2007

HEMETS – Human Error Modeling for Error Tolerant Systems

Michael E. Fotta

Shawn Nicholson

Michael D. Byrne

HEMETS – HUMAN ERROR MODELING FOR ERROR TOLERANT SYSTEMS

Michael E. Fotta & Shawn Nicholson
Innovative Management & Technology Services
Fairmont WV

Michael D. Byrne
Rice University
Houston TX

HEMETS simulates a human operator interacting with a prototype user interface for a proposed system design. HEMETS is built upon a Human Error Modeling Architecture (HEMA) – extension of ACT-R. A subsystem, HEMA System Interface (HEMA SI), has also been developed to enable the real time translation of user interface information into a cognitive model readable form. HEMETS will enable designers to foresee the human error consequences of a design and take steps to eliminate, reduce or recover from those errors.

Human Error Modeling

Past and current efforts at simulating human error are generally aimed at taxonomic approaches or small subsets of very controlled errors. Some of the most well known of the taxonomies are the Generic Error Modeling System (GEMS) approach (Reason, 1990), the stages-of-action model (Norman, 1986) and the Cognitive Reliability and Error Analysis Method (CREAM) (Hollnagel, 1998). They do not, however, extend to modeling behavior to predict errors.

There have been attempts to predict errors at a more mechanistic level, but these usually address only one type or class of error. For example, Byrne and Bovair (1997), presented a computational account of a class of errors known as post completion errors (e.g., leaving a bankcard in an ATM). Anderson, Bothell, Lebiere, and Matessa (1998), used ACT-R as a model and showed that it was possible to predict both the rate and content of the errors made in a task.

There is no strong model or architectural foundation for modeling the wide variety of errors which humans can make in a complex domain. However, computational cognitive architectures such as ACT-R, EPIC or SOAR provide software environments necessary to model much of the human behavior necessary to simulate human error. Thus, a cognitive architecture, properly modified, could serve as a basis for a predictive model of human error (Byrne, 2003).

We chose to use ACT-R (Anderson, Bothell, Byrne, Douglass, Lebiere, & Quin, 2004) as the basis for developing a Human Error Modeling Architecture (HEMA). ACT-R contains mechanisms which can produce "erroneous" behaviors even when the ostensibly "correct" pieces of knowledge are present in the system.

Application of HEMA

HEMA is being implemented within a complete modeling tool (HEMETS – Human Error Modeling for Error Tolerant Systems) which can model both correct and erroneous human performance. This work addresses a Small Business Innovative Research (SBIR) grant to develop a software system to assess a user interface in order to predict the human errors likely to occur if the proposed system is implemented. This can help to decide between alternative systems and recommend design changes.

HEMETS Approach

Figure 1 shows the HEMETS process for assessing errors by cognitive error modeling. There are three major activities. First there is 1. Model, the modeling of the operator's knowledge and interactions with the user interface. This operator model is built using the Human Error Modeling Architecture (HEMA).

Second, is 2. Run Simulation which would use the operator model interacting in real time with the user interface. The HEMA SI (HEMA System Interface) enables direct real time translation of the running UI to HEMA readable information. This enables the operator model to simulate interacting with the UI in real time.

Finally, there is 3. Feedback to Designer. This activity will use an Error Tracer mechanism to find the Predicted Errors output from the operator performance resulting from the simulation. An analyst (or in the future an expert system) with knowledge of both the UI and good UI design practices analyzes the predicted errors and produces a report (REDUCE - REcommend Design Upgrades reducing Cognitive Errors) which the system designer can use to improve the design.

Phase 1: Developing HEMA

Since HEMA was to be based on ACT-R, we needed to establish how we should modify and/or add to ACT-R to better model error. In order to accomplish this we had to determine what error related mechanisms are needed to model a large portion of the errors which humans make. In order to establish a reasonable, but still large, set of errors to consider, we used the GEMS, CREAM and Situation Awareness taxonomies established by Reason (1990), Hollnagle (1998) and Endsley (1999) respectively. This yielded a set of 78 error types spanning a large gamut of perceptual, cognitive and some motor errors. For more detail the interested reader is referred to Fotta, Byrne & Luther (2005).

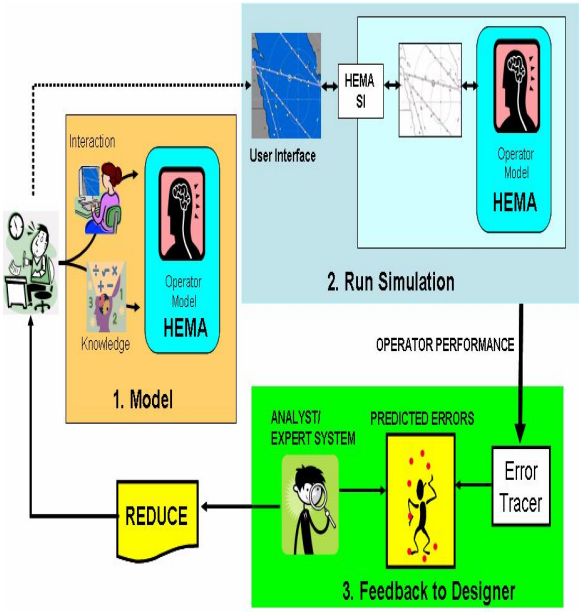

Figure 1. HEMETS Activities

From Errors to Error Mechanisms

Since we are using a cognitive architecture this means we must identify what cognitive mechanisms are at work but failing to work properly (or pushed beyond limits) in order to produce each error type. This was done by first establishing a general Framework of Human Performance (FHP) which describes the process of how an operator performs cognitive functions in an environment likely to produce a variety of human errors (Fotta, et. al., 2005). The source within this framework and a perceptual/cognitive/motor explanation for each error was then identified.

We then inspected these explanations for commonalities, i.e. the same or very similar explanations for different types of errors. Our reasoning was that if we could identify commonalities within these explanations this would enable us to derive a constrained list of General Error Mechanisms. In fact we found it necessary to hypothesize the existence of only 15 error mechanisms, although two did have subsets.

The error-related general mechanisms needed to simulate errors are:

- *Plan developer*: Develops plans. Can develop incomplete or inappropriate plan.
- *Compare Actions:* Compares expected action to action taken. Fails due to monitor or bias.
- *Monitor*: Performs comparison to evaluate plans. Can fail to monitor.
- *Plan Controller*: Runs plans. Can fail in various ways, e.g., by failing to continue running plans.
- *Attention (for perceptual information):* Allocates attention to perception, but can fail to do so.
- *Bias mechanism*: A bias which would tend to yield positive comparisons in Compare Action.
- *Rule match*: Matches current information to stored rules. Can fail to retrieve correct rules or apply correct action side. Contains subsets.
- *Schema match*: Matches information to entire schema. Can fail to correctly match.
- *Time constraint mechanism:* Places time constraint on activities, e.g., choosing a rule.
- *Decay:* Reduction of strength of information (e.g., with chunks, rules, intention, plan).
- *Poor Learning (encoding):* Stores incorrect rules, but need to be (somewhat) logically related to learning in previous similar situations. Contains subsets.
- *Retrieval mechanism*: Retrieves information, but can fail to correctly retrieve.
- *Perceptual Mechanism:* Inputs perceptual information. Can fail to correctly perceive.
- *Association Developer*: Develops associations from memory, but can fail, e.g., by developing narrow association net when deeper one needed.
- *Motor Mechanism:* Performs motor execution, but can fail to perform necessary action.

Note that not all of these mechanisms are directly error causing mechanisms. Many are functions that will have to be simulated (e.g., the Plan Developer) to account for the cognitive processes which lead to the error types within the FHP framework. Some mechanisms on the other hand are directly related to errors (e.g., the Bias Mechanism).

Human Error Modeling Architecture

Based on our analysis we proposed a HEMA shown in Figure 2. Changes from ACT-R are shown in red. Much of HEMA could be based on ACT- R, but there would be substantial changes needed to the Perceptual and Motor modules and to the way Chunks are represented.

Planning would need to be accommodated by implementing a Plan Developer and a Plan Controller module with appropriate buffers (Intention buffer for the former, Plan Buffer for the latter). It is also likely that Attention (instead of being accomplished as part of other modules) would need to be implemented as a separate module.

A detailed report of the Phase 1 work is presented in D.N. American, 2003.

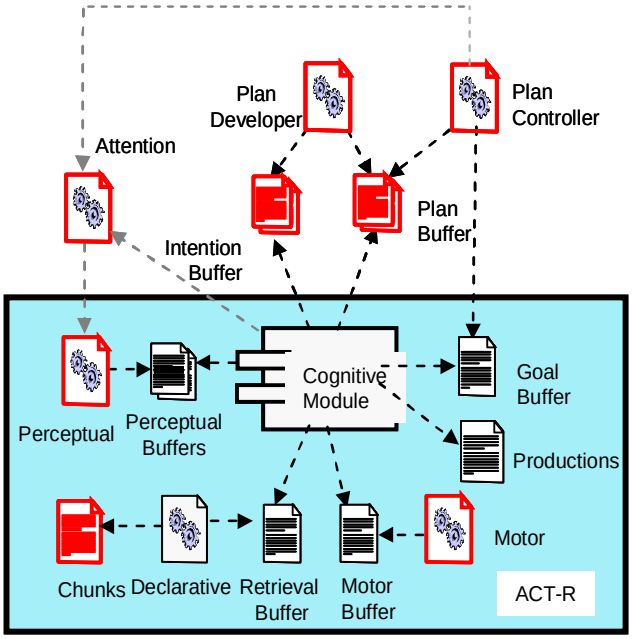

Figure 2. Human Error Modeling Architecture (HEMA)

Phase II: Implementing HEMETS

In Phase II we began the implementation of the entire HEMETS concept. Specifically we performed tasks to enable the first two HEMETS activities shown in Figure 1. Tasks to enable the third activity will be performed during the SBIR options.

In order to implement HEMETS we performed or are performing the following:

- Identified scope of a realistic "Proposed Design".
- Model operator interaction and operator knowledge.
- Implement HEMA – develop new modules, modify ACT-R.
- Develop HEMA System Interface (HEMA SI).
- Validate HEMETS: Compare HEMETS simulations to human performance.

Proposed Design – An Air Warfare Simulation

We first identified a "system" for which we could prototype HEMETS. In order to have access to all code and the capability to modify the system for data collection it was decided that using a simulation would give us the best control.

Figure 3 shows the user interface we are using in place of a proposed design - the CHEX (Change History EXplicit) air warfare simulation produced by Pacific Science & Engineering (PSE).

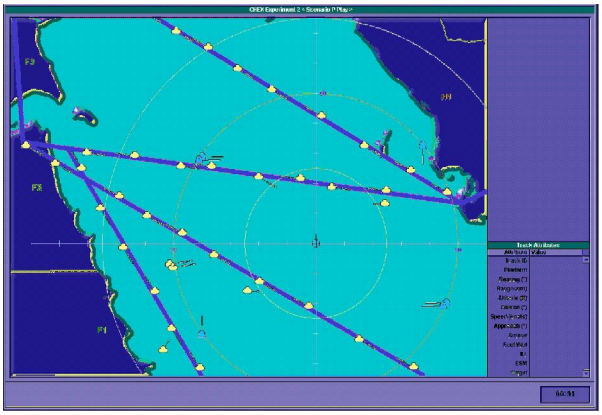

Figure 3. User interface used

Operators monitor the screen as unknown aircraft (small yellow objects) moved around their ship - ownship. The latter is the crossed-circle in the center of innermost ring. Their task was to detect seven types of aircraft changes and report a change by moving a cursor to an aircraft of interest and then clicking the mouse. This brought up text information in the lower right hand corner. The operator then clicked on the changed attribute. The seven attributes which could change are:

- Cross Rings – crossing a "range rings".
- Airlane – leaving an airlane.
- Approach – when an aircraft changes direction from toward ownship.
- Speed – accelerating (any increase in speed).
- Altitude – descending (any decrease in altitude).
- Detect – aircraft appearance.
- Feet wet – crossing from land to water.

There were also interruptions that lasted either 30 seconds or 120 seconds. This was to simulate the interruptions CIC personnel often encounter.

Modeling Human Interaction and Knowledge

We began with information from PSE's experience with this interface and their interaction with actual Combat Information Center personnel. Starting with a cognitive task analysis of the user's performance with this interface we first developed a general process flow of the user's activities. This was then detailed to account for each likely perceptual, memory, decision and motor step an operator was taking in interacting with this interface. We then defined what we would have to do to model each step using the ACT-R environment and where specific user interaction mechanisms (such as a user setting a reminder to return to an aircraft) were needed in ACT-R. Finally the model was developed using ACT-R with some new modifications and additions as discussed in the next section.

HEMA Implementation

Improving ACT-R's Perceptual Modeling. In ACT-R perception is accomplished by first defining each object's properties. Next tests are performed for constraints on these properties and relationships. The relationships may be between an object's properties and a desired property (e.g. equality or inequality) or a relationship between an object and a position. Two examples of this latter position relationship are *Nearest* (to current attended position) or *Highest x-coordinate* on the screen. With properties and relationships ACT-R can find, for example, the Nearest object to the object whose Kind = square and color = blue.

However, we found the current set of allowed properties in ACT-R (seven as shown in Table 1) and relationships (five) to be too restrictive to allow the scanning we needed to model, e.g., around range rings or along air lanes as shown in Figure 4. We therefore modified ACT-R's perceptual module to allow the user to define any property or relationship.

Table 1. Properties & Relationships in ACT-R

Constraint Type	Possible Types
Properties	Color Size Kind Value X-pos Y-pos Z-pos (dist.)
Relationships	Nearest Equalities Inequalities Highest Lowest

This enable us to define an "Around" relationship to scan around a range ring and an "Along" relationship to scan along the air lanes as shown in Figure 4.

Although we only needed to define a few new relationships for this specific UI our extension from limited pre-defined to unlimited user definable properties and relationships will enable HEMA to be used with a wide variety of complex interfaces.

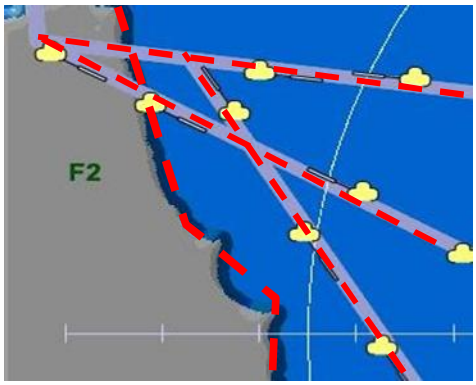

Figure 4. Scan "along" objects

Refining ACT-R's Memory Modeling. Currently in ACT-R information is stored into the declarative memory module when chunks are cleared from other modules' buffers. In our situation, this would mean that when the visual buffer was cleared, we would have in memory the chunk describing all of the object's visual features for later recall.

However, when you tried to recall an aircraft and its associated properties later, if you were successful at retrieving the right aircraft, you would have all the correct values for all of the properties of that aircraft. From an error generating standpoint this is not psychologically sound. We want to be able to store some attributes of an aircraft recalled correctly while allowing other attributes to be misremembered or not recalled at all.

To do this, we construct a chunk representing the individual attribute data to be remembered and then store this chunk in the declarative memory module. In this way, we can custom tailor chunks in memory to contain certain information about an aircraft which we can later recall. Attributes of an aircraft can then be separated from each other, so recall of one doesn't guarantee a right/wrong answer to another.

This appears to be a more reasonable approach to modeling human error production when searching interfaces such as CHEX Replay or Combat Information Centers.

Developing New Planning Module. As discussed above we needed to implement new mechanisms to handle errors due to planning failures. While we had initially proposed only Plan Controller to mimic human tracking of a plan, on implementation it became apparent that we needed to also have a Task Controller. A task is an "atomic" sub-set of a plan which logically groups together and can be written as a series of productions. The Task Controller runs and tracks the tasks within a plan. A schematic of this current implementation is shown in Figure 5.

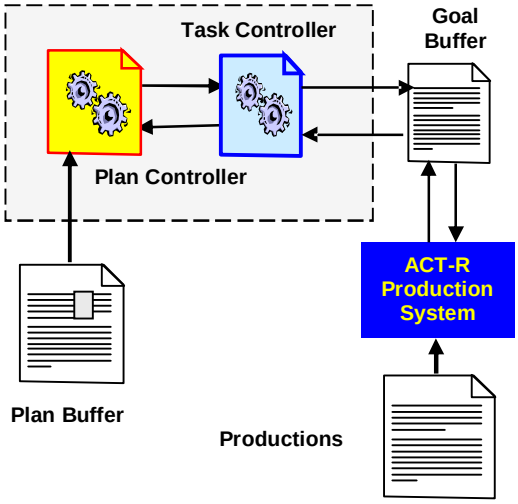

Figure 5. Planning in HEMA

The Plan Controller reads the current plan from the Plan Buffer. An example plan for the simulation used is: Scan Scene. The Plan Controller passes the appropriate task (e.g., Scan 25 mile range ring) to the Task Controller.

The Task Controller then places the first step in this task (Locate Object) in ACT-R's Goal Buffer. The Productions to complete each task are previously coded in ACT-R. Thus, ACT-R's Production System can use these productions to model a person following the detailed steps in a plan; in this case Locating an Object on the 25 mile ring. When this goal is completed the goal buffer is cleared and the current state information is passed to the Task Controller and saved in declarative memory. The Task Controller then passes the next step to the goal buffer. This process continues until the Scan 25 mile ring task is completed at which time the Plan Controller advances the Scan Scene plan to the next task and passes this to the Task Controller.

HEMA SI Implementation

As Figure 1 shows, in order to run a complete simulation we must have a subsystem which can directly translate the user interface in real time into ACT-R readable data. This HEMA System Interface (HEMA SI) is built on a scene analysis system - SegMan (St. Amant, Riedl, Ritter, & Reifers, 2005). SegMan is an image processing utility designed to be a bridge between user interfaces and cognitive modeling systems such as ACT-R. The development of HEMA SI extends SegMan, resulting in a design of greater generality for processing visual scenes.

SegMan examines the color of pixels in a scene, groups sets of like-colored pixels, and then classifies them based on shape. Since HEMA SI is applied in a real-time simulation, efficiency is critical. We have redesigned SegMan at the system and algorithmic level to provide the ability to recognize new classes of objects in a more robust fashion.

In the CHEX Air Warfare task simulation interface (Figure 1), the key elements for recognition include the Air Lanes, Land, Range Rings, and Aircraft. The cognitive model must simulate a user detecting critical changes to the Aircraft such as crossing a Range Ring or leaving an Airlane.

Supporting these detection tasks required addressing shortcomings in SegMan's recognition of objects due to occlusion and the dynamic nature of a simulation (objects moving around and over each other). These issues are handled as follows:

- *Specialized visual processing micro-strategies.* An aircraft is represented as an icon with a speed and direction indicator projection. We identify the icon by shape, visually trace a circle to find the projection, and finally track the linear shape of the projection to determine the aircraft's direction.
- *Object continuity computation.* Several aircraft at various points on an air lane interrupt air lane continuity by occlusion. We identify the air lanes' separate segments, use a least-squared line fitting algorithm, and a line projection algorithm to merge the segments into a single line object (see Figure 6).
- *Variable-resolution scanning.* Processing every pixel in an image is too time-consuming for real-time performance. HEMA SI includes variable-resolution scanning, i.e., processing pixels on the intersections of a grid overlaid on the image. Once any pixel in an object has been distinguished from its background, a specialized strategy can take over for recognition.

- *Persistency testing.* We added a persistency update algorithm to allow HEMA SI to compare aircraft in the current scene to aircraft in the previous scene and, based on position, determine the correlation between them.

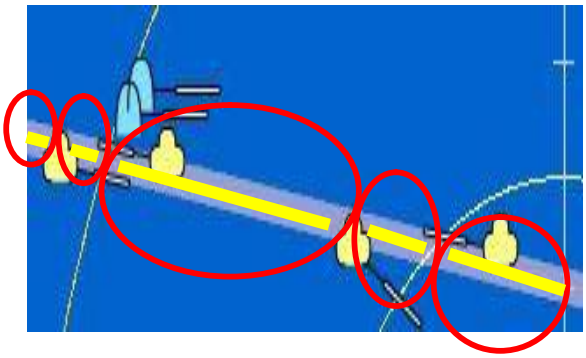

Figure 6. Recognizing an air lane

Currently HEMA SI can take a visual scene presented by a real-time user interface and provide output to any system that needs to process objects from it. HEMA SI is currently implemented to output the objects to an ACT-R system, but can be interfaced with any cognitive modeling system using position, object, and movement information from the scene.

Validation

In order to validate HEMETS we will compare the results of repeatedly running our modeled operator in HEMA, interacting with the user interface, to the error results of users interacting with the air warfare simulation. This phase of the study is being undertaken now.

Summary

We have identified a set of error related General Mechanisms for inclusion in an error modeling cognitive model and designed the Human Error Modeling Architecture (HEMA). Improvements to ACT-R's perceptual module and memory modeling have been developed as well as a new plan controller and task controller modules. A subsystem, HEMA SI, has been developed to enable real time translation of user interface information into a cognitive model. Validation is being performed by running our modeled operator and comparing results against human users of the air warfare simulation.

Future work will involve developing the automated Error Tracer to enable feedback to the designers and extending HEMA SI capabilities to other user interfaces.

References

Anderson, J. R., Bothell, D., Byrne, M. D., Douglass, S., Lebiere, C., & Quin, Y. (2004). An integrated theory of the mind. *Psychological Review*, 111(1036-1060).

Anderson, J.R., Bothell, D., Lebiere, C., and Matessa M. (1998). An Integrated Theory of List Memory. *Journal of Memory and Language*, 38, 341-380.

Byrne, M. D. (2003). A mechanism-based framework for predicting routine procedural errors. In R. Alterman & D. Kirsh (Eds.), *Proceedings of the Twenty-Fifth Annual Conference of the Cognitive Science Society*. Austin, TX: Cognitive Science Society.

Byrne, M.D. and Bovair, S. (1997). A working memory model of a procedural error. *Cognitive Science*, 21, 31-61.

D.N. American, *Human Error Modeling Architecture (HEMA) Technical Report*, Serial Number: N00014-03-M-0357-0001AD, December 26, 2003.

Endsley, M.R. (1999). Situation Awareness and Human Error: Designing to Support Human Performance. *Proceedings of the High Consequence Systems Surety Conference*, Albuquerque.

Fotta, Michael E., Byrne, Michael D. and Luther, Michael S., (2005) Developing a Human Error Modeling Architecture (HEMA), *HCI International, Vol. 11 - Foundations of Augmented Cognition*, Lawrence Erlbaum and Associates, Las Vegas

Hollnagel, E. (1998). *Cognitive Reliability and Error Analysis Method*. Oxford, UK: Elsevier.

Norman, D.A. (1986). Cognitive Engineering. In D.A. Norman, S.W. Draper (Eds.) *User Centered System Design: New Perspectives on Computer Interaction* (pp. 31 - 61). Hillsdale: Lawrence Erlbaum Associates.

Reason, J.T. (1990). *Human Error*. New York: Cambridge University Press.

St. Amant, R., Riedl, M. O., Ritter, F. E., & Reifers, A. (2005). Image processing in cognitive models with SegMan. (Invited.) *Proceedings of HCI International*.