Exploring the usability of adaptive menus with a simple object system

Bruno Emond,
Institute for Information technology, Computational Video Group,
National Research Council Canada.

Robert L. West,
Department of Psychology, and Department of Cognitive Science,
Carleton University.
Overview

• The ACT-R simulation tool space.

• Simple Object System - ACT-R/SOS.

• Modelling user interactions with adaptive menus.
  – Can we make design decisions based on ACT-R simulations?
The ACT-R simulation tool space

• Interaction with external applications or environment

• High fidelity simulated task environments
  – ACT-R/PM.

• Low fidelity simulated task environments
  – ACT-R/SOS.
Simple Object System - ACT-R/SOS

• What is it?
  – Work in progress
  – Tool to build low fidelity simulated environments to run against ACT-R cognitive models.
  – Focused on “What”, not “Where” are external objects.
  – Definition of plus-rhs buffer functions:
    • Perception module: modification of parameters (object selection method, and cost method).
    • Action module: support for calling motor actions defined in a model (action-cost, and object-response-time).
  – Definition of object classes, methods, and motor action methods:
    • Inheritance, class application and chunk slots.
    • Object methods for motor buffers.
Simple Object System - ACT-R/SOS

• Why bother?
  – Cognitive modelling and simulation development through successive refinements.
  – Make explicit, in the model, the mechanisms of perception and motor action.
  – Make explicit, in the model, the external objects behaviour.
  – Tool to learn ACT-R.
  – Link to ACT-R/PM as a device plugin.

(yet-another-task23
  isa to-do-list-item
  list to-do-list45
  description sos-as-an-ACT_R/PM-device-plugin)
Simple Object System - ACT-R/SOS

• Who would be interested anyway?

  – People who want to learn ACT-R.
  
  – People who want to explore buffer computational properties.
  
  – People who want to generate some hypothesis based on simulation results.
  
  – People who want to use simulated users for usability testing.
  
  – People on the rush, they just want to get going.
Model structure

• Class and method definitions

• Class instances

• Buffer definitions

• ACT-R model
Classes and method definitions (WYSIWYG)

(define-sos-object-class target-list
  :inherit-from (interface-object)
  :application-slots (current-targets sos-menu)
  :chunk-slots (current-target-name))

(defun set-target ((target-list target-list))
  (let ((target (car (current-targets target-list))))
    (setf (current-targets target-list)
          (cdr (current-targets target-list))
          target))

(define-sos-object-action-method get-target ((target-list target-list))
  :action-cost #'(lambda () 0.05)
  :sos-object-response-time #'(lambda () (system-busy-meter))
  (setf (current-target-name target-list)
        (set-target target-list)))

The Chunk type for sos-object-class TARGET-LIST is:
  (CHUNK-TYPE TARGET-LIST CURRENT-TARGET-NAME)

The Chunk type for sos-object-class MOTOR-ACTION is:
  (CHUNK-TYPE MOTOR-ACTION TARGET-OBJECT ACTION-METHOD)
(add-sos-objects
 (mt01 isa-sos-object target-list
  current-targets (t01 t02 t03 t01 t03 end)
  sos-menu sos-menu01)
 (sos-menu01 isa-sos-object sos-menu))
Definition of plus-rhs buffer functions

(defparameter *perceptual* nil)
(defparameter *motor* nil)

(define-plus-rhs-perception-function find-sos-object
  :selection-function #'(lambda (indx-obs sos-obs)
      (if indx-obs
          (nth (random (length indx-obs)) indx-obs)
          (nth (random (length sos-obs)) sos-obs))))
  :cost-function #'(lambda (indx-obs sos-obs)
      (declare (ignore indx-obs sos-obs)
        *default-action-time*)))

(define-plus-rhs-motor-function sos-action)

(define-buffer perceptual *perceptual* :plus-rhs find-sos-object)
(define-buffer motor *motor* :plus-rhs sos-action)
A production

(p get-target-menu
  =goal>
  isa goal
  step get-target-menu

  =perceptual>
  isa target-list

==>

=goal>
step look-at-target-menu

+motor>
isa motor-action
target-object =perceptual
action-method get-target

+perceptual>
isa target-list)
An example: Simulation of adaptive menus

• Motivation for the simulation
  – Are adaptive user interfaces usable?
  – Can ACT-R help us making design decisions?

• Adaptive menu options
  – Random: It says it all, never the same
  – Fixed: It says it all, always the same
  – Stacked: Last chosen goes on top, pushing down the rest
  – Frequency: Sorted based on frequency access
  – Activation: Sorted based on activation (frequency and time). *No model telepathy*
Distribution of menu items

• A simulated subject sees 10 successive sets of 30 targets in the four adaptive menu conditions (menu of size 12)
  – Random targets 1-30: ("t08" "t08" "t08" "t11" "t11" "t11" "t02" "t02" "t05")
  – Early targets 1-15: ("t07" "t07" "t07" "t07" "t10" "t10" "t01" "t01" "t04")
  – Late targets 16-30: ("t09" "t09" "t09" "t09" "t12" "t12" "t03" "t03" "t06")

• The model is reset for each menu condition.

• Parameters
  – Randomness and Base level learning (0.5).

• Productions.
  – get-new-target, retrieve-target-position (with success or failure), scan-menu-for-target (upward or downward), test-target-success, add-target-to-menu.
Simulation results

Average number of scans to find a target

Learning cycles (number of targets presented)
Conclusion

• Data collection on adaptive menu task

• ACT-R simulations for user interface design decision
Current and other work

• Usability testing with simulated users.
  – Robert West and COGNOS.

• Modelling media player usage in the context of music learning.
  – Reviewing music coaching session (ensemble).
  – MusicGrid: NRC, NAC, CRC, School boards.

• Modelling quality of experience judgments and person-person interaction.
  – Advanced collaborative environments: NRC, CRC, NewMic.
Thank you :) 

Bruno.Emond@nrc-cnrc.gc.ca