

# Paradigms

## 1. Time Sharing

- Allowed multiple users to access one computer simultaneously.
  - Shifted from batch processing to interactive computing
  - Enabled real-time feedback for programmers.
  - Marked the beginning of true human–computer interaction.
  - Popularized the concept of the computer as a “partner.”
- 

## 2. Video Display Units (VDUs)

- Replaced printed output with on-screen visual output.
  - Enabled graphical interaction instead of text-only.
  - Ivan Sutherland’s Sketchpad (1962) showed direct visual manipulation.
  - Provided immediate feedback and new forms of visualization.
  - Foundation for modern GUIs.
- 

## 3. Programming Toolkits

- Initiated by Douglas Engelbart’s Augmentation Research Center.
  - Concept of bootstrapping: build small tools → combine to create larger systems.
  - Tools (mouse, word processing concepts) later became standard UI elements.
  - Increased developer productivity and interaction innovation.
- 

## 4. Personal Computing

- Alan Kay’s vision: computers should be accessible to everyone, not just experts.
  - LOGO language showed even children could interact with computers.
  - Emergence of powerful, small, single-user machines.
  - Led to modern PCs, tablets, laptops.
- 

## 5. WIMP Interface (Windows, Icons, Menus, Pointer)

- First commercialized by Xerox Star (1981), later by Apple Macintosh.
  - Supports multitasking via multiple windows.
  - Icons and menus simplify command recall.
  - Pointer (mouse) enables fast selection.
  - Dominant desktop interaction style for decades.
- 

## 6. Metaphor

- Interfaces borrow from real-world concepts for familiarity.
    - Example: Desktop → files, folders, trash bin.
  - Useful for beginners, helps reduce learning curve.
  - But metaphors can break (e.g., dragging floppy disk to trash to eject).
  - Cultural bias can limit metaphor effectiveness globally.
- 

## 7. Direct Manipulation

- Term coined by Ben Shneiderman.
  - Key features:
    - Visibility of objects
    - Immediate feedback
    - Reversible actions (Undo)
    - Only valid actions allowed
    - “Physical” interaction with digital objects (drag, drop)
  - Popularized by Macintosh and modern GUIs.
  - Strength: Learnability and low error rate.
- 

## 8. Language vs Action Paradigm

- Action: direct manipulation (drag, click, move).
  - Language: commands, scripts, queries (SQL, terminal).
  - Action = simpler tasks, visually intuitive.
  - Language = complex, repetitive tasks easier to express.
  - Modern systems mix both (e.g., macros, programming by demonstration).
- 

## 9. Hypertext

- Introduced by Ted Nelson; inspired by Vannevar Bush's memex.
  - Non-linear documents linked via associations.
  - Basis for modern web navigation.
  - Supports flexible access to information instead of linear reading.
- 

## **10. Multi-modality**

- Uses multiple human communication channels together:
    - Speech + gesture
    - Touch + vision
    - Pen + voice
  - Allows more natural interaction.
  - Reduces dependence on keyboard/mouse.
  - Key technologies: speech recognition, gesture tracking, haptics.
- 

## **11. CSCW (Computer-Supported Cooperative Work)**

- Supports collaboration between multiple users.
  - Two types:
    - Synchronous: real-time (video calls, shared whiteboards)
    - Asynchronous: time-independent (email, forums)
  - Led to groupware systems like Slack, Google Docs.
  - Emphasizes social and group dynamics in interaction design.
- 

## **12. World Wide Web**

- Invented by Tim Berners-Lee (1989).
  - Hypertext + internet = global information system.
  - Mosaic browser (1993) made the web graphical.
  - Simple technologies (HTML, URLs, HTTP) enabled mass adoption.
  - Became the dominant platform for information, commerce, communication.
- 

## **13. Agent-Based Interfaces**

- Software acts autonomously on user's behalf.

- Examples:
    - Email filters
    - Recommender systems
    - Web crawlers
    - Spreadsheet auto-sum
  - Agents may be embodied (icons) or invisible.
  - Key challenge: letting users express intent clearly and safely.
- 

## 14. Ubiquitous Computing

- Concept by Mark Weiser, Xerox PARC.
  - Computers become part of everyday environment; “invisible computing.”
  - Device scales:
    - Yard (large displays, murals)
    - Foot (tablets, PCs)
    - Inch (phones, wearables)
  - Goal: computing everywhere without demanding attention.
- 

## 15. Sensor-Based & Context-Aware Computing

- Systems sense environment or user context and respond automatically:
  - Motion-activated lights
  - Auto-brightness in phones
  - Smart thermostats
  - Recommendation engines
- Uses sensors: GPS, cameras, accelerometers, biometrics.
- Must follow “appropriate intelligence”:
  1. Be right often.
  2. Cause minimal harm when wrong.