A Toolset for Supporting Iterative Human – Automation Interaction in Design

Michael Feary, PhD

NASA Ames Research Center

Michael.S.Feary@nasa.gov

ABSTRACT

The addition of automation has greatly extended humans' capability to accomplish tasks, including those that are difficult, complex and safety critical. The majority of Human - Automation Interaction (HAI) results in more efficient and safe operations, however certain unexpected automation behaviors, or "automation surprises" can be frustrating and, in certain safety critical operations (e.g. transportation, manufacturing control, medicine), may result in injuries or the loss of life. (Mellor, 1994; Leveson, 1995; FAA, 1995; BASI, 1998; Sheridan, 2002). This paper describes the development of a design tool that enables on the rapid development and evaluation of automation prototypes. The ultimate goal of the work is to provide a design platform upon which automation surprise vulnerability analyses can be integrated.

Introduction

Recent analyses of aircraft accidents (FAA, 1995; BASI, 1998) have shown that aircraft automation is increasing as the major contributing factor to aircraft incidents and accidents. These accidents have shown a disturbing trend in that the automation was performing as designed, and was operated by well-trained operators, but users were surprised with unexpected automation behavior.

These "automation surprise vulnerabilities" are due to a failure in the specification of the behavior of the automation, rather than a failure in the implementation of the automation. The vulnerabilities could be due to a number of possible factors, including: inadequate coverage of the possible situations the automation needs to be able to respond to, or a weakness in the presentation of the automation behavior, such that the human user and the automation do not share a common understanding of the goals, the situation, or the proper behavior to accomplish a goal for a given situation. In either case, the focus needs to be on presenting human operators with predictable automation behavior.

The ultimate goal of the research described in this paper is the development of a viable means of identifying Human-Automation Interaction (HAI) vulnerabilities early in the design process. The focus for these HAI analyses is on the "cognitive" behavioral aspects of the user and the software or digital hardware in computers. The analyses aim to identify vulnerabilities in the communication of behavioral expectations or intent between the user and the automation.

The Automation Design and Evaluation Prototyping Toolset (ADEPT) was developed to respond to this need, and to focus on the iterative specification of decision logic of the automation being designed. The tool is intended to produce an accurate and complete specification. In addition to the focus on specifying decision logic, the tool was intended to provide a platform for integrating HAI testing and analysis.

The focus of this paper is an examination of the suitability of ADEPT to serve as a platform to upon which to integrate HAI analyses. ADEPT was developed to be usable by a domain expert designer without requiring extensive programming language expertise. This requirement was intended to enable ADEPT to be used early in the design process, by many different design team members (e.g. training, procedure, interface, etc.) The tool should foster communication between design experts from different domains, meaning that the tool should provide a structure that provides specific transition points for design team members to interact with each other.

ADEPT combines a graphical user interface design capability with an automation behavior specification capability and an automatic code generator to enable domain expert designers to create testable software prototypes.

• The User Interface Editor enables the designer to specify the look and feel of the of user interface by placing graphic objects on a canvas. The graphic objects include buttons, knobs, displays, and the ability to import static and dynamic graphical objects created in other software applications. The

properties (i.e. font, size, color, etc) of the graphical objects in can be changed in a property browser for the User Interface Editor, or can be dragged into the Logic Editor to allow graphic properties to be changed dynamically corresponding to the automation behavior.

• The Logic Editor enables the designer to specify the decision logic and automation behavior of the device, the environment in which the device operates, as well as the behavior of the user interface objects on the user-interface corresponding to the reflect the current state of the device and environment.

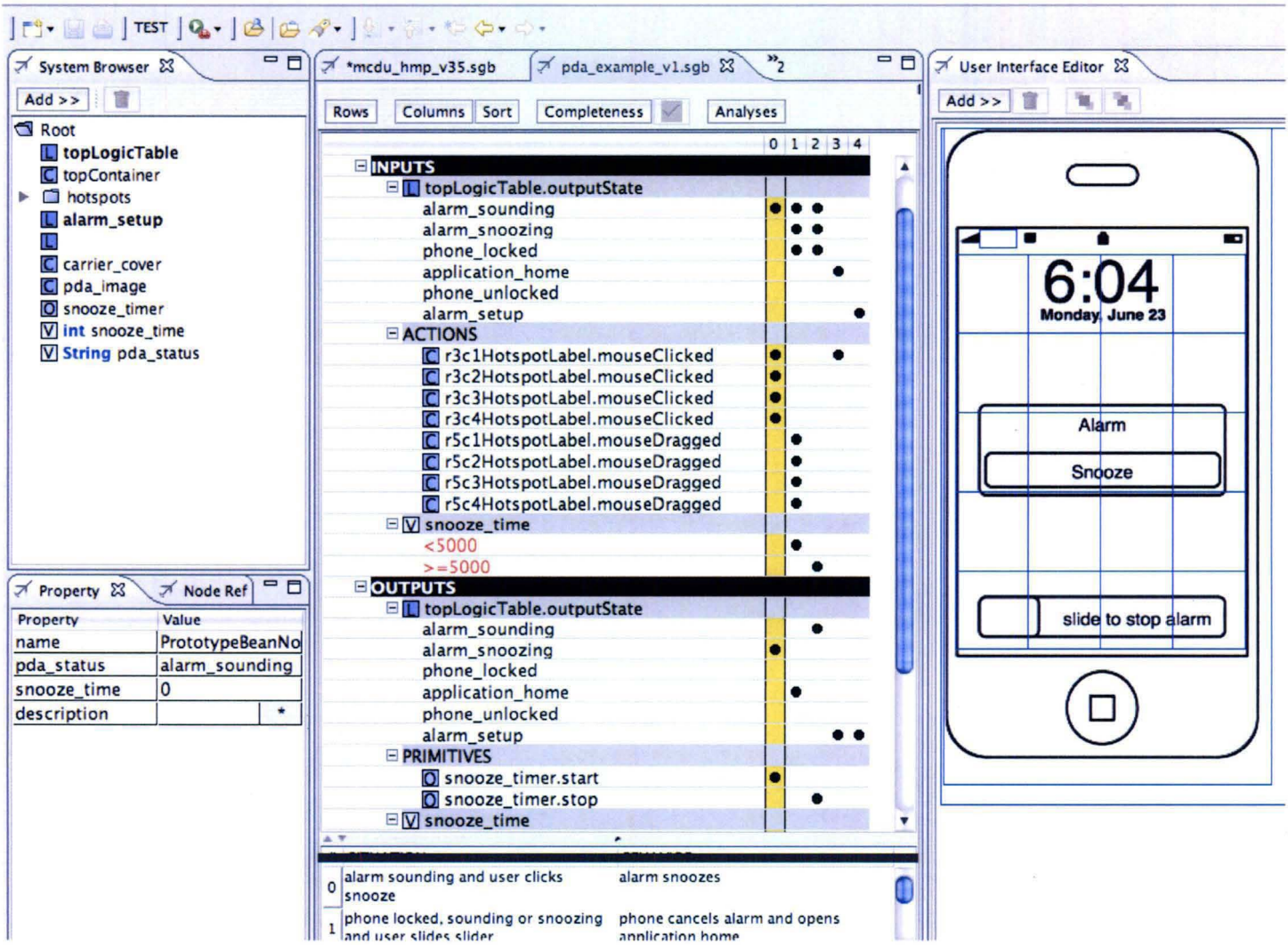

Figure 1. ADEPT in *Build* mode

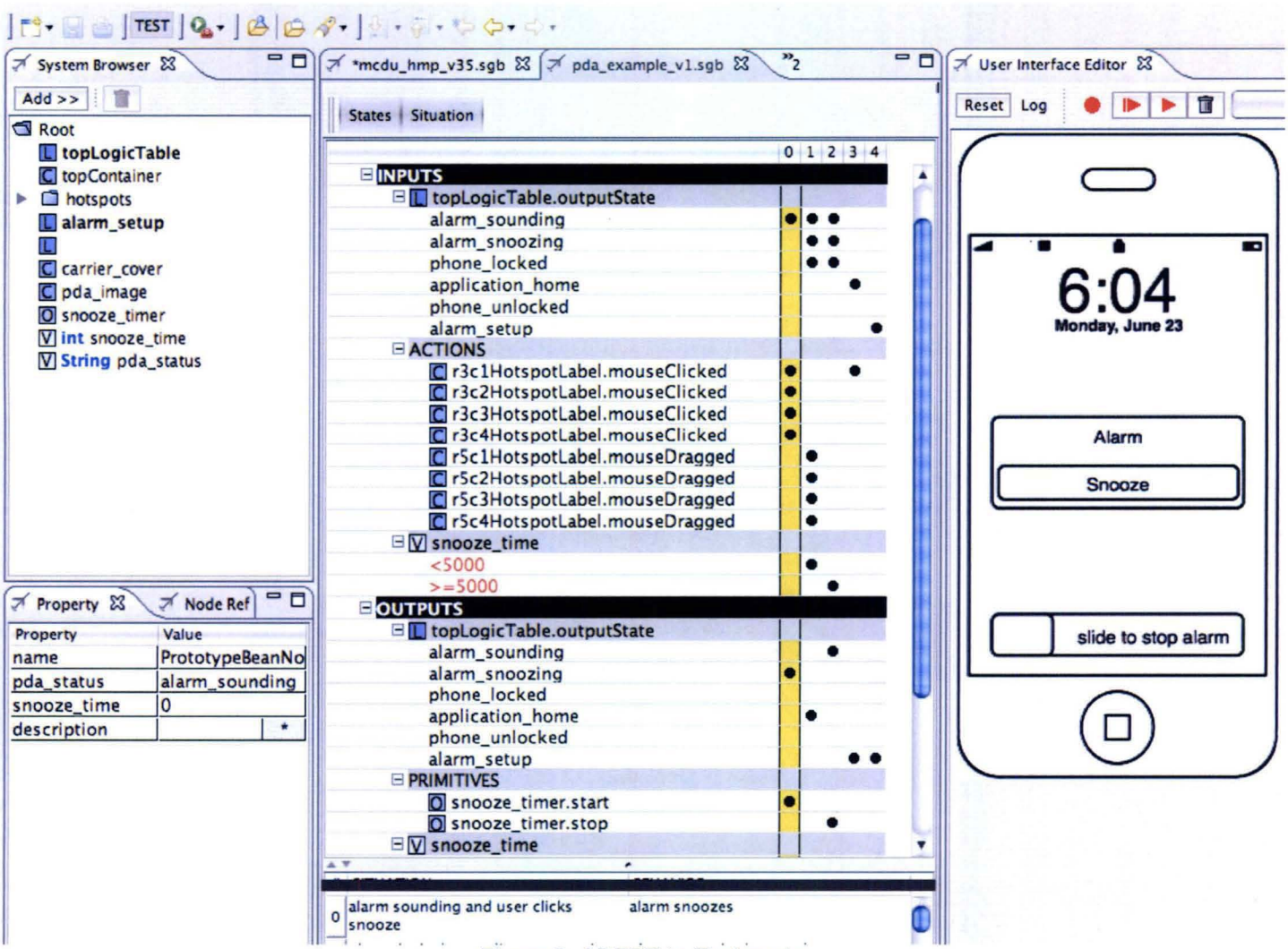

Figure 2. ADEPT in *Test* mode

Iterative Build and Test

ADEPT works in two modes, *Build* and *Test*. In Build mode the designer creates graphic objects to the User Interface editor, and adds these as well as system objects (e.g. sensor inputs) to the Logic Editor. The designer then uses these objects to construct the logic table. By testing each column as it is added, the designer can start with very simple behavior and iteratively add complexity.

The User Interface Editor

The User Interface (UI) Editor provides the tools to allow the designer to construct the interface. The User Interface Editor Design Mode Menu is shown at the top right of figure 1. The menu allows UI objects to be added, deleted and arranged.

Figure 1 also shows an example interface constructed in the User Interface Editor with transparent objects on top of the image (shown with blue outlines in figure 1) to create the functionality. The interface could be made to look even more realistic by importing higher quality images built in other graphical applications.

The Logic Editor

The Logic Editor is what differentiates ADEPT from a graphics design application. The Logic Editor, derived from the Operational Procedure Table (OPT) method (Sherry, 1996) allows the designer to specify the behavior of the device, and/or the device interface built in the UI Editor.

ADEPT uses a tabular representation of a finite state machine. In contrast to typical state transition tables, the representation used by ADEPT focuses more on presenting information about the situation (input combination) - automation behavior (output combination), and less on presenting information about state transition in a summarized form information. This focus allows a more compact notation, which enables the designer to see more behaviors, making it easier to make a complete specification.

The primary method for building a Logic Table is to select an object in the Object Browser and Drag and Drop it into the table as an input or output. This works for adding inputs and outputs, but it also works for adding variables and other objects as input conditions and output functions.

The table consists of a listing of Inputs and Outputs on the Y-axis, and columns of situation-automation behavior pairs along the X-axis, as shown in figure 2. Figure 2 shows that a black separator bar denotes the Inputs and Output Fields. The Input bar can be translated as an "**IF**" statement, while the Output bar is read as a "**THEN**" statement. Between the Inputs and Outputs bars, the thick gray lines between represent "**AND**" statements, and thin gray lines represent "**OR**" statements. Note that the outputs only contain "**ANDs**". The thin gray lines are only used to make the table easier to read.

The pda example shown in figures 1 and 2 can illustrate how the tables are used. In this example, there are two ways to silence the alarm. First, the user can press the snooze button (shown in column 0), or the user can unlock the pda (and go to the alarm page to stop the alarm, which isn't shown in this example).

Examining through the table, column 0 says:

IF
The alarm is sounding
AND
Any of the row three interface areas (i.e. the snooze button for this page) is clicked
THEN
The pda status will change to
alarm_snoozing

Similarly, column 1 is read as:

IF
The alarm is sounding
OR
Snoozing
OR
Locked
AND
The row five interface areas (i.e. the lock slider button for this page) is clicked and dragged
AND
The snooze time is less than 300000 milliseconds (i.e. 5 minutes)
THEN
Go to the application page

Figure 2 also illustrates how ADEPT can be used to design iteratively. Once the inputs and outputs have been defined, the tabular representation enables the designer to add and test situation-behavior pairs individually using the automatic code generator described in the next section.

Evaluating Prototypes built in ADEPT

A number of features have been incorporated into ADEPT to aid the designer in evaluating a device and its interface behavior. The automatic code generator creates an executable specification enabling rapid build and test cycles. Figure 2 shows the different functions available in the Test mode of the UI Editor, in contrast to the menu available in design mode, shown in figure 1. The menu shows the buttons for the Reset function, the Log function, and the Scenario Management function, which includes the Record, Reset - Play, Play, and Delete buttons and the Configuration menu.

The Log function is used to begin to record all user actions and all automation behaviors of the device prototype. The Log function generates two files at the moment, one of which is used for traditional usability evaluation and the other is used as a data source for computational human performance models.

The Scenario Management utility consists of the ability to record, playback and delete various configurations that is useful for evaluating the device against different tasks. Pressing the Record button once records all of the user actions and device information. Pressing the Reset-Play button first resets then plays the configuration selected on the configuration menu, while pressing play configures the prototype starting from the existing configuration.

Method

Three case studies were conducted to test the usability of the ADEPT software. The case studies examined three participants using the ADEPT to design actual prototypes. As the case studies examined the use of the tool across different applications with varying complexity of design, traditional performance metrics (e.g. time, errors, etc.) were not applicable. Therefore descriptive and qualitative measures were used, consisting of complexity metrics and questions about the usability and usefulness of ADEPT.

The four questions consisted of:

1: Can designers build testable prototypes in ADEPT?
2: Does ADEPT support rapid iteration and modification?
3: Does ADEPT focus the design activity on precise and complete specification of the automation behavior?
4: Does ADEPT support communication with other design team members?

The three case studies involved the use of ADEPT over a period of between one and six months, and the information gained during these time periods would not have been adequately captured through the use of interview or questionnaire techniques.

Results

Table 1 provides an illustration of the size and scale of the different projects shown in columns corresponding to each participant (P1, P2, and P3).

Table 1. Complexity Metrics for the 3 case studies

Metrics/Questions	P1	P2	P3
Source Lines of Code	1500	20000	4300
Automation behaviors	13	560	13
GUI objects	18	160	100

Table 2 shows the responses to the four questions of the 3 case studies.

Table 2. Question responses for each case study

Questions			
1: Construction?	Yes	Yes	Yes
2: Rapid Iteration?	Yes	Yes	Yes
3: Completeness?	Yes	Yes	Yes
4: Communication?	N/A	Yes, needs improvement	Yes, Needs improvement

The results show that tool did enable all of the designers to build prototypes that suited their purposes. This by itself is a notable success for the designers without programming expertise. The designers reported that they felt the tool supported rapid iteration in the design process, a key component of good design. The participants also reported that they felt ADEPT helped them to build more precise and complete specifications of automation behavior, however they felt that some work was needed to make the prototypes they designed understandable to others in their design group.

All of the participants expressed some displeasure with the organization of the hierarchy, and the means of using variables to transfer behavior information in design projects with multiple tables. One of the primary objectives of the tool is the facilitation of communication between design team members.

The evaluation was only intended to validate that domain expert designers can use the proof of concept version of ADEPT to construct testable prototypes, however the case studies served an additional purpose beyond simple validation. Valuable lessons were learned from the length of the case studies and the wide range of expertise of the three participants.

Discussion

Given the constraints with evaluating new design tools the case studies provide an example of the strengths and weaknesses of ADEPT. Although none of the case studies involved the use of a complete version of ADEPT in a real-world design process, the case studies did test different portions of the tool in real-word design problems. The results of the design exercises and the impressions of the users were positive enough to validate the initial proof-of-concept version of the tool.

This resolves the first development challenge, as domain expert designers can use ADEPT to design testable prototypes without extensive programming expertise or training. The responses from the case study participants indicated that they were able to focus on domain goals and objectives for the devices they were constructing, which was defined as the primary obstacle in software design Curtis et al. (1988).

The case studies have shown that while interpretation of individual tables by novice designers is achievable with the tabular representation, the current organization of multiple tables or representation of multiple tables in a project may obscure the understanding of complete behavior specification of a device.

This has been modified in subsequent versions of ADEPT with the creation of a "Logic Table" object, and the replacement of the action – behavior – feedback table hierarchy with only one "Top Logic Table". In this way individual designers can tailor the organization of multiple tables to suit their needs by connecting the different tables with the Logic Table objects. Additionally, new visualization techniques are being explored.

The case study evaluations also revealed a need for the creation of a library of objects to ease the construction of devices. This is especially true of complex projects where an architecture template can speed the initial construction of the device. This need will be addressed over time as ADEPT gains exposure.

Conclusions and Future Work

The initial results from the case studies have shown that ADEPT is usable by domain expert designers without requiring extensive programming expertise. While further development work is needed, these results show that ADEPT is suitable as a platform upon which to integrate HAI analyses.

In addition, the case studies showed that an ADEPT-like tool could help fill a niche. By creating a lower fidelity, but still testable prototype in less time with fewer resources, more iteration is possible, which can improve the design process (Gould and Lewis, 1985; Poltrock and Grudin, 1996).

A challenge for the future is the integration of task information. A decision was made to focus on the specification of automation behavior, and leave task information specification for future versions. Rasmussen (1994) Hoffman et al. (2002), and Feltovich et al. (2004) have expressed the need for greater involvement of domain experts in the design process, and the case studies have shown how ADEPT can facilitate greater involvement. A plan to add integrated Human-Automation interaction analyses, should begin to address this need, however the addition of a usable means for integrating or importing the results of other task decomposition or task analysis tools is an idea that deserves future research.

Acknowledgements

Thank you for technical assistance from Lance Sherry, (George Mason University), Arash Aghevli, and Eugene Turkov, (Singer – Gaffarian Technologies) and Rohit Deshmukh, (San Jose State University).

REFERENCES

BASI (1999) Advanced Technology Aircraft Safety Survey Report. *Flight Safety Digest Special Issue*. Flight Safety Foundation June – August, 1999 pages 137 – 216. Available at http://www.basi.gov.au/sprog/advtek.htm (9/00)

Curtis, B., Krasner, H., and Iscoe, N. (1988). A Field Study of the Software Design Process for Large Systems, *Communications of the ACM*, 31(11), ACM, p. 1268-1287.

Federal Aviation Administration (1996) *The interface between flightcrews and modern flight deck system.* FAA human factors team. Federal Aviation Administration., Washington, DC, Available at http://www.faa.gov/education_research/training/aqp/library/media/interfac.pdf

Feltovich, P. J., Hoffman, R. R., Woods, D., and Roesler, A. (2004). Keeping It Simple: How the Reductive Tendency Affects Cognitive Engineering, *IEEE Intelligent Systems*, May-June, IEEE Computer Society Publications Office, Los Alamitos, CA, p.90-95.

Gould, J. D. and Lewis, C. (1985) Designing for Usability: Key Principles and What Designers Think, *Communications of the ACM*, 28, (3), ACM Press, New York, NY, p. 300-311.

Hoffman, R. R., Klein, G., and Laughery, K. R. (2002) The State of Cognitive Systems Engineering. *IEEE Intelligent Systems*, January-February, IEEE Computer Society Publications Office, Los Alamitos, CA, p.73-75.

Leveson, N. (1995) Safeware: System Safety and Computers. Addison-Wesley, New York, NY, USA.

Mellor, P. (1994) CAD: Computer-Aided Disaster, *High Integrity Systems*, 1(2), p.101-156.

Poltrock, S. E. and Grudin, J. (1996). Organizational Obstacles to Interface Design and Development: Two Participant Observer Studies, In *Human Computer Interface Design* Rudisill, M., Lewis, C., Polson, P. G., and McKay, T. D. (Eds.), Morgan Kaufmann Publishers Inc., San Mateo, CA. P. 303-337.

Rasmussen, J. (1994) *Cognitive Systems Engineering.* John Wiley and Sons, New York, NY.

Sheridan, T. (2002) *Humans and Automation: System Design and Research Issues*, John Wiley & Sons, Inc., Santa Monica, CA.

Sherry, L. (1996) Personal communication regarding the use of the Operational Procedure methodology in the design of Honeywell product development.