Active Delivery for Lessons Learned Systems[1]

Rosina Weber[1,2], David W. Aha[2], Hector Muñoz-Ávila[2,3], Leonard A. Breslow[2]

[1]Department of Computer Science, University of Wyoming, Laramie, WY 82071-3682
[2]Navy Center for Applied Research in Artificial Intelligence
Naval Research Laboratory, Washington, DC 20375
[3]Department of Computer Science, University of Maryland, College Park, MD 20742-3255
lastname@aic.nrl.navy.mil

Abstract. Lessons learned processes, and software systems that support them, have been developed by many organizations (e.g., all USA military branches, NASA, several Department of Energy organizations, the Construction Industry Institute). Their purpose is to promote the dissemination of knowledge gained from the experiences of an organization's employees. Unfortunately, lessons learned systems are usually ineffective because they invariably *introduce* new processes when, instead, they should be *embedded* into the processes that they are meant to improve. We developed an embedded case-based approach for lesson dissemination and reuse that brings lessons to the attention of users rather than requiring them to fetch lessons from a standalone software tool. We demonstrate this active lessons delivery architecture in the context of HICAP, a decision support tool for plan authoring. We also show the potential of active lessons delivery to increase plan quality for a new travel domain.

1 Introduction

Lessons learned (LL) are validated working knowledge derived from successes or failures that, when reused, can significantly impact an organization's processes (Secchi, 1999). *LL processes* refer to an organization's efforts for managing lessons learned. Hundreds of organizations have or plan to develop LL processes with the goal of enhancing job safety, reducing costs, improving quality, and/or increasing problem-solving speed.

LL systems (i.e., software for supporting LL processes) have been implemented in military organizations since the mid 1980s and in government and commercial organizations during the 1990s as knowledge management (KM) (Davenport & Prusak, 1998) solutions for documenting tacit lessons whose reusable content can benefit targeted organizational processes. Surveyed organizations typically address five LL sub-processes: collect, verify, store, disseminate, and reuse (Weber et al., 2000a). LL systems have been used to support collection (e.g., online lesson submission forms) and dissemination (e.g., standalone retrieval tools). Unfortunately, standalone dissemination systems do not reflect a central KM tenet: *to be successful,*

[1] In E. Blanzieri and L. portinale (Eds.) Advances in Case-Based Reasoning, pp. 322-334, 5th European Workshop, EWCBR2K. Trento, Italy:Springer-Verlag.

knowledge dissemination methods must be embedded in the processes they are intended to support (Reimer, 1998; Aha et al. 1999; Leake et al., 2000). Thus, they are typically underutilized and do not promote knowledge sharing and reuse.

Previously deployed LL systems for lesson dissemination have not used AI approaches, except for RECALL (Sary and Mackey, 1995), which uses conversational case retrieval to retrieve lessons. However, RECALL did not employ a knowledge representation that was optimized to promote lesson reuse, and it is a standalone LL tool. We address issues concerning lesson representation in (Weber et al., 2000c). This paper addresses how AI approaches can be used to develop embedded software architectures for LL systems. In particular, we introduce an embedded case-based architecture for LL *dissemination* and *reuse* in HICAP (Muñoz-Avila et al., 1999), a multi-modal plan authoring tool, and demonstrate its utility in an evaluation on a new travel planning domain. We begin by summarizing related work in Section 2 and introducing LL processes and systems in Section 3. Section 4 then describes HICAP and introduces our embedded active lessons delivery module. Section 5 describes an initial empirical study. Finally, we discuss implications and future needs in Section 6.

2 Related Work

The most ambitious investigation of LL processes was performed by the Construction Industry Institute's (CII's) Modeling Lessons Learned Research Team (Fisher et al., 1998). They surveyed 2400 organizations, characterized the 145 initial responses as describing 50 distinct LL processes, and performed follow-up, detailed investigations with 25 organizations. They found strong evidence that most organizations were using insufficient dissemination processes.

Secchi et al. (1999) found that only 4 of the 40 organizations who responded to their survey used software to support their LL process. In both surveys, none of the responding organizations implemented an active LL process for lesson dissemination, probably because software was not used to control the process(es) targeted by the lessons, or elicited lessons were immediately/manually incorporated into the targeted process (e.g., into the organization's best practice manuals, or by requiring project members to read through project-relevant lessons prior to initiating a new project).

Similarly, contributions at previous workshops on LL processes (SELLS, 1999; Secchi, 1999) were limited to standalone systems, and few authors have discussed using artificial intelligence (AI) to assist LL processes (e.g., Vandeville and Shaikh (1999) briefly mention using fuzzy set theory to analyze elicited lessons). The AAAI 2000 Workshop on *Intelligent Lessons Learned Systems* (Aha & Weber, 2000) will be the first workshop to specifically promote AI contributions to this topic.

We concluded from our own survey (Weber et al., 2000a)[2] that no deployed LL system uses an embedded architecture for lesson dissemination, and only three such architectures for LL and related systems have been proposed: ACPA (Johnson et al., 2000), ALDS (Weber et al., 2000f), and CALVIN (Leake et al., 2000). These architectures each employ case retrieval. However, ACPA uses a form of task-model

[2] Please see www.aic.nrl.navy.mil/~aha/lessons for pointers to on-line lessons learned systems.

tracking to prompt users with corporate stories, and CALVIN collects task-specific lessons for searching information sources. In contrast, ALDS focuses on disseminating and reusing organization-specific lessons for multiple tasks in planning domains, and is the first LL dissemination architecture that supports executable lesson reuse. This paper discusses the implementation and initial analysis of ALDS in HICAP.

In planning contexts, LL dissemination sub-processes can be used to adapt plans by applying lessons. This is related to using cases to guide case adaptation (Leake & Kinley, 1998). Previous such approaches explicitly recorded case adaptation cases and made them available in subsequent adaptation episodes. In our work, lessons are first recorded from experiences in executing plans (e.g., in military operations), then are checked in a systematic validation process by LL experts, and those that are accepted are finally made available for modifying future planning efforts. Therefore, unlike adaptation cases, lessons are not restricted to modifying case solutions, but instead are intended to modify existing organizational (e.g., planning) processes. Also, our approach is immersed in a decision support environment, in which users can optionally ignore or amend a lesson's recommendation.

3 Lessons, Processes, and Systems

The following definition has been adopted by NASA and European Space Agencies:

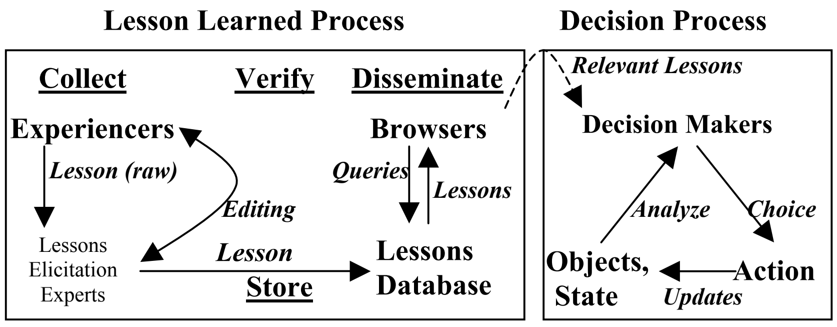
> *"A **lesson learned** is a knowledge or understanding gained by experience. The experience may be positive, as in a successful test or mission, or negative, as in a mishap or failure. Successes are also considered sources of lessons learned. A lesson must be significant in that it has a real or assumed impact on operations; valid in that is factually and technically correct; and applicable in that 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)

This definition highlights several criteria for lessons learned: they are knowledge artifacts, obtained from either positive or negative experiences, are validated for correctness, and, when reused, can significantly impact an organization's processes.

A **LL process** implements a strategy to collect, verify, store, disseminate, and reuse lessons to continually support an organization's goals. Thus, it defines how to use *tacit* experiential knowledge in an organization's activities, capturing the experiential knowledge from employees whose knowledge might be lost when they leave the company, shift projects, retire, or otherwise become unavailable. LL processes 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 LL processes abound; organizations produce them to communicate how lessons are to be acquired, verified, and disseminated (SELLS, 1999; Fisher et al., 1998; Secchi, 1999). Figure 1 displays a typical LL process, composed of five sub-processes (i.e., collect, verify, store, disseminate, and reuse), where reuse does not take place in the same environment as the other sub-processes.

Existing, deployed *LL systems* do not support all LL processes. 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 tool to support lesson dissemination (Weber et al., 2000a). Users interacting with this standalone tool are expected to browse the stored lessons, studying some that can assist them with their decision-making process(es). However, based on our interviews and discussions with members of several LL organizations (e.g., at the Joint Warfighting Center, the Department of Energy, NASA's Goddard Space Flight Center, the Navy Facilities Engineering Command, CII), and many intended users of LL systems, we concluded that users do not use available standalone LL systems, which are usually ineffective because they force users to master a *separate* process from the one they are addressing, and impose the following unrealistic assumptions:



1. Users are convinced that using LL systems is beneficial, and can find them.
2. Users have the time and skills to successfully retrieve relevant lessons.
3. Users can correctly interpret retrieved lessons and apply them successfully.
4. Users are reminded of the potential utility of lessons when needed.

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

This motivated us to develop an *active lessons delivery* architecture for lesson dissemination (Figure 2). In this approach, reuse occurs in the same environment as other sub-processes; the decision process and lessons learned process are in the same context. This embedded architecture has the following characteristics/implications:

1. The LL process 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.
2. Users perform or plan their decision-making process using a software tool.
3. 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.
4. 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.
5. The lesson may be applied automatically to the targeted process.

This process shifts the burden of lesson dissemination from a user to the software, but requires intelligent software modules and systematic knowledge engineering efforts.

4 A case-based approach for active lessons delivery

This section details a case-based approach for active lessons delivery, which we have implemented in a module of HICAP (Muñoz-Avila et al., 1999). Sections 4.1 and 4.2 detail HICAP and its active lessons delivery module, respectively.

4.1 Plan authoring in HICAP

HICAP (Hierarchical Interactive Case-based Architecture for Planning) is a multi-modal reasoning system that helps users to formulate a task hierarchy, which is represented as a triple $H = \{T,<,\wedge\}$, where each task $t \in T$ is defined by its name t_n, $<$ defines a (partial) ordering relation on tasks, and $t_1 \wedge t_2$ means that t_1 is a parent of t_2 in T. Task hierarchies are created in the context of a state $\mathbf{S}=\{<q,a>^+\}$, represented as a set of question/answer pairs.

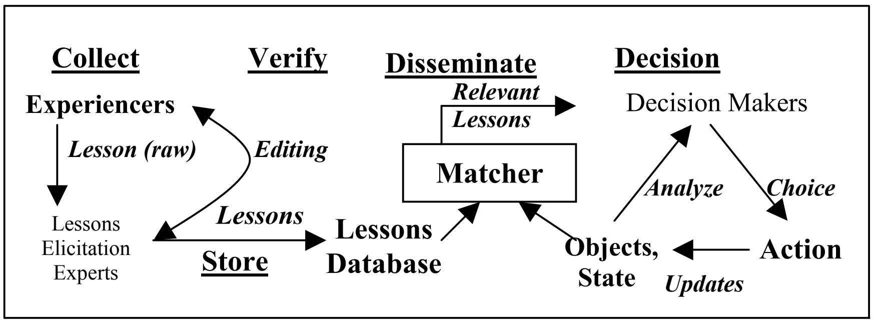


Fig. 2. Active lessons delivery for lesson dissemination integrated with the decision process.

Although HICAP manipulates several objects (e.g., resources) and decisions (e.g., resolve a resource conflict), we focus this first implementation of ALDS on task decomposition. HICAP provides users with three ways to decompose tasks into subtasks. First, it supports manual task decomposition. Second, users can select a task *t* to decompose and use HICAP's conversational case retriever (NaCoDAE/HTN) to *interactively* select a stored task decomposition for t, assuming that specific cases exist for decomposing t. This involves a conversational interface, where the user sees displays of top-ranking solutions and unanswered questions, can iteratively answer any of the displayed questions (which adds a <q,a> pair to **S** and updates the two displays), and can end the conversation by selecting a displayed solution (i.e., a task decomposition). Third, users can select a generative planner (SHOP) to *automatically* decompose t, assuming methods or operators exist for decomposing t. We describe the tight integration of SHOP with NaCoDAE/HTN, in the SiN algorithm, in (Muñoz-Avila et al., 2000).

4.2 A case-based active lessons delivery module

Planning tasks, even everyday ones like those for planning how to travel between two locations, can involve several decisions whose effect on plan performance variables (e.g., time, cost) depends on a variety of state variables (e.g., weather, amount of luggage). Without a complete domain theory, HICAP cannot guarantee it will produce a correct plan for all possible states. But obtaining a complete domain theory is often difficult, if not impossible. Besides 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 the active lessons delivery module, which 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.

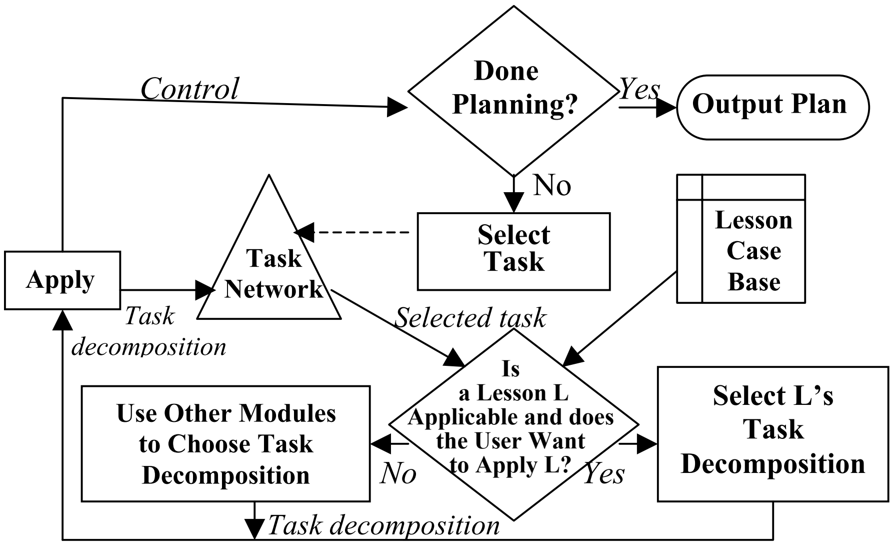


Fig. 3. HICAP's lessons delivery sub-process during plan elaboration.

This first implementation of ALDS incorporates a simple strategy. First, instead of representing lessons using all six of the characteristics of an idealized lesson representation (Weber et al., 2000c), we instead borrowed the representation used by NaCoDAE/HTN for task decomposition cases. That is, a lesson is indexed by the *originating task* that it can decompose and a set of <question, answer> pairs defining its *conditions*, and contains a *suggestion* (e.g., a task decomposition). Thus, if 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.

Second, we borrowed NaCoDAE/HTN's similarity function for cases, and used a thresholded version to determine when a lesson should be prompted to a user.

Finally, we only supported one of many possible lesson suggestions (i.e., task decomposition). In future implementations, 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 other edits to any of the objects used by HICAP to define plans.

5 Empirical study

Our hypothesis is that the ALDS architecture is superior to the traditional standalone architecture for supporting lesson dissemination in planning tasks. That is, we claim our architecture is better suited for promoting lesson reuse. This hypothesis is difficult to evaluate; it requires evaluating the plans created by operational users who use these two approaches in repeated planning tasks. Dependent variables would primarily include agreed-upon measures of plan quality, which would be planning domain dependent. Unfortunately, at this time, we do not have access to sufficient users nor realistic plan evaluation simulators (e.g., although our group used ModSAF to evaluate plans generated by HICAP for noncombatant evacuation operations (NEOs) in (Muñoz-Avila et al., 1999), the simulator could only be used for a single subtask of an entire NEO plan.). Therefore, we instead focused our experiment on simulated users in a novel travel planning domain for which we could create an adequate simulator (Section 5.1). Also, because it's difficult to simulate how a user might benefit from a standalone lesson dissemination tool, we instead compared plan generation when using the active lessons delivery module vs. not using it (Section 5.2), where our hypothesis is that *using lessons will improve plan quality*.

5.1 Personal travel domain

This domain was inspired by DARPA's Active Templates program, which concerns (in part) the development of inferencing tools for travel plan generation. We developed a knowledge base and plan evaluator for this domain, in which the objective was to create a plan to travel from one of ten locations in the Washington, DC metropolitan area (e.g., Adam's Morgan, Bethesda, DuPont Circle) to downtown New York City. We represented seven transportation methods, including four inter-city and three intra-city methods: {airplane, city bus, inter-city bus, shuttle van, subway, taxi, train}. Plans have 3-5 segments, while plan hierarchies have exactly four levels. This defines a space of approximately 756 possible plans. A travel *state* is defined by up to eight variables, represented as questions (Table 1).

Table 1. Questions used to define personal travel states.

Question	Possible Answers
Is it raining hard?	Y/N
Is there any major accident or unusual event?	Y/N
What is the chance of large snow accumulation?	High, moderate, low
How many passengers?	1,2,3,3+
Is there any luggage to check?	Y/N
Are any children or seniors involved?	Y/N
Is travel occurring on a holiday? If so, which one?	Holiday name, or No
What is the starting day of the week for this trip?	7 distinct answers

Our hand-coded knowledge base consists of ten methods and one (movement) operator for HICAP's JSHOP module, and 40 cases for NaCoDAE/HTN. Each plan's task decompositions were obtained by applying 4-6 methods, 3-5 operators (i.e., the same operator 3-5 times), and 3-5 cases. An example plan is: take(taxi,Bethesda, BWI), take(plane, BWI, JFK), take(shuttle, JFK, Downtown NYC).

Head	**Head**
travelCity(Penn Station, DowntnNYC)	travelCity (taxi, Penn Station, DowntnNYC)
Questions	**Questions**
Is it raining hard? **No**	Is it raining hard? **Yes**
How many passengers? **2**	How many passengers? **2**
Subtasks	**Subtasks**
take(taxi, Penn Station, DowntnNYC)	take(subway, Penn Station, DowntnNYC)

Fig. 4. Example case (left) and lesson (right) from the personal travel domain.

The version of HICAP used in this paper is deterministic; given a state and a top-level goal (i.e., start and destination locations), it will always generate the same plan. However, our personal travel plan evaluator is non-deterministic; it can generate a different time duration for each of a given plan's segments each time the plan is executed. The evaluator is given a plan and the current world state. For each run, it outputs whether the plan succeeded and, if it did succeed, how long the trip took as well as its cost. We relied on interviews with domain experts for information on costs and expected time duration. A plan will fail if either a huge delay occurs (e.g., due to an unusual event) that causes a plan cancellation, or when segment delays cause a late arrival for a segment requiring a fixed time departure (e.g., an airplane flight). For each segment, we applied an exponential delay function that is influenced by world state conditions. For example, a flight segment will incur a longer delay for higher chances of large snow accumulation, especially on holidays (i.e., high travel days). Segments are categorized into short, medium, and long lengths, and delays can range from 0 up to 4.5 times a segment's anticipated duration before a plan is cancelled.

Figure 4 depicts an example case from the personal travel domain and a lesson that modifies it. In the case, a taxi is taken from Penn Station to downtown New York City when there is no rain and the number of passengers is 2. The lesson modifies this case by indicating that, when it is raining, the subway should be taken instead. (It is well known that it is almost impossible to find a cab when it rains in New York City.)

5.2 Experiments: Methodology and results

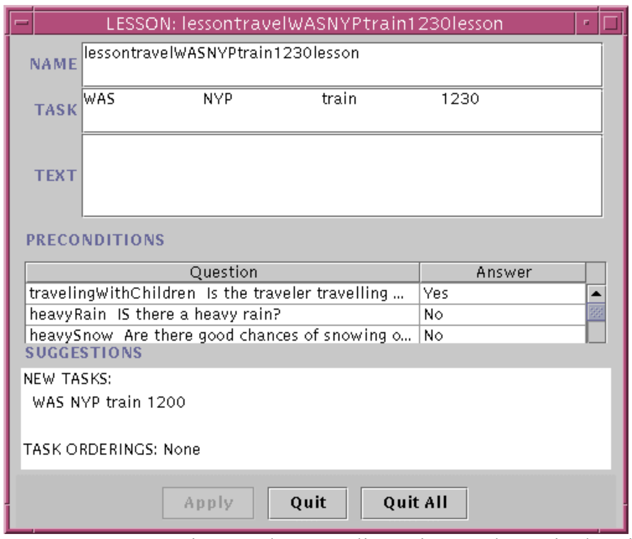
We explored whether a case-based active lessons delivery strategy for this travel domain can improve plan performance, as measured by success frequency and mean time duration, without increasing travel cost.

We generated lessons by analyzing some problem results. We define a ***goal*** to be a pair of <start, destination> locations and a ***problem*** as a pair of <goal, world state>. We selected ten goals, corresponding to the ten ways in which travel could take place between Washington, DC and NYC. For each goal, we generated 10 random world states (i.e., providing answers to *all* 8 questions), thus yielding 100 total problems. HICAP was then used to generate a plan for each problem. Then, for each of the 100 <plan, world state> pairs, the plan evaluator was run 10 times. For each run, we measured whether the plan executed to completion, the plan's (simulated) duration, and the number of lessons used to create it.

We then used the following three manual procedures, five times each, to create lessons. These procedures were designed to reduce human biases in lesson creation.

1. *Plan failure lessons*: Select a problem g_p (with plan p) whose success rate was lowest for a randomly selected goal g's problems. Then try to reduce its trip duration as follows. Locate p's segment p_f where failure most frequently occurred, and replace a segment preceding it with a faster transportation method. If this increased trip cost more than 15%, then also replace the transportation method of a segment in p following p_f with a cheaper transportation method. However, if this could not account for the cost increase, then abandon trying to reduce trip duration and instead simply start plan p earlier by the mean time that the traveler was late for p_f.
2. *Positive (short) duration lessons*: Randomly select a goal g and its subset of problems P whose success rates were at least 50%. Select problems g_{short} and g_{ave} from P whose mean duration were shortest and most similar to the average, respectively. Let t_{short} and t_{ave} be the starting times of the inter-city plan segments p_{short} and p_{ave} in the plans of these two problems. Let $\Delta = t_{ave} - t_{short}$. If the mean time that the traveler waited for p_{ave}, minus the s.d. for this problem, is greater than Δ, then substitute p_{short} for p_{ave} in the plan for g_{ave}.
3. *Negative (long) duration lessons:* Same as (3), but focusing on g_{ave} and g_{long}. Figure 5 displays one of the *plan duration* lessons.

Each lesson was encoded with 8 <question, answer> pairs, corresponding to the world state during the experience from which it was derived. These procedures produced 15 lessons. We tested HICAP's performance with vs. without lessons by examining whether the lessons could improve plan performance on the 100 problems from which they were derived.


LESSON: lessontravelWASNYPtrain1230lesson

NAME: lessontravelWASNYPtrain1230lesson

TASK: WAS NYP train 1230

TEXT:

PRECONDITIONS

Question	Answer
travelingWithChildren Is the traveler travelling ...	Yes
heavyRain IS there a heavy rain?	No
heavySnow Are there good chances of snowing o...	No

SUGGESTIONS

NEW TASKS:
WAS NYP train 1200

TASK ORDERINGS: None

Fig. 5. Lesson suggesting to take an earlier train to reduce trip duration.

We implemented a simulated HICAP user that uses the process shown in Figure 3 and the following behavior during NaCoDAE/HTN conversations:

- It always answers the top-ranking displayed question for which it has an answer.
- It answered questions until either none remained unanswered or one of the solutions exceeded a retrieval threshold, which we set to 50%.

The results of this experiment are summarized in Table 2. In the training set, plans generated by HICAP with the case-based active lessons delivery module reduced average trip duration by over 30 minutes at the same price level, and even improved the success rate. This improvement demonstrates the potential benefit of lesson reuse because it changed the relation between trip duration and success rate (*i.e.*, we always increase the chance of success if we depart earlier).

These results suggest that the active lessons delivery approach can potentially generate better plans for realistic problem domains (e.g., planning for NEO operations). Similar improvements may yield improvements in plans for domains where safety issues and speed are paramount to success.

Table 2. Experimental results with the 100 problems.

#Lessons	Success Rate	Mean (s.d.) Trip Duration	Lesson use rate	Average Price
0	70%	9h45min (3h14min)	0%	$ 221
20	71%	9h14min (3h35min)	31%	$ 220

6 Discussion

Our evaluation of the active lessons delivery module demonstrates how this approach, 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 in which humans interact with HICAP to generate plans. Perhaps the most unique aspect of our approach, ALDS, is that it allows users to execute a lesson's suggestion (i.e., here, a task decomposition), rather than limit them to browsing the suggestion.

We are currently investigating the lesson selection heuristics and their capability in generating useful lessons. In future research, we intend to explore the utility of lessons that are not restricted to task decomposition recommendations. We will also return to the NEO planning and/or alternative military planning domains as we investigate the utility of lessons in more applied domains.

Conceptually, HICAP's NaCoDAE/HTN module uses cases that represent task decompositions corresponding to either (generalized) standard operating procedures or decompositions that were derived from executing plans. These cases are limited to one purpose: updating the task hierarchy. In contrast, the ALDS module in HICAP uses lessons that capture experiences that, if reused, can significantly impact the performance of subsequent plans. Importantly, lessons are not 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). In summary, although we have implemented lessons as cases in this implementation of ALDS, it was only for convenience; lessons have a more general scope than task decomposition.

7 Conclusion

We introduced a case-based active lessons delivery approach for a lessons learned dissemination sub-process. In comparison to traditional standalone systems, active delivery systems bring lessons to a user's attention rather than requiring them to consult a separate retrieval process, which can be problematic. In the context of a HICAP module, we showed that, for a personal travel domain, using lessons can increase plan quality (e.g., reduce mean travel duration with the same level of cost and success).

Our research goals include developing a more mature version of this module and evaluating it on a more realistic planning domain (e.g., noncombatant evacuation operations). We are currently pursing potential transitions of HICAP, including its active lessons delivery module, to deployed (e.g., military) systems, and have begun working with the Joint Warfighting Center towards this goal. We have also begun to develop strategies for systems that use HICAP to assist with lesson collection, and plan to evaluate these strategies in our future efforts.

Acknowledgements

This research was supported by grants from the Office of Naval Research and the Naval Research Laboratory. The travel domain was inspired by Maj. Doug Dyer's presentations for the Active Templates Program. Finally, thanks to Karen Spearman for assisting with developing the travel domain, and for our reviewers' suggestions.

References

Aha, D.W., Becerra-Fernandez, I., Maurer, F., and Muñoz-Avila, H. (Eds.). 1999. Exploring Synergies of Knowledge Management and Case-Based Reasoning: Papers from the AAAI 1999 Workshop (Technical Report WS-99-10). Menlo Park, CA: AAAI Press.

Aha, D.W., & Weber, R. (2000). *Intelligent Lessons Learned Systems: Papers from the AAAI Workshop* (Technical Report WS-00-08). Menlo Park, CA: AAAI Press.

Davenport, T.H., & Prusak, L. (1998). *Working knowledge: How organizations manage what they know.* Boston, MA: Harvard Business School Press.

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

Johnson, C., Birnbaum, L., Bareiss, R., & Hinrichs, T. (2000). War Stories: Harnessing Organizational Memories to Support Task Performance. *Intelligence*, 11(1), 17-31.

Leake, D.B., Bauer, T., Maguitman, A., & Wilson, D.C. (2000). Capture, storage, and reuse of lessons about information resources: Supporting task-based information search. To appear in (Aha & Weber, 2000).

Leake, D.B. Kinley, A. (1998). Integrating CBR components within a Case-Based Planner. *Papers from the 1998 Workshop on Case-Based Reasoning Integrations* (80-84). Technical Report WS-98-15.

Muñoz-Avila, H., Aha, D.W., Breslow, L.A., Nau, D., & Weber, R. (2000). Integrating conversational case retrieval with generative planning. To appear in *Proceedings of the Fifth European Workshop on Case-Based Reasoning.* Trento, Italy: Springer.

Muñoz-Avila, H., McFarlane, D., Aha, D.W., Ballas, J., Breslow, L.A., & Nau, D. (1999). Using guidelines to constrain interactive case-based HTN planning. *Proceedings of the Third International Conference on Case-Based Reasoning* (pp. 288-302). Munich: Springer.

Reimer, U. (1998). Knowledge Integration for Building Organisational Memories. In: *Proceedings of the Eleventh Banff Knowledge Acquisition for Knowledge-Based Systems Workshop*. Banff, Canada.

Sary, C., & Mackey, W. (1995). A case-based reasoning approach for the access and reuse of lessons learned. *Proceedings of the Fifth Annual International Symposium of the National Council on Systems Engineering* (pp. 249-256). St. Louis, Missouri: NCOSE.

Secchi, P. (Ed.) (1999). *Proceedings of Alerts and Lessons Learned: An Effective way to prevent failures and problems* (Technical Report WPP-167). Noordwijk, The Netherlands: ESTEC. [www.estec.esa.nl/CONFANNOUN/99c06]

Secchi, P., Ciaschi, R., & Spence, D. (1999a). The ESA alert system. In (Secchi, 1999).

SELLS (1999). *Proceedings of the Society for Effective Lessons Learned Sharing's Spring Meeting*. Las Vegas, NV: Unpublished. [www.tis.eh.doe.gov/ll/sells/proceedings399.htm]

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

Weber, R., Aha, D.W., & Becerra-Fernandez, I. (2000a). To appear in the *International Journal of Expert Systems Research & Applications* Special issue on *Artificial Intelligence and Knowledge Management*.

Weber, R., Aha, D.W., Branting, L.K., Lucas, J.R., & Becerra-Fernandez, I. (2000b). Active case-based reasoning for lessons delivery systems. *Proceedings of the Thirteenth Annual Conference of the International Florida Artificial Intelligence Research Society* (pp. 170-174). Orlando, FL: AAAI Press.

Weber, R., Aha, D.W., Munoz, H., & Breslow, L.A. (2000c). An intelligent lessons learned process. To appear in *Proceedings of the Twelfth International Symposium on Methodologies for Intelligent Systems*. Charlotte, NC: Springer-Verlag.