In our overview, we have seen two approaches involving cognitive models of user knowledge versus the lively dynamic of actual usage (8 Interaction Styles and the 3 General Principles).

In this field, it is now generally acknowledged in strict cognitive psychology as applied to HCI studies, that design cannot be separated from patterns of use. This is because when using a particular CHI (Computer-Human Interface) and having difficulties, users ask for help from neighbours, experts, alternate sources of knowledge, etc., resulting in physical and social environments being inextricably intertwined with computer use and information technologies. Such awareness now terms this approach as situated action and distributed cognition.

This next overview is one that tries to combine both approaches and is called the Object-Action Interface Model (OAI, pronounced as Oo-Ah).

Object-Action Interface Model -- OAI

As we have been taught, we learn to piecewise refine and thereby decompose to determine a solution.

OAI does exactly this using task hierarchies of objects and actions as shown below.
Object-Action Interface Model -- OAI

- **OAI Model Example:**
  - Consider a user who needs to write a business letter using word processing software on a computer
    - High-level task action = concept of writing a letter
    - High-level task = letter production
    - High-level task object = letter
    - Interface object = letter will be stored as an electronic document
    - Interface action = know the details of using the <Save> command
    - Low-level task object = using spelling characters to form words/sentences, etc.
    - Low-level interface object = know where the keys on the keyboard are for each letter.
  - The goal of minimizing interface concepts (e.g. syntax of command language), while presenting a visual representation of the task objects and actions is the heart of the direct manipulation approach to design.

Object Orientation and OAI

- **OAI Model & OO:**
  - Note that
    - High-level design with a visual programming language is helpful in using OAI, where **widgets** are used as part of user-interface-building tools that have familiar and simple syntax (click, drag & drop, etc.) and simple forms of feedback (highlighting, scrolling, etc.)
    - The manner in which the OAI is cast allows implementations that encompass the **object-oriented programming methods** and which are part of modern software engineering trends.
    - In fact, the syntax of most building blocks of CHI is rapidly disappearing as the graphic widgets library becomes more universal. Carried to the extreme, if application syntax AND behaviour (methods) can be completely well-defined by widgets then this would allow on-line modification using no lines of programmable code or the **zero lines of code** capability.
    - [Reminds one of SE & CS programmers becoming obsolete!]
Visual Environments and OOPS

• Remember the Object Oriented Paradigm (as applied to GUIs)
  – OO Summary:
    • **Encapsulation** hides or makes private the object's *implementation* details (internal data structure and methods), while publishing its *interface* or those abstractions necessary to use the object.
    • **Abstraction** extracts the properties (essential to write software) from the properties that are irrelevant for that purpose.
    • **Inheritance** enables developers to derive classes from existing classes (*superclass/subclass* hierarchies) in order to instantiate objects from these classes that perform some activity.
    • **Polymorphism** means that an object will respond to a message appropriate to that message; therefore an object is a self-contained entity that communicates with another object via its *public interface* and required no knowledge of what the other object is nor what it will do with the message.

The MVC(Model View Controller) Concept

• Introduction
  – Some of us have seen this model applied to software application programming, which helps to show that a good program should separate the modules involving Inputs, Processing and Outputs.
  – This is the first step in piecewise refinement, which is
    
    Input $\rightarrow$ Processing $\rightarrow$ Output
    OR
    Controller $\rightarrow$ Model $\rightarrow$ View

• HCI Connection
  – For us
    • The Input can be the (visual) User Input
    • The Model can be the HCI program, and
    • The View can be the (visual) Feedback to the User
The MVC Concept -- 2

- **MVC and HCI**
  - **The Controller (Input)**
    - Accepts input from the user via mouse, keyboard, etc.
    - Maps these actions into commands that are sent to the Model
    - May to interact with the Viewport also
  - **The Model (Processing)**
    - Encapsulates data and methods
    - Serves as abstraction of real-world process
    - Captures how the system works
    - Keeps track of the system state as changes are made
  - **The View or Viewport (Output)**
    - Responsible for mapping graphics to a device
    - Remaps the display when model changes state if necessary
    - Can be multiple viewports each of which answers to the Model

- **An Example (at upper right)**
  - Consider using PowerPoint (or Impress) with a portable computer and an attached display (or projector). In this case
    - The Model is the PowerPoint executing on the computer in Windows OS.
    - The Controller is the display input screen open for mouse clicks and keyboard commands
    - The Viewport (main display) shows the workspace usually the opening screen of the application
    - BUT, the Controller allows the extra display to be set up and send a different display to the remote screen, as the presentation itself. That is the Controller can tell the Model to set up another remote screen and provide another display section to allow its control as opposed to the normal display that controls the usual or main display