

SOS

**A Simple Operating System
for modeling HCI with
ACT-R**

Rapid Prototyping

- Used early in the design process for computer interfaces
- System is often not well developed, most functions not implemented
- Rapid, iterative testing of different designs
- Testing done on low fidelity mock ups with limited functionality
- Testing is usually done on 5 to 7 subjects

Problems

- Testing 5 to 7 users is not enough
 - Individual differences
 - Interaction permutations
- Hard to find motivated subjects
- Problems with reusing subjects
- No theory development
- No way to check the validity of the results

- Solution
 - Augment the process by testing simulated users created with ACT-R

Advantages

- Virtually unlimited n
- Fully motivated subjects
- Subjects unaffected by mockup quality
- Can test longer term learning
- Can model experts
- Can test permutations

- Disadvantages
 - Hard to evaluate visual/attentional characteristics, e.g., how intuitive is an icon, will something be noticed??
- Solution
 - use real subjects to do this

Interface prototyping

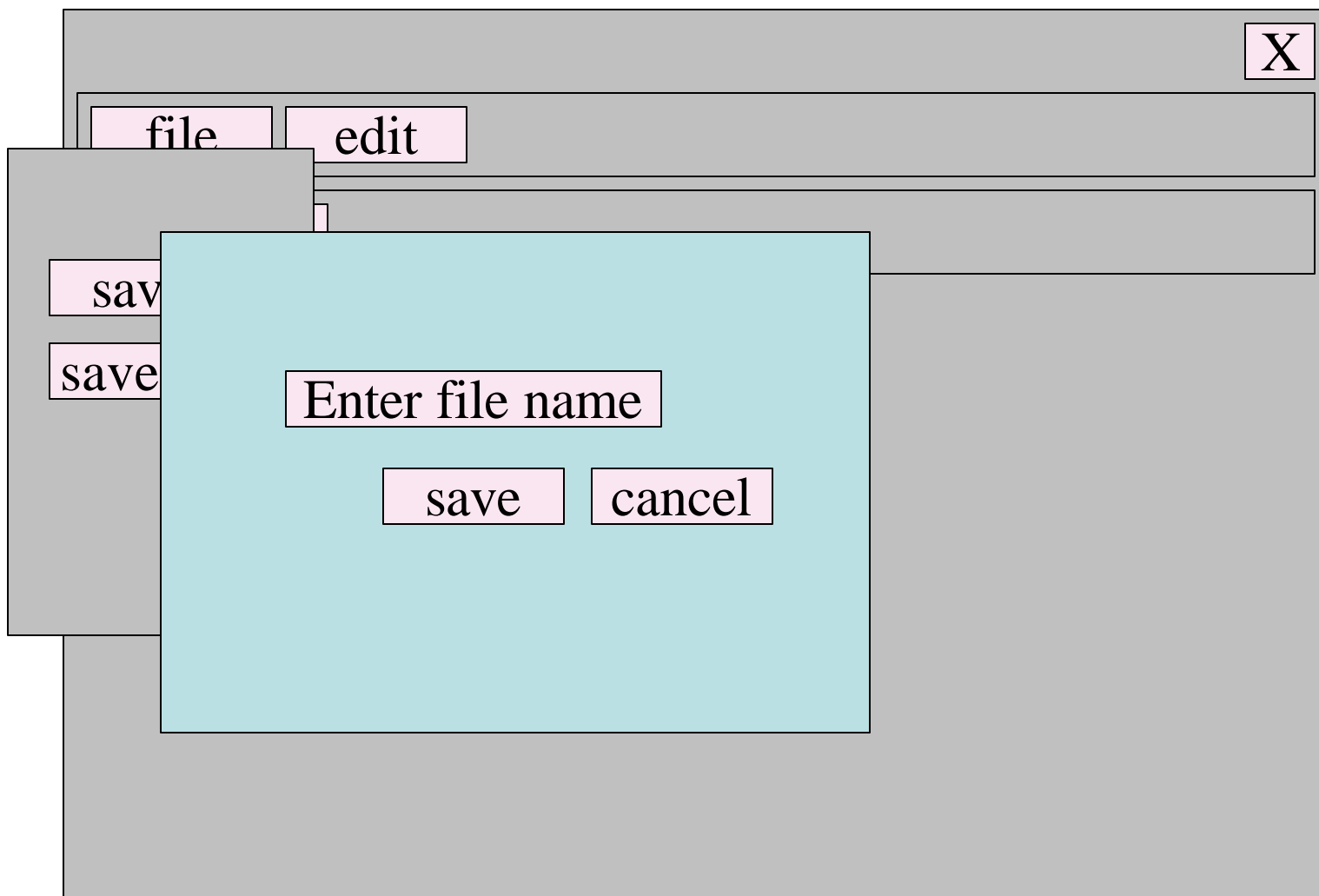
- Characteristics of rapid prototype testing
 - Task is usually not time pressured
 - Errors are not catastrophic, easy to correct
 - Visual elements are usually static, fairly obvious, and often familiar
 - Only limited functionality has been specified
 - Prototyping and testing is time pressured
- Therefore, interface prototyping tends to focus on easy to create, low fidelity mockups

SOS

- Stands for Simple Operating System
- SOS breaks all interfaces down into *containers* and *objects*.
 - Containers contain objects and other containers and can be opened or closed
 - Objects provide information (e.g., labels, text) and can trigger actions (e.g., a button can trigger the opening of a container)
- Designed to be easy to use for ACT-R programmers
- We argue that most Windows based software programs can be mocked up using this simple approach







Problem space

- More generally, SOS is a type of problem space representation
- The current node is represented by the contents of the available containers
- Operators are represented by the objects that trigger actions (i.e., change the available containers)
- Objects are simply triggered, but they're treated differently by the simulated user based on the way they're labeled

Finding and activating objects

- Objects and containers have locations
- The location is needed to trigger it
- The simulated user can search:
 - A specific location
 - Within specific containers
 - Across the whole screen
- If the object is found a chunk with it's location is placed in the visual buffer
- If it is not found a failure chunk is placed in the visual buffer
- Average times are used for finding and activating

Easy to use

- (sos:define-object-type button-holder (sos-container))
- (sos:define-object-type button (sos-object) type)
- (sos:add-sos
- (menu-bar
 - isa-sos button-holder
 - location loc1
 - available t
 - is-part-of program-window
 - has-parts
 - (file-menu-open))
- (file-menu-open
 - isa-sos button
 - type file-menu-open
 - location loc6
 - available t
 - is-part-of file-menu-bar
 - ;; is-part-of icon-menu-bar ;; not in the expected place
 - actions ((sos::open-container file-menu))

SOS relationship to GOMS

- Engineering/modeling approach applied to learning a new interface
- Idea that using average times will provide a *reasonably* accurate model of average behavior (i.e., identify major problems, identify most problems)
- Focus on top down, knowledge driven behavior (i.e., as opposed to situated action, affordances, etc.)

Simulated user relationship to GOMS

- Usability testing conceived of in GOMS terms
 - Instructions = unit tasks
 - Unit tasks completed by applying methods
 - Methods made up of operators (looking and activating)
- Attempts to build a GOMS model of the task in declarative memory by adapting and adjusting default methods
- Strategy and error correction based on production system - i.e., an expert system for exploring

Extending SOS

- Vision
 - Bottom up attention
 - Object salience
 - Hands
 - Object based attention index
 - Agents

Conclusions so far

- Exploring novel interfaces is a very interesting and challenging domain
- Ultimate goal – to model the process from exploring to expert user
- Building a working model tells you a lot about an interface design
- SOS is also useful in other domains - objects and containers are not limited to interfaces
- Large software companies should invest large amounts of money into ACT-R modeling