Architectural Explorations for Modeling Procedural Skill Decay

Ronald S. Chong (rchong@gmu.edu)
Applied Research in Cognition and Human Factors
Department of Psychology, MSN 3F5
George Mason University
Fairfax, Virginia 22030-4444

Abstract

There is great interest in mission-critical domains to minimize the decay of expert skill: the decrement in performance observed after a period of disuse. To better understand skill decay and to test theories and systems aimed at minimizing its effects, it is useful to create computational models capable of exhibiting its main effects. No models of expert knowledge (represented as production rules) skill decay have been developed within cognitive architectures. This work explores how skill decay might be realized within an architecture. It also discusses implementation issues with a focus on psychologically-plausible approaches and explanations of skill decay effects. Although implemented in the EASE architecture, the issues discussed in this paper are relevant to all rule-learning, production-system based cognitive architectures.

Introduction

The adage "practice makes perfect" summarizes the fact that effort is needed to attain and maintain expert levels of performance. Empirical work has shown a that a break from training or performance results in decrements in task proficiency; see Arthur, Bennet, Stanush & McNelly (1998) for a review. Specifically, this robust phenomena, termed "skill decay" produces performance decrements that are positively associated with the length of the period of non-use (*retention interval*).

There is great interest in both civilian and military domains to find ways to minimize the effect of skill decay. To better understand the issues surrounding skill decay and retention, and to then test theories and systems aimed at minimizing its effects, it is useful to create computational models capable of exhibiting its main effects.

Four computational cognitive architectures—EPIC, ACT-R, Soar, and EASE—represent *expert* knowledge as production rules. However, to date, no models of procedural skill decay have been demonstrated in these systems:

- EPIC (Kieras & Meyer, 1997) does not have a rule learning mechanism.
- ACT-R (Anderson & Lebiere, 1998) contains many learning mechanisms, but most are focused on declarative knowledge. Many different production learning approaches have been explored and abandoned. The most recent approach being developed is production compilation (Taatgen & Lee, 2003).
- Soar (Newell, 1990), from its inception, has included a rule learning mechanism called chunking (Laird, Rosenbloom, & Newell, 1984). Chunking, a form of knowledge compilation, has been used to model many forms of learning but not the decay of skill.
- EASE (for **E**lements of **A**CT-R, **S**oar and **E**PIC) is a recent integrated hybrid architecture based on the recognition that the previous architectures contain complementary mechanisms. Initially, the cognitive processor of EPIC was substituted for Soar while keeping EPIC's sensory, perceptual, and motor processes. This integration was called EPIC-Soar (Chong, 1998). With the addition of a version of ACT-R's base-level learning (BLL) mechanism to Soar's declarative memory (Chong, 2003) and the work reported here, the integration was renamed to EASE. With regard to modeling skill decay prior to this work, EASE has no greater capability than its ancestor architectures.

Unlike declarative memory in ACT-R and EASE, learned rules (procedural memory elements) in ACT-R, Soar, and EASE are retained regardless of their recency or frequency of use or their utility in accomplishing the model's goals. Beginning with the premise that procedural skill decay cannot be modeled using the existing set of mechanisms, the research question then is: what kind of mechanism is necessary and what are its implications?

This paper reports on exploratory work that applies ACT-R's base-level learning mechanism to learned rules (procedural memory) in the EASE architecture. An analogical approach has guided this work: a declarative memory mechanism has been applied to procedural memory and the challenge has then been to map and implement declarative mechanism concepts into the procedural mechanism.

This paper briefly describes chunking, the base-level learning mechanism, and discusses novel issues that arise when applying the mechanism to rules. Also presented are some preliminary findings and a discussion of design issues.

Approaches to Modeling Procedural Skill Decay

The literature generally discusses skill decay in terms of knowledge *retention* or *forgetting*. This characterization makes ACT-R's base-level learning mechanism, a mechanism that mediates knowledge availability, a natural candidate mechanism to applying to learned procedural knowledge (rules). This is the approach used in this work.

However, other explanations of the phenomena are possible. Skill decay may be due to interference, where the decrement in performance is due to interference between the task knowledge and the non-task activities performed during the retention interval.

The production system formalism used in these architectures suggests another possible explanation: if the conditions (cues) of a rule are not satisfied by the declarative elements in working memory, the rule will not match and fire. Therefore, skill decay may not be due to forgetting, but rather due to *cue unavailability*. What may appear to be rule forgetting may be the inability to retrieve (match) a rule due to insufficient declarative cues. Perhaps the cues were not perceived. Perhaps they were perceived but not fully processed by cognition. Perhaps they were fully processed but are not accessible (decay, interference, etc.).

While a complete story of skill decay surely involves these and other mechanisms, this work focuses only on skill decay as knowledge retention/forgetting and the architectural issues associated with forgetting procedural knowledge.

Chunking in a Nutshell

EASE, by inheritance from Soar, incorporates chunking, a mechanism for learning production rules. When a model, as it is progressing towards a goal, reaches a point where it cannot continue due to a lack of knowledge—an *impasse*—the architecture (Soar or EASE) automatically creates a new problem solving context (a subgoal). Processing in the subgoal is directed at resolving the impasse so that progress towards the initial goal can be resumed. As a by-product of resolving an impasse and removing the subgoal, the chunking mechanism creates rules called *chunks*[1] that summarize the processing in the subgoal that resolved the impasse. Chunks are used the next time the model is in the same (or similar) situation that previously caused an impasse, thereby avoiding the impasse and the associated processing time.

This learning mechanism has been found to be sufficient for producing a variety of learning such as concept learning, learning from instruction, learning multiple-task coordination, and correcting faulty knowledge.

Base-Level Learning Mechanism: Details and Parameters

Base-level learning (BLL) is the ACT-R subsymbolic mechanism that determines the activation of working memory elements as a function of recency and frequency of their use. An element's activation then contributes to its availability and retrieval time. The functional role of this mechanism is to adapt declarative memory to the complexities of behavior, the task, and the environment.

A hypothesis of this work is that a similar mechanism, applied to procedural memory, is necessary to model aspects of skill decay. The version of the BLL mechanism applied to declarative memory in EASE (Chong, 2003) was applied to EASE's procedural memory. The equations that govern the mechanism are:

$$A_i = B_i + \varepsilon$$
$$B_i = \beta + \ln(\sum_{j=1}^{n} t_j^{-d})$$

A_i describes the activation of a chunk (learned rule) as the sum of the intrinsic (base-level) activation of the chunk (B_i) plus a noise component, ε. In the base-level learning equation, t_j is the time since the j^{th} reference of chunk c_i and represents a history of uses of the chunk. B_i therefore is a function of the frequency and recency of a chunk's use. As a simplification, the mechanism and equations do not use the spreading activation component found in the ACT-R mechanism and equations.

Most of the ACT-R parameters associated with this base level learning mechanism are present in this version of the mechanism. The mechanism is controlled by four free parameters, three of which are variables in the activation equation:

- *Base-level constant* (β): this value specifies the initial activation given to a newly created chunk. β in ACT-R is commonly set to 1.0.
- *Learning rate* (d): the rate of activation decay. The ACT-R default is 0.5.
- *Transient noise* (ε): noise is sampled from a zero-centered logistic distribution. A common value is 0.25.
- *Retrieval threshold*: when activation falls below this value, the chunk cannot be retrieved and is effectively forgotten. The ACT-R default value is 0.0.

These ACT-R common and default values were adopted for this work because treating established parameters as constants, instead as tunable variables, imposes strong and desirable constraints on model development and fitting, leading to more informative models of behavior.

Base-Level Learning Applied to Chunks

The mechanism as described operates similarly to base-level learning for declarative memory in ACT-R. When a chunk is learned, it receives an initial level of activation (β). The activation decays logarithmically as a function of time and frequency of a chunk's use. Each time a chunk is used, its activation is momentarily boosted and then begins to decay. When the activation of a chunk falls below the retrieval threshold, the chunk will not be retrievable and effectively forgotten.

However, because the base-level learning mechanism is being applied to rules, there are important differences in the implementation details relative to its use for declarative memory. These will be discussed now.

Handling "decayed" (forgotten) chunks

When a chunk's activation falls below the retrieval threshold (the chunk has "decayed"), the chunk must not be executed. There are two ways to accomplish this. The first is simply to excise decayed chunks. The second option is to allow

[1] In discussions of ACT-R and Soar, the term *chunk* is a common point of confusion. An ACT-R chunk is a declarative structure. A chunk in Soar (and EASE) is a rule; a procedural structure. This latter meaning is used throughout this paper. The phrase "learned rule" will sometimes be used to emphasize this meaning.

decayed chunks to remain in procedural memory and to match, but to prevent their firing.

The first option is the easiest to implement, particularly since EASE and Soar use the very complex, but highly efficient, Rete match algorithm (Forgy, 1982). However, excising decay chunks entails deleting a chunk's history of use. The consequence is that a relearned chunk will have the same likelihood of being forgotten as the original chunk. Yet, in humans, knowledge becomes *more durable* through repeated learning and use.

ACT-R's BLL mechanism accounts for this observation by retaining decayed (forgotten) declarative memory elements. When a duplicate memory element is created it is "merged" with the decayed element—their histories of use are concatenated—thereby enhancing the durability of the decayed element.

To exhibit the same effect for procedural knowledge, in EASE, decayed chunk (and their history lists) are retained (and can match) but are inhibited from firing. The next time the decayed chunk would have fired, the impasse that originally caused the chunk to be learned would recur. Once the impasse has been resolved, a new chunk would be learned and likely be a duplicate of the decayed chunk. If so, then their histories are merged, resulting in immediately higher activation and greater durability for the decayed chunk.

Activation Boosting: The Definition of "Use"

There are two conditions that cause a momentary increase in the activation of a chunk. The first condition, described above, occurs when a newly learned chunk is merged with a decayed duplicate chunk.

The second condition occurs when a chunk is *used*. In most architectures, "used" would be defined as "when the chunk fires." However, EASE (due to Soar) provides an additional definition: "a chunk is 'used' when it influences the goal-directed behavior of a model". This definition arises from EASE's fine-grained procedural knowledge representation (e.g. proposal, application, and preference knowledge) and a control structure where *every goal-directed action is subject to deliberation*.

Consider, for example, a model for playing a maze-based game. At any intersection point in the maze, the model has many options for its next move; e.g. move-forward, turn-left, turn-right, move-back. Because EASE simultaneously fires all rules that match, all four options (goals) could be simultaneously *proposed*. To resolve the conflict (only one goal can be pursued at a time), other production rules are needed that specify context-specific goal *preferences*. These can be unary ("move-left is *best* when...") or binary ("move-back is *better than* move-forward when...") propositions. After deliberation of the preference ordering, one option is selected (say, move-left) as the goal the model will pursue.[2]

Now suppose that all four options were proposed by chunks. If "use" was defined as "when the chunk fires", then all four chunks would be boosted equally. Over time, the model would not adapt to the situation where one action may be consistently preferred.

In contrast, if the mechanism boosts only the chunk that influenced the goal-directed behavior of the model (the chunk that proposed the selected move-left goal), then that chunk would have a relatively higher activation than the other goal proposal chunks (move-forward, move-right, move-back) With experience, these other chunks would decay and be forgotten from disuse, thus shortening or even possibly eliminating the deliberation process used to resolve conflicts.

Alternatively, suppose that the *preferences* were generated by chunks; i.e. the model had previously learned which goals to prefer in certain contexts. In this case, if the mechanism boosts only the preference chunk that contributed to the selection of move-left, then that chunk would have a higher activation relative to the other goal preference chunks.[3] With experience, the unused goal preference chunks would decay and be forgotten, thus simplifying goal selection. This would allow incorrect or contradictory preference knowledge learned early in training to be supplanted by higher-utility preferences learned later in training.

In summary, EASE and Soar have always been able to quickly learn (e.g. one-trial learning) new proposal, application, and preference knowledge. With the application of base-level learning to learned procedural knowledge, it appears that EASE may also be able to adapt more slowly through experience on the task.

Preliminary Evaluation of the Mechanism

Testing and evaluating the effect of the new chunk decay mechanism was performed using an existing Soar model of category learning. Miller & Laird (1996) report on a rule-based model of supervised category learning called SCA (symbolic concept acquisition). This model represents category acquisition purely as the acquisition of production rules, making it an appropriate and readily available testbed.

Subjects in a category learning task were presented a stimulus with three binary-featured dimensions; e.g. fuel = 20% or 40%; size = small or large; turbulence = light or heavy. Using only the values of the three dimensions (eight possible exemplars), subjects either allowed or denied an aircraft's altitude change request.

Patterned after Nosofsky, et al. (1994), the task consisted of three task difficulty conditions—Types 1, 3, and 6. In Type 1, the easiest condition, one feature alone was sufficient to identify the category; e.g. fuel=20% -> allow. Type 6, the most difficult condition, required considering all three features. Type 3 was of intermediate difficulty.

A between-subject design, in task difficulty, was used. Subjects performed eight ten-minute blocks. In a block, each of the eight possible exemplars were shown twice; i.e. sixteen presentations per block; 128 presentations per study. Blocks occurred back-to-back, with the exception of a brief subjective workload assessment after blocks 1 and 4.

[2] Note that Soar resolves *goal* conflict symbolically through explicit knowledge. ACT-R resolves *rule* conflict through subsymbolic mechanisms; i.e. expected utility. It also provides an optional facility to decay the parameters used in computing expected utility. These are complementary approaches that can demonstrate adaptation at different levels and time-scales.

[3] This aspect of the mechanism was not needed for the results shown in this paper, so it is still being developed.

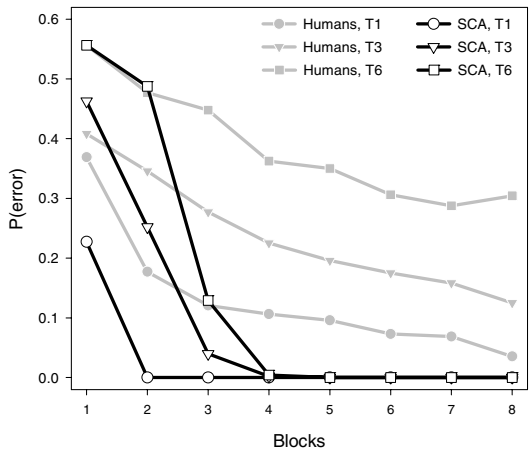

Figure 1: Comparison of SCA and human data learning rates in the category learning task (Gluck & Pew, 2002).

Figure 1 shows the fit of SCA without the new mechanism to the human data collected in the AMBR project (Gluck & Pew, 2002). The model does show an effect by problem type. However, because all learned rules are retained, it learns significantly faster than the human mean.

Introducing the chunk decay mechanism should have several effects. First, it should dramatically slow learning. It should also exhibit a skill decay effect after a retention interval. Finally, the model should also attain pre-interval performance more quickly with retraining after the retention interval than during initial training.

Comparison of Learning Rates

Figure 2 shows the result of chunk decay mechanism on SCA's learning rates. The inter-stimulus times experienced by the model were the same as those experienced by subjects. The inter-block delay, due to the subjective workload test, was not simulated for the model. To get this fit, chunks were rehearsed seven (7) times on each use. The effect of the new mechanism was close to what was predicted. The model's performance for Type 3 and 6 was significantly slowed and is now a much better qualitative match to the human data.

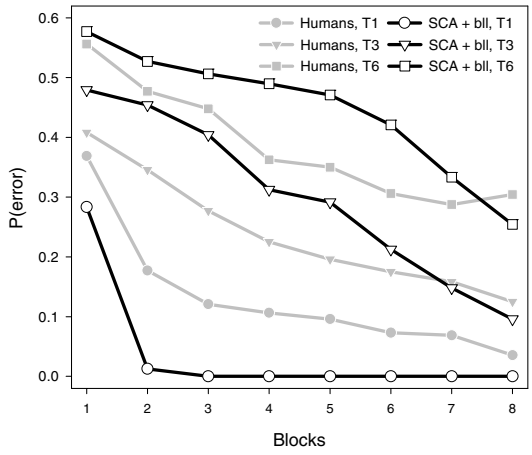

Figure 2: Effect of the chunk decay on SCA learning rates.

Interestingly, Type 1 was virtually unaffected by the chunk decay mechanism. Type 1 categorization is determined by only one stimulus feature. This suggests that the granularity of SCA's acquisition is large enough to quickly acquire the Type 1 category. The model learns Type 1, on average, by the third of the sixteen trials per block. Therefore, the rule's activation is increased with each use on subsequent trials, which quickly leads to perfect performance in later blocks.

It should be emphasized that it was neither the goal nor the expectation that applying the base-level learning mechanism to chunks would result in quantitative fits to the human data. SCA uses a normative learning strategy and the simple addition of a decay mechanism could not plausibly account for or explain the wide variety of learning strategies that subjects are known to employ. Although forgetting does influence learning in this task, there are other influences that are not represented in this model. For example, Wray & Chong (2003) demonstrate good fits and predictions to the human data by elaborating the SCA model to consider irrelevant features in the environment; a behavior that was surprisingly prevalent in post-study interviews. Even though the extended SCA model was sufficient to produce good fits to the learning data, it is unclear how it could be made to produce skill decay effects without an architectural change such as the one investigated here. A more complete story of learning in this task might be achieved by a combination of knowledge-level and architectural extensions.

Demonstrating Skill Decay

Since no skill decay data was gathered as part of the AMBR project, no comparison to human data can be made. Nevertheless, model predictions for performance after a retention interval were generated. Three retention intervals were simulated—1 hour, 6 hours, and 24 hours. These were followed by a retraining phase where the training blocks were repeated.

The three plots in Figure 3 shows that the mechanism does produce decrements in performance. It also shows that the amount of decrement increases with the length of the interval. One of the characteristics of skill decay is that reacquiring pre-interval levels of proficiency generally occurs much more quickly in retraining than under initial training. This effect was observed where block 8 performance was attained by blocks 10, 11, and 13 for retention intervals of 1, 6 and 24 hours, respectively. Note that the time to reacquire previous proficiency also increases with retention interval. This is another general observation of skill decay.

Discussion

The objective of this work was to investigate architectural approaches to modeling skill decay. The preliminary results have shown that base-level learning, when applied to learned rules, can produce the desired skill decay effects. While the new mechanism is promising, there are many open issues still to be addressed.

Effect of Skill Decay on Other Performance Metrics

Although the mechanism does show an increase in prediction error across the retention interval that is consistent with skill decay, there are other performance metrics that could be

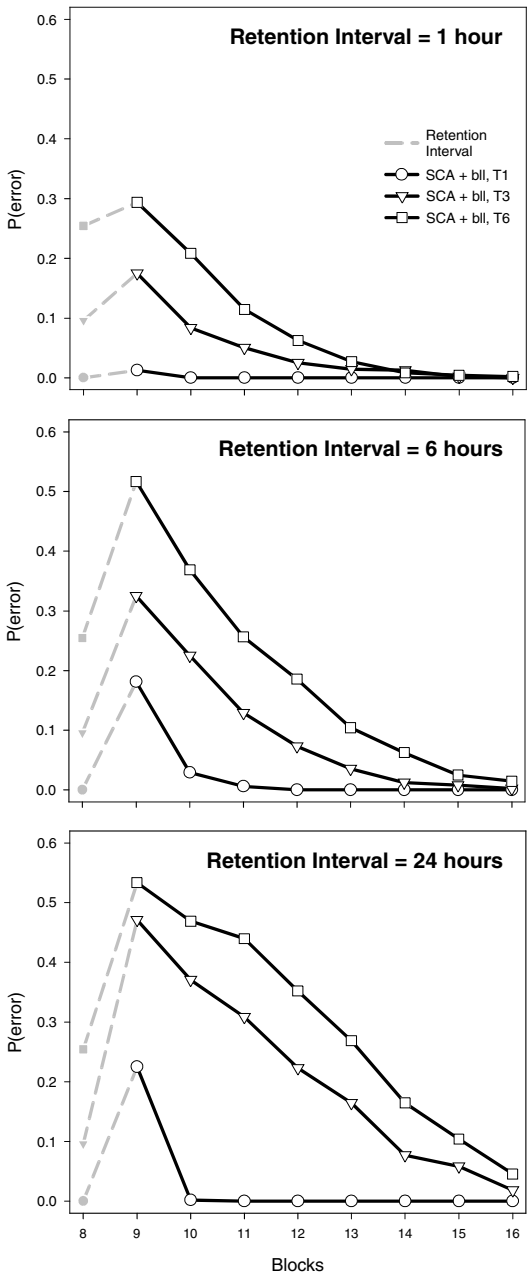

Figure 3: Performance decrement and subsequent relearning for retention intervals of 1, 6, and 24 hours.

affected by skill decay. For example, a retention interval is often found to adversely affect performance speed (e.g. reaction time).

In ACT-R, the latency of retrieving a declarative element has a functional relationship (computed by the subsymbolic layer) to the element's activation where retrieval latency increases as activation decreases. There is as yet no such mechanism in EASE.

At present, EASE, like Soar, exhibits speed improvements only as a function of practice through its chunking mechanism. Chunking compiles the potentially lengthy processing necessary to resolve an impasse, allowing the system to avoid the impasse if the model is later exposed to the same or similar situation (Rosenbloom & Newell, 1993). Through chunk merging, a decayed chunk's activation would be boosted, making it less susceptible to decay.

Approaches to Modeling the Faster Return to Proficiency

It is generally expected that only a limited amount of refresher training is necessary to reattain previous levels of performance (Arthur, et al., 1998). The chunk decay mechanism was able to recreate this effect primarily through chunk merging. There are, however, other explanations of this effect.

For example, it could be that retraining allows for relearning knowledge for selecting the best option among several candidates. As mentioned earlier EASE has a mechanism where preferences (possibly encoded as chunks) are used to select a candidate from a set of alternatives. For tasks that involve decision making, strategic thinking, or more generally selecting an action from among several candidates, EASE may be able to demonstrate the faster return to proficiency by relearning critical preferences. This observation, if correct, suggests that refresher training courses should target and exercise the knowledge used to make correct context-specific decisions.

Faster return to proficiency can also be seen as the reactivation of knowledge that then activates other associated knowledge sources that may have decayed during the retention interval. In this manner, small learning opportunities can lead to large increases in proficiency. This view suggests a spreading activation kind of effect, where chunks that are used or relearned will spread activation to associated chunks, thereby increasing their activation and potentially making a decayed chunk retrievable. ACT-R contains a theory of associative links between declarative elements and the spread of activation through these links. The application of these ideas to rules may be worth further investigation.

Chunk Rehearsals?

Thus far, declarative BLL concepts (e.g. the definition/implementation of "use", boosting, merging, and forgetting) have been mapped into the procedural BLL mechanism. The final declarative mechanism concept to be mapped is *rehearsal*, a common declarative memory reinforcement technique used in ACT-R and EASE. Declarative memory rehearsals are performed with a set of productions that repeatedly retrieve a memory element, thereby increasing its activation. (ACT-R provides an additional rehearsal facility through a Lisp function call.)

Rule rehearsal, though an admittedly odd concept, does warrant consideration, at least as a thought exercise. When considering rule rehearsal, it quickly becomes apparent that it entails two significant complications. (These complications were avoided in this work by using a temporary architectural "hack" to rehearse chunks.)

The first is that the model has to cause the desired rule to be match and fire. In other words, a model must populate declarative memory with the elements necessary to make a rule match and fire. This is a non-trivial requirement since rules are cognitively impenetrable—a model does not have access to the conditions or actions of its production rules.

The second complication is that once the desired chunk matches, its firing can have undesirable effects. For example, if the chunk to be rehearsed produces a motor command, then as it is repeatedly fired, it repeatedly sends commands to the motor system, producing possibly unwanted actions in the world. The also applies to rehearsed rules that change the model's internal state.

Previous work in Soar has leveraged its automatic subgoaling mechanism to recreate state and to isolate actions from the world. Laird (2001) created an "imagination" subgoal where the model performed planning to anticipate the behavior of opponents in a first-person-shooter. Chong (1998) created a "reflection" subgoal whenever the model discovered a performance error in a simple interactive task. In the subgoal, the model recreated the state of the environment that preceded the error. This caused the model to refire the same interactive rules that led to the error; the model effectively *replayed* these actions. In both approaches, rules fire in a context isolated from the environment.

These approaches hint that a general chunk rehearsal procedure may be possible. Other work such as modeling episodic knowledge (Altmann & John, 1999) may also play a critical role in recreating state.

Conclusions

In describing the novice to expert transition, Anderson (1982) posits that task knowledge begins in a declarative form and requires a slow, interpretive process to produce behavior. With practice on the task, the declarative knowledge slowly becomes proceduralized (e.g. converted to production rules) and does not require interpretation. With extensive practice on the task, performance is generated solely by procedural knowledge.

Modeling skill decay along the continuum from novice to expert behavior may require mechanisms that act on both declarative and procedural representation. This work has applied a version of ACT-R's base-level learning mechanism to learned procedural knowledge. The preliminary results of the mechanism on a model of category learning are promising: mean learning rates were slowed to better fit the human data; skill decrement is positively associated with the duration of the retention interval; pre-interval levels of proficiency are quickly attain during retraining. There is, however, much future work: further thought and development of a general (non-architectural) procedure for rehearsing knowledge, and performing a detailed comparison of model behavior against human skill decay data.

Acknowledgements

This work was supported by ONR Grant N000140210039.

References

Altmann, E. M., & John, B. E. (1999). Episodic Indexing: A model of memory for attention events. *Cognitive Science*, 23(2), 117-156.

Anderson, J. R. (1982). Acquisition of cognitive skill. *Psychological Review*, 89, 369-406.

Anderson, J. R. & Lebiere, C. (1998). *Atomic components of thought*. Hillsdale, NJ: Lawrence Erlbaum.

Arthur, W., Bennet, W., Stanush, P. L., & McNelly, T. L. (1998). Factors the Influence Skill Decay and Retention: A Quantitative Review and Analysis. *Human Performance*, 11(1), 57-101.

Chong, R. S. (2003). The addition of an activation and decay mechanism to the Soar architecture. In the *Proceedings of the Fifth International Conference on Cognitive Modeling*, Bamberg, Germany.

Chong, R. S. (1998). *Modeling dual-task performance improvements: Casting executive process knowledge acquisition as strategy refinement*. Doctoral dissertation, The University of Michigan, Ann Arbor, Michigan.

Chong, R. S. & Wray, R. E. (2004, submitted). Inheriting Constraint In Hybrid Cognitive Architectures: Applying the EASE Architecture to Performance and Learning in a Simplified Air-Traffic Control Task. In R. W. Pew & K. Gluck (Eds.) *Modeling Human Behavior with Integrated Cognitive Architectures: Comparison, Evaluation, and Validation*, Hillsdale, NJ:Lawrence Erlbaum.

Forgy, C. (1982). Rete: A fast algorithm for the many patterns/many objects match problem. *Artificial Intelligence*, 19, 17-37.

Gluck, K. A. & Pew, R. W. (2002). The AMBR Model Comparison Project: Round III—Modeling Category Learning. In the *Proceedings of the Twenty-Fourth Annual Conference of the Cognitive Science Society.* Hillsdale, NJ:Lawrence Erlbaum.

Kieras, D.E. & Meyer, D.E. (1997). An overview of the EPIC architecture for cognition and performance with application to human-computer interaction. *Human-Computer Interaction*, 12, 391-438.

Laird, J. E. (2001). It Knows What You're Going To Do: Adding Anticipation to a Quakebot. In the *Proceedings of the Fifth International Conference on Autonomous Agents,* Montreal, Canada.

Laird, J. E., Rosenbloom, P. S., & Newell, A. (1984). Towards chunking as a general learning mechanism. In *Proceedings of AAAI-84*. American Association of Artificial Intelligence.

Miller, C. S., & Laird, J. E. (1996). Accounting for graded performance within a discrete search framework. *Cognitive Science*, 20, 499-537.

Newell, A. (1990). *Unified theories of cognition*. Cambridge, MA: Harvard University Press.

Nosofsky, R. M., Gluck, M. A., Palmeri, T. J., McKinley, S. C., & Glauthier, P. T. (1994). Comparing models of rule-based classification learning: A replication and extension of Shepard, Hovland, and Jenkins (1961). *Memory & Cognition,* 22, 352-369.

Rosenbloom, P. S. & Newell, A. (1993). Learning by chunking: A production-system model of practice. In Rosenbloom, P. S., Laird, J. E., & Newell, A. (Ed.), *The Soar Papers*, Cambridge MA:M.I.T. Press.

Taatgen, N.A. & Lee, F.J. (2003). Production Compilation: A simple mechanism to model Complex Skill Acquisition. *Human Factors*, 45(1), 61-76.

Wray, R. E. & Chong, R. S. (2003). Quantitative explorations of category learning with symbolic concept acquisition. In the *Proceedings of the Fifth International Conference on Cognitive Modeling*, Bamberg, Germany.