Bridging the Lesson Distribution Gap*

David W. Aha[1], Rosina Weber[21], Héctor Muñoz-Avila[31], Leonard A. Breslow[1], Kalyan Moy Gupta[4]
[1]Navy Center for Applied Research in Artificial Intelligence, Naval Research Laboratory, Code 5515
4555 Overlook Avenue, SW, Washington, DC 20375-5337, USA
[2]Department of Computer Science, University of Wyoming, Laramie, WY 82071-3682
[3]Department of Computer Science, University of Maryland, College Park, MD 20742-3255
[4]IIT Industries, AES Division, Alexandria, VA 22303
surname@aic.nrl.navy.mil

Abstract

Many organizations employ lessons learned (LL) processes to collect, analyze, store, and distribute, validated experiential knowledge (lessons) of their members that, when reused, can substantially improve organizational decision processes. Unfortunately, deployed LL systems do not facilitate lesson reuse and fail to bring lessons to the attention of the users when and where they are needed and applicable (i.e., they fail to bridge the *lesson distribution gap*). Our approach for solving this problem, named *monitored distribution*, tightly integrates lesson distribution with these decision processes. We describe a case-based implementation of monitored distribution (ALDS) in a plan authoring tool suite (HICAP). We evaluate its utility in a simulated military planning domain. Our results show that monitored distribution can significantly improve plan evaluation measures for this domain.

1 Introduction

Verified experiential lessons teach improvements about a work practice [Fisher *et al*., 1998]. Many large government (e.g., DOD, DOE, NASA) and private organizations develop *lessons learned* (LL) systems to assist with the knowledge management process of collecting, analyzing, storing, distributing, and reusing lessons [Davenport and Prusak, 1998; Weber *et al*., 2001a]. Lessons record *tacit* experiential knowledge from an organization's employees whose knowledge might be lost when they leave the company, shift projects, retire, or otherwise become unavailable. It is often crucial to record lessons; lives are sometimes saved by preventing recorded catastrophes from recurring [DOE, 1999]. Thus, sharing lessons, even if they are used infrequently, can be *very* important. LL processes and systems are needed to assist with lesson sharing, which can be complicated, especially for large organizations or large lesson databases.

Lessons are usually in unstructured text format, and distribution is commonly supported using standalone text or keyword retrieval tools that require users to "pull" lessons from a repository. Unfortunately, problems with text representations and with this approach to distribution negatively affect lesson reuse, which results in widespread underutilization [Weber et al., 2001a]. In particular, they are responsible for what we term the *lesson distribution gap*. This gap exists when an organization fails to properly promote lesson reuse and available lessons are not deployed when and where they are needed and applicable.

At least three approaches exist to eliminate this gap. First, identified lessons can be incorporated directly into *doctrine*, which defines the processes to be employed by an organization's members. The doctrine is updated to include the knowledge contained in the lesson. For example, the Army's CALL Center [CALL, 2001] deploys teams of lesson analysts and doctrine experts to perform such updates. However, not all lessons can be incorporated into rule-like doctrine (e.g., because they may be true exceptions), and not all organizations have close working relations between doctrine and lessons learned personnel.

A second way to bridge this gap involves "pushing" lessons to potential users, such as via list servers (e.g., [SELLS, 2000]) or intelligent spiders. For example, two of the DOE's sites already employ portals containing spiders [SELLS, 2000]. However, spiders are not integrated with the decision support processes that the lessons target. Thus, after retrieving lessons with a spider, users must characterize the situations for which they are useful, recall them when they encounter an applicable decision support context, and interpret them correctly so that they are properly reused. These are challenging tasks, requiring a high level of expertise and time that most users do not have.

We investigate a third approach to bridging the lesson distribution gap that involves tightly integrating the lesson repository with a decision support tool. Our approach, detailed in Section 2, requires inserting a monitor into the decision support process so that it can determine when a

* Proceedings of IJCAI'01 (Seattle, WA, Aug 2001), Morgan Kaufmann Publishers, Inc., 987-992.

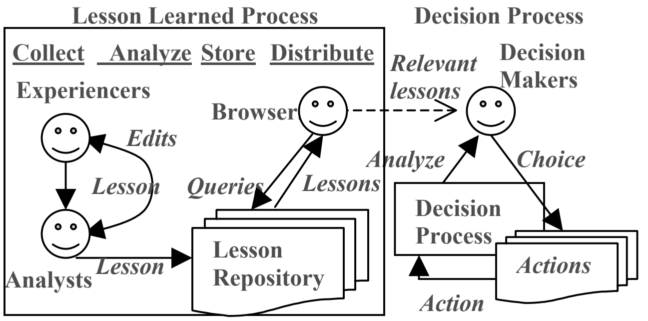

Figure 1. Most lessons learned processes are separated from the decision processes they support.

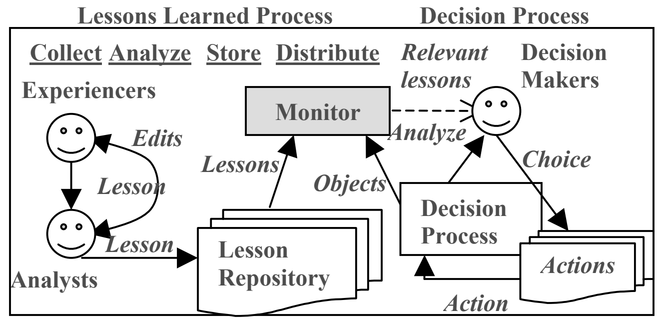

Figure 2. Monitored lesson distribution integrates the lessons learned process with lesson-targeted decision processes.

lesson's conditions are well matched by a decision context. In Section 3, we describe a case-based implementation of this *monitored distribution* approach in ALDS (Active Lesson Delivery System), a module of the HICAP plan authoring tool suite [Muñoz-Avila *et al*., 1999]. We describe ALDS's evaluation in Section 4, where we provide evidence that it can significantly improve performance measures for HICAP-generated plans for a simulated military planning domain (i.e., noncombatant evacuation operations (NEOs)).

Based on a recent survey [Weber *et al*., 2001a] and analysis of the AAAI'00 Workshop on Intelligent Lessons Learned System*s* [Aha and Weber, 2000], we believe that monitored distribution is novel with respect to deployed LL systems, and has great potential for deployment. We discuss the implications of our findings and future research issues in Section 5.

2 Monitored Lessons Learned Processes

A **lesson** is a knowledge artifact that represents a validated (i.e., factually and technically correct) distillation of a person's experience, either positive or negative, that, if reused by others in their organization, could significantly improve a process in that organization. In particular, it identifies a specific design, process, or decision that reduces or eliminates the potential for failures and mishaps, or reinforces a positive result [Secchi *et al*., 1999]. The knowledge management process involving lessons (i.e., the **lessons learned process** (LLP)) implements strategies for collecting, analyzing, storing, distributing, and reusing a repository of lessons to continually support an organization's goals.

LLPs typically target decision-making or execution processes for various types of user groups (i.e., managerial, technical) and organizations (e.g., commercial, military). In this paper, we focus on managing lessons to support *planning* processes.

Flowcharts describing LLPs abound; organizations produce them to communicate how lessons are to be collected, analyzed, and distributed [SELLS, 2000; Fisher *et al*., 1998; Secchi, 1999]. Figure 1 displays a typical LLP, composed of the five sub-processes mentioned above, where reuse does not take place in the same environment as the other sub-processes.

Existing, deployed *LL systems* do not support all processes in a LLP. In particular, organizations typically do not develop software to support verification or reuse. Instead, they use electronic submission forms to facilitate lesson collection, and use a standalone retrieval tool for lesson distribution [Weber *et al*., 2001a]. Users interacting with this standalone tool are expected to browse the stored lessons, studying some that can assist them with their decision-making processes. However, based on our interviews and discussions with members of several LL organizations (e.g., in the Navy, Joint Warfighting Center, Department of Energy, and NASA), and many intended users, we found that they do not use available standalone LL systems, which are usually ineffective because (1) they force users to master a *separate* process from the one they are addressing, and (2) they impose the following unrealistic assumptions:

- Users are convinced that using an LL system is beneficial (e.g., contain relevant lessons).
- Users have the time and skills to successfully retrieve relevant lessons.
- Users can correctly interpret retrieved lessons and apply them successfully.
- Users are reminded of the potential utility of lessons when needed.

We believe that lessons should be shared when and where they are applicable, thus promoting their reuse. This motivated us to develop an architecture for proactive, integrated lesson distribution (Figure 2). In this *monitored distribution* approach, reuse occurs in the same environment as other sub-processes; the decision process and LLP are in the same context. This embedded architecture has the following characteristics/implications:

- The LLP interacts directly with the targeted decision-making processes, and users do not need to know that the LL module exists nor learn how to use it.
- Users perform or plan their decision-making process using a software tool.
- Lessons are brought to the user's attention by an embedded LL module in the decision-making environment of the user's decision support tool.

- A lesson is suggested to the user only if it is applicable to the user's current decision-making task and if its conditions are similar to the current conditions.
- The lesson may be applied automatically to the targeted process.

This process shifts the burden of lesson distribution from a user to the software, but requires an intelligent "monitoring" module to determine whether/when a lesson should be brought to a decision maker's attention.

3 Implementation

We implemented the monitored distribution process in ALDS, a module of HICAP [Muñoz-Avila *et al.*, 1999]. This section details HICAP and then ALDS.

3.1 Plan authoring using HICAP

HICAP (Hierarchical Interactive Case-based Architecture for Planning) is a multi-modal reasoning system that helps users to refine a planning hierarchy [Muñoz-Avila *et al.*, 1999]. A hierarchy is represented as a triple H = $\{T,\prec,:\}$, where T is a set of tasks, $\prec$ defines a (partial) ordering relation on T, and t_1:t_2 means that t_1 is a parent of t_2 in T. Task hierarchies are created in the context of a state **S**=$\{\langle q,a\rangle^+\}$, represented as a set of <question,answer> pairs.

HICAP provides three ways to refine tasks into subtasks. First, it supports manual task decomposition. Second, users can decompose a selected task using HICAP's *interactive* case retriever (NaCoDAE/HTN), which involves iteratively answering prompted questions that refer to state variables. Third, users can select a generative planner (SHOP) to *automatically* decompose *t*.

3.2 Monitored lesson distribution using ALDS

Planning tasks (e.g., for military operations) involve several decisions whose affect on plan performance variables (e.g., execution time) depends on a variety of state variables (e.g., available friendly forces). Without a complete domain theory, HICAP cannot be guaranteed to produce a correct plan for all possible states. However, obtaining a complete domain theory is often difficult, if not impossible. In addition to representing typical experiential knowledge, lessons can help fill gaps in a domain theory so that, when reused appropriately during planning, they can improve plan performance. This is the motivation for applying lessons while using HICAP.

Figure 3 summarizes the behavior of ALDS, the monitored distribution module. ALDS monitors task selections, decompositions, and state conditions to assess similarities between them and the stored lessons. When a stored lesson's applicable decision matches the current decision and its conditions are a good match with the current state, then the lesson is brought to the user's attention to influence decision-making. When a user implements a prompted lesson's task decomposition (i.e., reusing the lesson), the current task hierarchy is modified appropriately.

Abstractly, reusable lessons contain indexing and reuse components. Indexing components include the *target task* and the lesson's applicability *conditions*. The reuse components include a *suggestion* that defines how to reuse an experience and an *explanation* that records how the lesson was learned. This explanation can be used to justify the lesson's use in a new situation. In ALDS, a lesson is indexed by the (target) task that it can modify and a set of <question,answer> pairs defining its applicability conditions, and contains a suggestion (e.g., a task substitution) and the lesson's originating event (i.e., the explanation).

We use a case-based approach for lesson distribution primarily because the indexing components (i.e., task and conditions) must support a partial matching capability. Furthermore, the applicability of a lesson depends on the context of the task that it targets, which suggests using domain-specific similarity functions.

Thus, if both the task and the conditions are a "good" match to the current planning state, then the user should consider decomposing the current task into the lesson's suggested subtasks. We borrowed NaCoDAE/HTN's similarity function for cases, and used a thresholded version to define "good" (i.e., determine when a lesson should be prompted to a user).

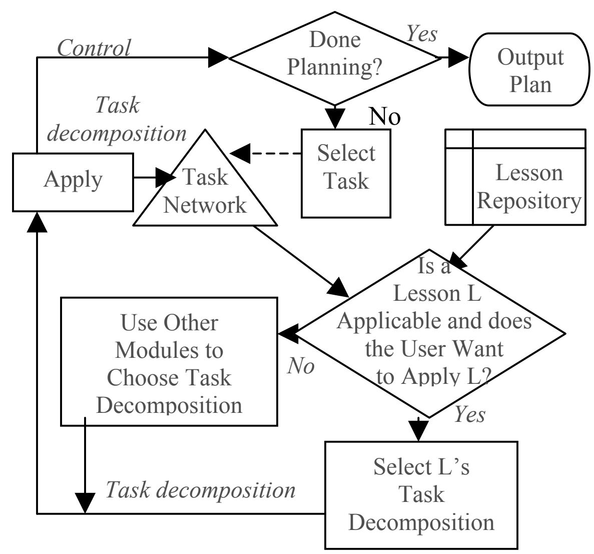

Figure 3. HICAP's lessons distribution sub-process, implemented in ALDS, during plan elaboration.

4 Evaluation

We wanted to evaluate the hypothesis that the monitored distribution approach (e.g., as implemented in ALDS) is superior to the traditional standalone approach for lesson distribution and promoting lesson reuse. For HICAP/ALDS, this hypothesis requires evaluating the plans created by operational users who use the two lesson distribution approaches in repeated planning tasks. Dependent variables would include agreed-upon measures of plan quality, which depend on the planning domain.

Unfortunately, HICAP/ALDS has not yet been scheduled for testing in a military training exercise, which prevents us from working with operational planners. Therefore, we instead performed an evaluation using simulated users on a simulated NEO (noncombatant evacuation operations) domain. Sophisticated full-scale NEO simulators do not yet exist. Therefore, we constructed our own plan evaluator for a simulated NEO domain (Section 4.1). This allowed us to evaluate HICAP/ALDS's plan authoring and lesson distribution capability for an entire plan, rather than be limited to an evaluation on a single task decomposition task [Muñoz-Avila *et al*., 1999].

Simulating how a user might benefit from a standalone lesson distribution tool is difficult. Therefore, we instead compared plan generation when using ALDS vs. not using it (Section 5.2), where our revised hypothesis is that *using lessons will improve plan quality*. This central hypothesis to LLPs, although simple, has ***not*** been previously investigated for lessons learned systems, and thus is appropriate for an initial evaluation focus.

4.1 Methodology

The plans authored by HICAP concerned performing a rescue mission where troops are grouped and move between an initial location (the assembly point) and the NEO site (where the evacuees are located), followed by evacuee relocation to a safe haven. 81 possible routes and 4 means of transportation were encoded. In addition, other conditions were determined during planning such as whether a communications expert was available and the method for processing evacuees. HICAP's plans had 18 steps, and its knowledge base included 6 operators, 22 methods, and 51 cases. We randomly selected 100 initial plan states (12 independent variables) and produced plans for each state with the simulated user interacting with HICAP. This user assigned, through task decomposition, an additional 18 variables (with from one to four values each) for each plan, which required HICAP an average of about 40 seconds to generate. The same set of initial states to produce plans in HICAP was used (to guide task decomposition) both with and without lessons. Each of the two sets of 100 plans (i.e., one set obtained using lessons, and the second set obtained without using lessons) authored by HICAP was input to the evaluator (Section 4.2). Due to the non-deterministic behavior of the evaluator, we executed each plan ten times.

The version of HICAP used in this paper is deterministic; given a state and a top-level goal (i.e., perform a NEO), it will always generate the same plan. A simulated user interacts with HICAP by choosing task decompositions to generate a plan, using the process shown in Figure 3. In NaCoDAE/HTN conversations, it always answers the top-ranking displayed question for which it has an answer, and it answered questions until either none remained unanswered or until one of the solutions exceeded a retrieval threshold, which we set to 50%.

We selected 11 lessons for our experiment, representing a subset of approximately 56 NEO-related lessons from the Active Navy lesson repository (containing 5120 lessons) from the November 2000 copy of the unclassified Navy Lessons Learned System. These were selected according to their relevance to NEOs and their clarity, so that we could recognize their relation to the plans authored using HICAP. For example, one lesson was defined as:

Task: Standard Medical Inventory
- Applicability Conditions: (<q,a> pairs)
- Is the medical inventory of standard size or is it standard minus 1/3? Yes
- Is the climate tropical? Yes

Suggestion: Add 1/3 to the medical inventory

4.2 Plan evaluation

We built a stochastic evaluator for NEO plans that take into account general knowledge of the NEO domain and computes the performance measures (described below). This evaluator is not a simulator because it does not use specific distributions for each type of event, but simply computes, according to a uniform distribution, what are the expected consequences of some choices in building a plan (i.e., the causal chain of events that are generated by these choices will influence each of the dependent variables differently). We built the evaluator and the HICAP knowledge base for mock NEOs based on available applicable lessons.

We defined plan quality based on official measures of NEOs, which are planning domain dependent. These measures are defined in the Universal Naval Task List [UNTL, 1996] under measures of performance suggested for Joint and Naval tasks. These measures primarily concern execution duration and casualty rates. To avoid a redundant evaluation, we have selected one measure for total duration of the operation, one for duration until evacuees receive medical assistance, and the percentage of casualties among evacuees, friendly forces, and enemies. These summarize the most important aspects suggested in the UNTL.

We defined bounds for variables based on actual NEOs. For example, we limited the percentage of casualties that occurred after a severe enemy attack takes place. Enemy attacks will only be possible in two planning segments (out of a total of five segments) and their likelihood increases when users choose land transportation and decreases when weather is troublesome. There is a small chance of a crash when helicopters are used that increases if the weather is not favorable; the resulting number of casualties is proportional to the number of passengers in each aircraft. A long planning segment flown by helicopter will have added the time and risks associated with in-flight refueling (e.g., [Siegel, 1991]).

4.3 Results

As summarized in Table 1, ALDS using lessons substantially improved the first four of five performance variables. A brief examination of the results (i.e., the first run for each of the 100 plans), using a standard student's t test, revealed significant differences for both overall duration (p<0.1, t=1.60, df=99) and duration until medical assistance arrived (p<0.1, t=1.39). All lessons were used in generated plans, and an average of approximately three lessons were used per plan.

Table 1. Experimental results with the 100 plans.

	Without lessons	With lessons	% Reduction with lessons
mean duration	39h50	32h48	18
s.d.	16h51	16h12	-
mean duration until medical asst.	29h37	24h13	18
s.d.	11h13	10h26	-
mean % casualties: to evacuees	11.48	8.69	24
to friendly forces	9.41	6.57	30
to enemies	3.08	3.14	-2

The significance of an overall reduction of 24% in the percentage of casualties among evacuees was estimated in each plan based on the parameter *number of evacuees*, which was randomly set to *dozens*, *hundreds*, or *thousands*. Based on the number of evacuees selected for these simulated NEOs, using the lessons reduced the average number of casualties by **24**, from 100 to 76.

These results suggest that the monitored distribution approach can potentially generate better plans for realistic problem domains (e.g., planning for NEO operations). However, the experimental conditions were designed so that lessons were available for a reasonable percentage of the generated plans, and thus could be prompted to the simulated HICAP user so that, when applied, they could improve plan quality (with high probability). Nonetheless, we expect that similar improvements may yield benefits in plans for domains where safety issues and speed are paramount to success.

The capabilities of certain learning algorithms can be evaluated by varying dataset characteristics to determine when certain learning algorithms can be expected to perform well (e.g., [Aha, 1992]). Similarly, we plan to characterize the set of experimental conditions for which ALDS can use lessons to significantly improve plan evaluation performance measures.

5 Discussion

This paper proposes a technology (i.e., case-based reasoning) solution to part of a knowledge management (KM) problem (i.e., managing lessons learned). However, KM problems typically require challenging organizational dynamics issues, and these require precedence in the context of bridging the lesson distribution gap. Thus, monitored distribution can at most play only one part of a much larger solution.

Our evaluation of ALDS demonstrates how monitored distribution, when embedded in a decision-making (i.e., planning) process, can improve the results of that process. Although we used simulated users in our experiments to reduce human biases during the evaluation, we stress that this is a mixed-initiative approach, where humans interact with HICAP to generate plans. The unique aspect of ALDS is that it allows users to execute a lesson's suggestion (i.e., here, a task substitution), rather than limit them to simply browsing the suggestion.

HICAP's NaCoDAE/HTN module manipulates *cases* that represent task decompositions corresponding to either standard operating procedures or decompositions that were derived from decision making during training exercises and actual operations. In contrast, ALDS manipulates *lessons* that capture experiences that, if reused, can significantly improve the performance of subsequent plans. Unlike cases, lessons are not conceptually limited to representing task decompositions, but can be used to apply edits to *any* of HICAP's objects (e.g., resource assignments, resources, task durations).

Several workshops (e.g., organized by the Department of Energy, the European Space Agency, the Joint Warfighting Center, and each branch of the armed services) have now taken place on the topic of lessons learned. However, few efforts on lessons learned systems have examined the potential utility of AI (e.g., Vandeville and Shaikh [1999] briefly mention using fuzzy set theory to analyze elicited lessons), and there is a lack of closely related work to monitored distribution. However, one recent workshop brought attention to this area from an AI perspective [Aha and Weber, 2000], and a few of its contributors touched on issues related to proactive lesson distribution. For example, Leake *et al.*'s [2000] CALVIN system implements a task-oriented LLP that collects lessons about research topics and research results with an active distribution sub-process. Like ALDS, CALVIN prompts users with suggestions (i.e., alternative WWW pages to browse) that can be immediately executed. However, while CALVIN focuses on a diagnosis task, ALDS operates in the context of a synthesis task (i.e., planning), and can potentially update any of the planning scenario's objects. Like both of these systems, Watson [2000] also describes a case retrieval system, in this case for extending Cool Air to distribute trouble tickets. However, Cool Air does not operate in a mixed-initiative setting. Some KM approaches [Reimer *et al.*, 2000; Abecker *et al.*, 2000] also target distribution in the context of organizational knowledge, but use formats that do not support indexing.

Several limitations exist concerning our approach and its implementation in ALDS. For example, lessons can be complex, and suggest changes to a variety of objects in the planning scenario. Although HICAP represents several such objects (e.g., resources, resource assignments), it is currently limited to processing only task substitution lessons. In future implementations of HICAP and ALDS, lessons will be able to represent suggestions that, when applied, will not be limited to task substitution. For example, a lesson might suggest a task decomposition, or using an alternative resource assignment for a given task, recommend changing some temporal orderings of tasks, or suggest edits to any of the objects used by HICAP to define plans.

In addition, to be useful, our approach assumes that the decision processes targeted by the lessons are managed by a software tool, thus allowing integration with ALDS. Furthermore, our approach requires identifying each lesson's indexing and reuse components, which requires significant knowledge engineering effort. We are currently developing lesson collection tools that reduce this effort. Weber *et al.* [2001b] describe interactive elicitation approaches that use taxonomies to guide lesson collection to populate ALDS's lesson repository.

In future work, we will conduct subject experiments that compare the monitored distribution approach vs. traditional keyword search tools for lesson distribution. We will also demonstrate how monitored distribution is not restricted to planning tasks.

6 Summary

We identified a problem with distributing lessons, called the l*esson distribution gap*, which is crucial to many lessons learned organizations. To address this problem, we introduced an approach called *monitored distribution*, which is characterized by a tight integration with a decision support tool that manages processes that the lessons can potentially improve. We implemented this approach in ALDS, a case retrieval system, and evaluated its capability in the context of a module for HICAP, a plan authoring tool. Our experiments with a simulated military planning domain (i.e., for noncombatant evacuation operations) showed that, by using lessons, monitored distribution can help to significantly improve plan performance measures. In summary, we demonstrated a technology that brings lessons to the attention of users when and where they are needed and applicable.

Acknowledgements

This research was supported by the Office of Naval Research and DARPA.

References

[Abecker *et al*., 2000] Abecker, A.; Bernardi, A.; Hinkelmann, K.; Kuehn, O.; and Sintek, M. Context-Aware, Proactive Delivery of Task-Specific Information: The KnowMore Project. Information Systems Frontiers 2:3/4, 253-276, 2000.

[Aha, 1992] Aha, D.W. Generalizing case studies: A case study. *Proceedings of the Ninth International Conference on Machine Learning* (pp. 1-10). Aberdeen: Morgan Kaufmann, 1992.

[Aha and Weber, 2000] Aha, D.W., and Weber, R. (Eds.). *Intelligent Lessons Learned Systems: Papers from the AAAI Workshop* (Technical Report WS-00-03). Menlo Park, CA: AAAI Press, 2000.

[CALL, 2001] Center for Army lessons learned: Virtual research library. http://call.army.mil.

[Davenport and Prisak, 1998] Davenport, T.H., and Prusak, L. *Working knowledge: How organizations manage what they know.* Boston, MA: Harvard Business School Press, 1998.

[DOE, 1999] DOE *The DOE corporate lessons learned program* (Technical Report DOE-STD-7501-99). Washington, DC: U.S. Department of Energy.

[Fisher *et al*., 1998] Fisher, D., Deshpande, S., & Livingston, J. *Modeling the lessons learned process* (Research Report 123-11). Albequerque, NM: The University of New Mexico, Department of Civil Engineering, 1998.

[Leake *et al.,* 2000] Leake, D.B., Bauer, T., Maguitman, A., and Wilson, D.C. Capture, storage, and reuse of lessons about information resources: Supporting task-based information search. In (Aha & Weber, 2000).

[Muñoz-Avila *et al*., 1999] Muñoz-Avila, H., Aha, D.W., Breslow, L.A., & Nau, D. HICAP: An interactive case-based planning architecture and its application to NEOs. *Proceedings of the Eleventh Conference on Innovative Applications of Artificial Intelligence* (pp. 870-875). Orlando, FL: AAAI Press, 1999.

[Reimer *et al*, 2000] Reimer, U., Margelisch, A., Staudt, M. A Knowledge-based Approach to Support Business Processes, In: Proceedings of the AAAI 2000 Spring Symposium Series: Bringing Knowledge to Business Processes, Stanford, CA, March 2000.

[Secchi *et al*., 1999] Secchi, P.; Ciaschi, R.; Spence, D. The ESA alert system. In P. Secchi (Ed.) *Proceedings of Alerts and Lessons Learned: An Effective way to prevent failures and problems* (Technical Report WPP-167). Noordwijk, The Netherlands: ESTEC, 1999.

[SELLS, 2000] SELLS *Proceedings of the SELLS Fall Meeting.* [tis.eh.doe.gov/ll/proceedings/proceedings1000.htm] [www.estec.esa.nl/CONFANNOUN/99c06], Fall, 2000.

[Siegel, 1991] Siegel, A. Eastern Exit: The Noncombatant Evacuation Operation (NEO) from Mogadishu, Somalia, in January 1991. Center for Naval Analyses, 1991.

[UNTL, 1996] UNTL *Universal Naval Task List* (OPNAVINST 3500.38). Arlington, VA: Navy Modeling and Simulation Office, September, 1996.

[Vandeville and Shaikh, 999] deville, J.V. and Shaikh, M. A. A structured approximate reasoning-based approach for gathering "lessons learned" information from system development projects. *Systems Engineering, 2*(4), 242-247, 1999.

[Watson, 2000] Watson, I. Lessons learned during HVAC installation. In (Aha & Weber, 2000).

[Weber *et al*., 2001a] Weber, R., Aha, D.W., and Becerra-Fernandez, I. Intelligent lessons learned systems. To appear in *Expert Systems with Applications*, *20*(1), 2001.

[Weber *et al.,* 2001b] Weber, R., Aha, D.W., Sandhu, N., & Muñoz-Avila, H. (2001). A Textual Case-Based Reasoning Framework for Knowledge Management Applications. Professionelles Wissenmanagement Erfahrungen und Visionen, 244-253. Aachen:Shaker Verlag.