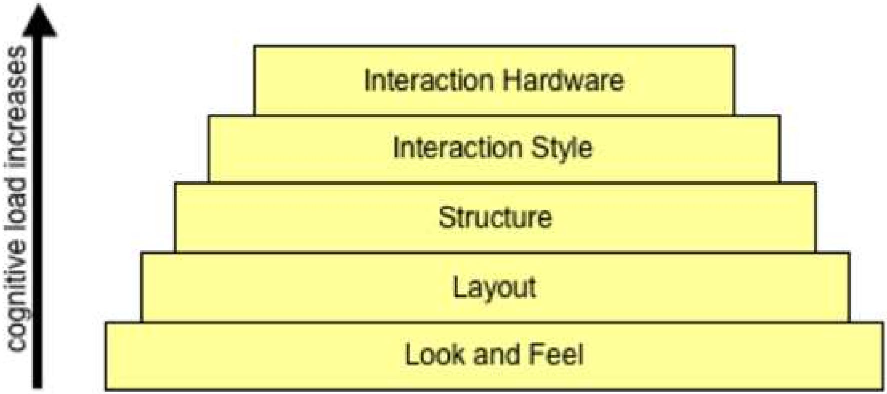


Image from H. Witt

Finding adaption rules

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

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

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

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

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

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

Matchingtask

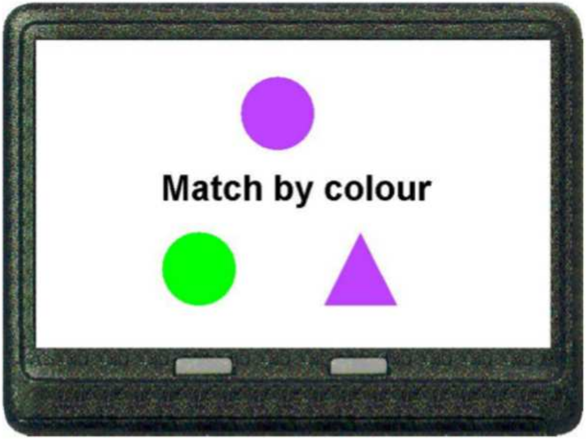


Image from H. Witt

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

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

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

Hotwire

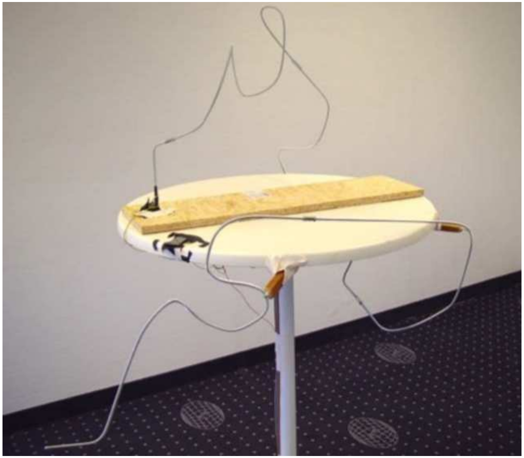


Image from H. Witt

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

Hotwire-Task

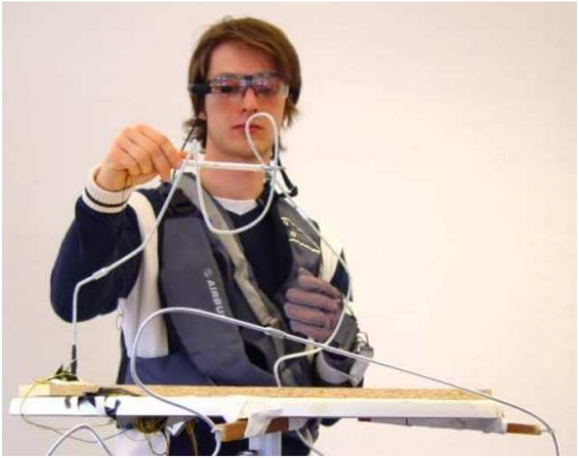


Image from H. Witt

Measuring Hotwire performance

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

Results

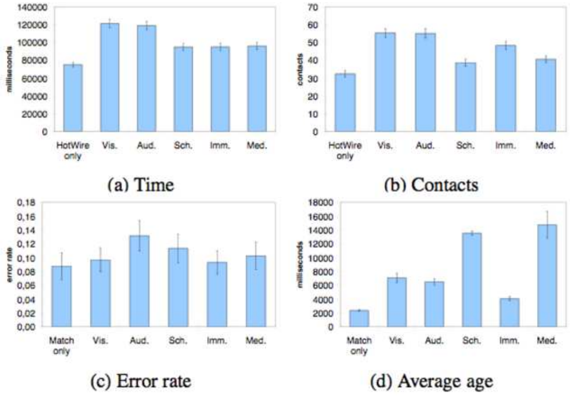


Figure 5. Averages of user performance.

Image from M. Drugge, H. Witt, ISWC06

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

Larger Hotwire (on CeBit)

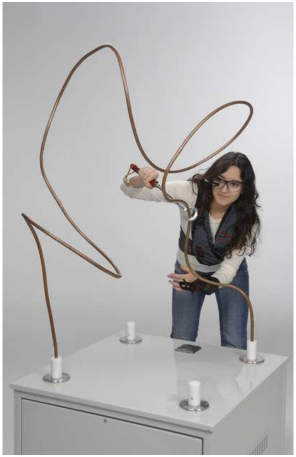


Image from mrc