Zusammenfassung People-Oriented Computing

Inhalt

[1. Paradigms of Interaction 2](#_Toc20144622)

[1.1 What is People-Oriented Computing? 2](#_Toc20144623)

[1.2 Interaction and Interaction Paradigms 2](#_Toc20144624)

[1.2.1 Communication as Interaction 2](#_Toc20144625)

[1.3 History of Human & Computing 2](#_Toc20144626)

[1.3.1 Time sharing (1950-1960) 2](#_Toc20144627)

[1.3.2 Video display units 3](#_Toc20144628)

[1.3.3 Programming Toolkits (1960s) 3](#_Toc20144629)

[1.3.4 Personal Computing (1970s-1980s) 4](#_Toc20144630)

[1.3.5 Windows and WIMP (1980s) 4](#_Toc20144631)

[1.3.6 Interface Metaphors 5](#_Toc20144632)

[1.3.7 Direct manipulation (1980s) 5](#_Toc20144633)

[1.3.8 Hypertext (1940s-1960s) 6](#_Toc20144634)

[1.3.9 The World Wide Web (1990s) 6](#_Toc20144635)

[1.3.10 Agent-Based Interfaces 6](#_Toc20144636)

[1.3.11 Multi-Modality 7](#_Toc20144637)

[1.3.12 Ubiquitous Computing (1990s-2000s) 7](#_Toc20144638)

[1.3.13 Sensor-based and Context-aware Interaction 7](#_Toc20144639)

[1.3.14 Augmented/ Virtual Reality 7](#_Toc20144640)

[2. Humans and interactive Systems 8](#_Toc20144641)

[2.1 An Interaction Framework 8](#_Toc20144642)

# 1. Paradigms of Interaction

## 1.1 What is People-Oriented Computing?

People-Oriented Computing provides a sampling of the many areas in which computing affects people, on

* An individual level
* A group or organizational level
* A societal level

## 1.2 Interaction and Interaction Paradigms

### 1.2.1 Communication as Interaction

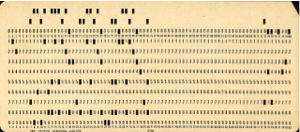
Communication includes means, abilities and channels of expressing intent, state or information and receiving input. It is necessary that there is enough shared understanding to interpret what is being communicated and to respond appropriately.

With humans, these abilities develop naturally, and certain structures have evolved over a long time. With computers these interactions are not natural. This means that it is necessary that computers can receive input so that it knows what it should do and must give the user feedback.

**Interaction Paradigms**: Successful approaches to interactive systems that have helped make it easier to use technology.

## 1.3 History of Human & Computing

### 1.3.1 Time sharing (1950-1960)



Previous approach **Batch sessions**: Individual programmers submitted jobs on punched cards or paper tape to an operator who then ran the individual jobs on a computer. Hardware advances in the 1940s and 1950s led to a massive increase in computing power through integrated chips. These hardware improvements necessitated parallel advancements in how to harness this power in use.

**Time sharing** was significant as a single computer could support multiple users at once and programming became an interactive activity. It gave rise to the “hacker” who could create increasingly complex programs. Time sharing shifted programming as a preplanned set of instructions for a computer to an exchange between programmer and computer.

### 1.3.2 Video display units

The first research in video display came in the 1950s for displaying images for military purposes.

In 1962 Ivan Sutherland landed a breakthrough with **Sketchpad**. It allowed data to be represented visually, abstracted, manipulated and changed. It enabled truly visual interaction and a more human way of interacting with data. It adapted the computer to the human’s way of thinking rather than the other way around.



### 1.3.3 Programming Toolkits (1960s)

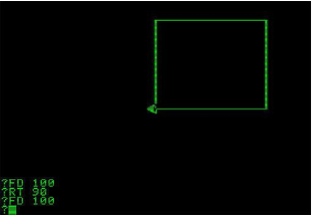
Up until this time Computers were considered something only experts and specialists could use. Douglas Engelbart’s vision was to enable humans to use computers to learn. He created programming tools that allow people to create complex programs more easily.

This allowed for bootstrapping to be possible.

**Bootstrapping**: Small programming components can be combined to create larger ones

### 1.3.4 Personal Computing (1970s-1980s)

The notion of computing for the masses without the need for substantial computing skills in order to benefit from computers became popular. Seymour Papert created a programming language for children called **LOGO** which demonstrated that powerful tools for hackers could be used by novices. It made use of a graphical “turtle” that could be commanded to draw shapes through simple English-based phrases (e.g. “turn left”). It illustrated that ease of use makes a system more powerful.



Alan Kay believed the future of computing was small, powerful machines dedicated to single users: **personal computers**. He believed in a shift away from mainframe computing and timesharing. With his team he created **Smalltalk**, a simple, but powerful, visually based programming environment especially for personal computing.

Kay also conceived of the **Dynabook** in the 1970s, a handheld personal computer for children.

### 1.3.5 Windows and WIMP (1980s)

The beginning of personal computing led to a focus on increased usability of single-user interaction with computers. Previous interfaces were command-line based. There was increased support for engaging in multiple tasks at once, with humans in control. Supporting multiple threads of interaction in conventional command line interfaces became complicated and difficult to manage. **Window-based** systems supported physical and logical separations of tasks.

The Xerox Star (1981) computer introduced the first commercial WIMP interface.

**WIMP**: Interface based on Windows, Icons, Menus and Pointers.

### 1.3.6 Interface Metaphors

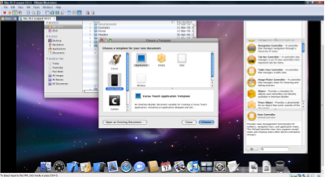
Metaphors helped people learn new concepts by putting them in terms of known concepts. Metaphors applied to computer interactions make unfamiliar concepts familiar and reduce the perception of complexity or difficulty.

Window and WIMP interfaces make extensive use of real-world metaphors:

* Windows
* Buttons
* Menus
* Palettes

Xerox Star and successors made use of an office desk metaphor:

* Desktop
* Folders
* Trash can
* …



Metaphors are naturally limited as it is not possible to completely map one set on concepts onto another. Because of this mismatches and false expectations can occur (i.e. Folders within folders, dragging media into trash eject).

### 1.3.7 Direct manipulation (1980s)

Traditional command line interfaces provided very limited feedback in interactions. Advancement in displays allowed for rapid audio and visual feedback with every interaction.

Rapid feedback facilitated an interaction technique called **direct manipulation**. It creates the illusion of operating directly on data and objects, rather than giving commands to a computer.

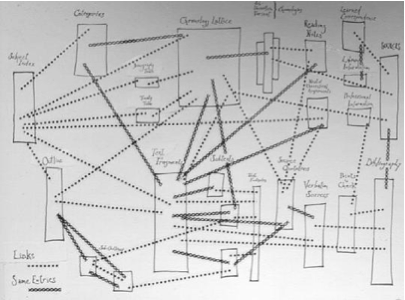
Features of direct manipulation are:

* Visibility of all objects of interest
* Incremental action at the interface with rapid feedback on all actions
* Reversibility of all actions so that users can explore without severe penalties
* Syntactic correctness of all actions so that possible action is a legal operation
* Replacement of complex command languages with actions

The first commercial success of a direct manipulation interface was the Apple Macintosh computer (1984).

It made files and directory structure visible to the user. Operations such as moving files between directories were mirrored in an action on a visible document that could be picked up and dragged. It was impossible to formulate a syntactically incorrect command and it gave continual visual feedback while the operation was being carried out.

### 1.3.8 Hypertext (1940s-1960s)



### 1.3.9 The World Wide Web (1990s)

### 1.3.10 Agent-Based Interfaces

### 1.3.11 Multi-Modality

### 1.3.12 Ubiquitous Computing (1990s-2000s)

### 1.3.13 Sensor-based and Context-aware Interaction

### 1.3.14 Augmented/ Virtual Reality

**Augmented reality** combines physical world and digital content. It requires knowledge of the environment like QR codes or IR sensors.

**Virtual reality** replaces the physical world with a digital world. It’s a full immersion with 3D interaction. It uses gesture recognition, eye gaze and/or full body sensing.

Both have many applications: entertainment, medicine or training.

# 2. Humans and interactive Systems

The purpose of an interactive system is traditionally to aid a user in accomplishing a goal within an application domain.

## 2.1 An Interaction Framework

An interaction Framework has four major components:

* The systems
* The user
* The input
* The output