Task Assistant: Personalized Task Management for Military Environments

Bart Peintner, Jason Dinger, Andres Rodriguez, Karen Myers
SRI International, 333 Ravenswood Ave. Menlo Park, CA USA
peintner@ai.sri.com, jason.dinger@sri.com, acr@ai.sri.com, myers@ai.sri.com

Abstract

We describe an AI-enhanced task management tool developed for a military environment, which differs from office environments in important ways: differing time scales, a focus on teams collaborating on tasks instead of an individual managing her own set of diverse tasks, and a focus on tasklists and standard operating procedures instead of individual tasks. We discuss the Task Assistant prototype, our process for adapting it from an office environment to a military one, and lessons learned about developing AI technology for a high-pressure operational environment.

Introduction

The problem of task management is ubiquitous. Workers in a wide range of fields must address the problems of knowing which tasks to perform, how to perform those tasks, how and when to delegate tasks, and how to deal with task overload. The military is no exception. Command and control staff follow Standard Operating Procedures (SOPs) and Tactics, Techniques and Procedures (TTPs) when responding to events in theater, when planning or executing missions, and when developing reports.

Given the universality of the task management problem, a significant number of task and To Do management approaches and systems have been developed over the years (see Related Work section). However, most have been targeted toward office environments and are not well suited to military environments. Given the high level of uncertainty and pressure in military environments, the need for effective and adaptive task management assistance goes beyond simply increasing productivity: accomplishing tasks more quickly or having fewer tasks drop through the cracks may produce immeasurable benefits. Other high-pressure environments, such as in hospitals, have task management needs similar to the military. A recent study has shown that when checklists are used during surgery, complications and deaths are reduced significantly (Haynes et al. 2009). Similarly, checklists have been shown to reduce the number of infections in an ICU (Pronovost et al. 2006).

In this paper, we describe Task Assistant, a lightweight Java-based workflow management tool specifically designed to assist users operating in military environments. The system manages tasklists, which are hierarchical groups of tasks that collectively achieve a goal. Task Assistant provides the ability to quickly create tasklists, access and customize tasklists created by others, and link functionality in other systems to tasks within each tasklist. Task Assistant assists the user in several ways: learning models that suggest which tasks should be delegated and to whom, learning models that link individual tasks to related automated procedures, and mining a library of existing tasklists for ones that can expand upon or replace a user's current tasks.

Task Assistant builds on Towel (Conley and Carpenter 2007), an earlier system developed at SRI designed for office environments. We describe our process for adapting the system to the military context, and present lessons learned from developing AI-based features for military users.

Differences from Office Environment

For any user-centric software tool to be effective, its use must align well with the current work practices of its users. In our case, this was a difficult problem, given that we, as technology developers, did not have first-hand experience in the target environment or even of the tasks typically performed. As we will describe later, we addressed this problem through frequent interaction with subject matter experts (SMEs). Much of the interaction involved the SMEs sharing scenarios and critiquing our planned features and implementations. One result of this interaction was an understanding of the important differences between task management in office environments and military environments.

First, the time scale is different. Deadlines are typically measured in hours rather than days. In an office environment, new tasks that are due in only a few hours rarely make it onto the To Do list: they are simply executed. In an Army environment, a notification about a downed aircraft implies tens of tasks, many of which are due in minutes.

Second, the focus is less on a single user managing diverse tasks and more on teams performing cohesive collections of tasks. Although the latter may evoke the term *project management*, the smaller time scale and diminished relative importance of assigning resources place the problem outside the reach of current project management tools.

Third, tasklists are the primary focus, not tasks. Military environments maintain a very large number of SOPs and

Basic task and tasklist management	U: Organize tasks into hierarchical tasklists U: Save tasklists in template form to a shared server; instantiate a template U: Easily add and delete tasks (modeled after entering bulleted list in MS PowerPoint) U: Attach files and URLs to tasks to provide task context
Collaboration and Delegation	T: Inform user who is currently "online" U: Delegate a task or group of tasks to another user U: Send another user or group of users a live copy of a tasklist U: Request a live copy of a tasklist U: Contribute tasks to another's tasklist (collaboratively build a tasklist) U: Request control of a task or group of tasks on another's tasklist
Task status monitoring	T: Display time until deadline: color indicates whether task is on track for completion T: Display progress bar for task groups, including those delegated to others T: Indicate information staleness with progress bar color, based on time since last update
Integration with other systems	T: Provide list of links or procedures available in connected systems U: Attach an external link or procedure to a task U: Execute a link attached to a task T: Allow other systems to perform actions in Task Assistant through API (e.g., load a workflow)
Learning and Suggestions	T: Learn which tasks to delegate and to whom T: Learn which automated links or procedures to attach to tasks T: Suggest which tasklists to insert to add detail to a task

Table 1: Capabilities available to the User (U:) and performed by Task Assistant (T:)

TTPs for the purposes of training and guiding responses to different events and planning scenarios. These can be very naturally represented as tasklists; encoding them in a software tool brings them to life, allowing easy access and customization to different environments and command styles.

In general, these differences significantly affect the requirements for a task management system. Although lessons from office-based task managers transfer in part, new principles and strategies are needed.

Task Assistant

Task Assistant is a lightweight Java-based task management tool designed to assist users operating in military environments, giving those users the ability to quickly create tasklists, access and customize tasklists created by others, and link functionality in other systems to individual tasks. These abilities and more are listed in Table 1 and explained below. We illustrate Task Assistant using sample Army response procedures for a Vehicle-Borne Improvised Explosive Device (VBIED) event and the sighting of a High Value Target (HVT).

Basic task and tasklist management. Figure 1 displays the Task Assistant interface, showing a "VBIED attack" tasklist extracted from an Army Brigade Combat Team (BCT) battle drill. Below the menu bar is a set of tabs, each of which represents an active tasklist. Each tasklist consists of a hierarchy of tasks or information items. Each task consists of a textual summary, a task type (task, information, branch point) and several optional properties, including deadlines, durations, a detailed description, and the role of the person who should execute the task. For example, the first task in the tasklist in Figure 1 has the summary "Unit response to explosive incident", a deadline in 18 minutes, and five subtasks.

Notice the file attachment for the task, "Cdr briefing.ppt". Multiple files or URLs can be attached to a task and opened with a single click. This allows the context of the task to be accessible when the task is executed.

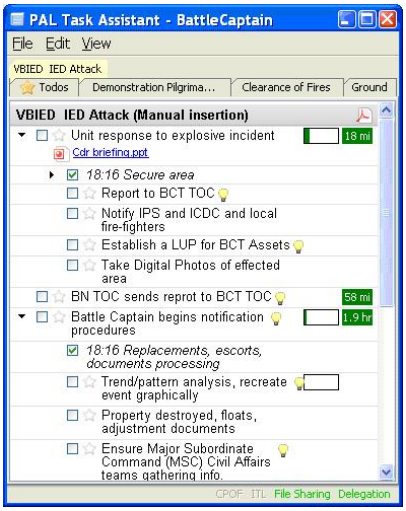

Figure 1: Task Assistant interface.

Creating or modifying a tasklist is fast and intuitive. We chose a PowerPoint-style interaction because most users are familiar with the keystrokes for creating new lines and indenting them (making the task a subtask of the one above it). In addition, Task Assistant supports cut and paste from bulleted lists in Office applications, PDFs, and the web, making it easy to create tasklists from existing sources.

The File menu contains options that allow a user to save and load tasklists from a personal or shared library on a server. The current library encodes approximately 60 Army SOPs and doctrinal tasklists. About half were created by cutting and pasting from Army manuals and half were created by SMEs and retired Army commanders during the buildup to a large demonstration for the DARPA Personalized Assistant that Learns program. Tasklists created using Army manuals were built in about 5 minutes. For those built without existing sources, the time required is only slightly

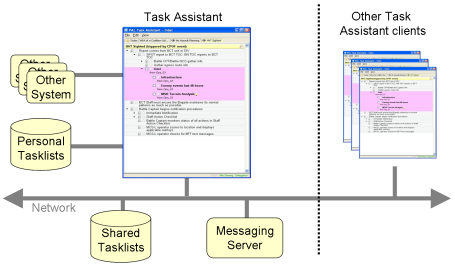

Figure 2: Connections between a Task Assistant client and tasklist libraries, external systems, and other clients.

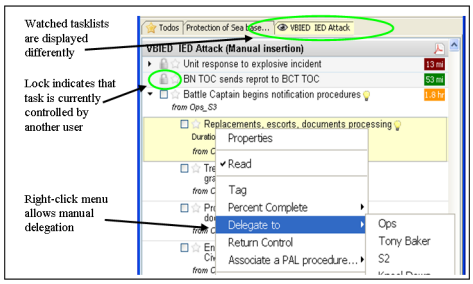

Figure 3: Task Assistant allow users to view tasklists of others, delegate tasks to others, and broadcast tasklists to others.

longer than creating the corresponding Word document. The current library has no structure; the next version will allow tags to be added to tasklists, which will enable different units to maintain their own versions of each tasklist.

The connection between a Task Assistant client and the tasklist libraries is shown in Figure 2, along with connections to external systems and other clients.

Collaboration and delegation. If a network connection is available, each Task Assistant client will announce its presence to other clients. Messaging enables the collaboration and delegation features of Task Assistant, which allow multiple users to build or execute a shared tasklist. The collaboration feature has two related elements. First, a user can delegate a set of tasks in a particular tasklist to another user. This results in the second user getting a new tab in his Task Assistant showing the original tasklist. The delegated tasks appear normally, but all others are annotated with a lock icon, indicating that another user (the user who delegated the tasks) has control of those tasks. Figure 3 shows an example where this user has been delegated a set of tasks in the "VBIED attack" tasklist. The "eye" icon in the tab of this tasklist indicates that this is a view into another's tasklist.

Any changes to the delegated tasks–attached files, changed properties–are immediately reflected on the Task Assistant client for the user who delegated the tasks. Changes to locked tasks are not allowed. However, a user may right-click on any locked task to request control of it, essentially asking the owner to delegate the task to him.

Monitor status of tasks. In Figure 1, the first task has five subtasks, one of which is marked as completed. Its completion is reflected in the progress bar to the left of its parent's deadline indicator, which shows that the deadline is in 18 minutes. The color of the progress bar indicates how recently it has been updated, giving the person who delegated the tasks a sense of whether the information is up to date. A green color denotes a recent update; brown denotes staleness; black denotes a significant amount of time since last update. We call this the 'avocado model', which is distinguished from the more universally known stoplight model used for the background color of the deadline text. Green, amber, or red indicates whether the time remaining is enough to accomplish the incomplete tasks.

These indicators are designed so that a user who delegates many tasks (e.g., a commander) can sense which tasks need attention with a quick glance down the right side of the UI. Any part of the indicator that is not green indicates that a task is falling behind or that information is out of date.

Integrates with other systems. Task Assistant was part of an effort to incorporate learning by demonstration technology into the Army's Command Post of the Future (CPOF) system and the Air Force's Web-enabled Temporal Analysis System (WebTAS). The technology allowed users to click a "Watch me" button in each of these applications, demonstrate a procedure, and then instruct the technology to generalize the sequence of actions into a reusable procedure (Garvey et al. 2009). As a result, the user could "teach" the tools to execute automated procedures.

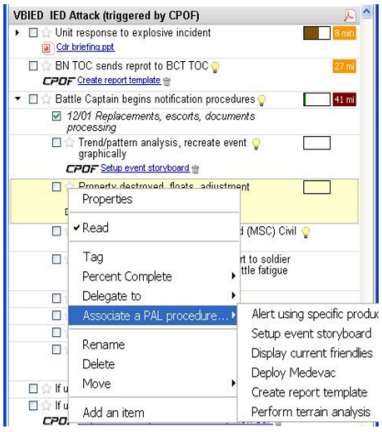

Figure 4: Task Assistant allows links and procedures in external programs to be attached to tasks.

Task Assistant integrated with both of these systems. Each system provided external applications with access to the list of their procedures and the ability to execute them remotely. Thus, a Task Assistant user had the ability to attach automated procedures from both CPOF and WebTAS to individual tasks in a tasklist. This significantly advanced Task Assistant's ability to connect context to each executing task. Not only could the user attach the products needed to perform the task (e.g., Word and PowerPoint files, URLs), the user could attach links to procedures that help execute the task. Figure 4 shows an example. Note that two of the tasks have links to CPOF procedures attached.

The title bar of the VBIED Attack tasklist in Figure 4 also contains the tag "triggered by CPOF", which indicates that the tasklist was inserted in response to an event in CPOF (other tags include "manually inserted", "delegated from User X", and "owner: User X"). Within CPOF, users can set triggers, which consist of a condition and procedure. Whenever the condition is met (e.g., when an element is added to a given area of a map or when an event of a particular type is added to a table), the procedure is executed. The procedure can be a learned CPOF procedure or it can be a command sent to Task Assistant, such as "Insert VBIED tasklist". This enables Task Assistant to be responsive to the environment. Rather than having the user search for a tasklist when an event occurs, the tasklist can load automatically.

Suggestions. One aspect that sets Task Assistant apart from other To Do and task managers is its ability to provide suggestions to the user in an unobtrusive manner (as would a human assistant). Task Assistant provides three types of automated recommendations: suggestions for inserting existing tasklists to expand a task; suggestions for whom to delegate tasks; and suggestions for external procedures to link to tasks. The first of these relies on similarity metrics to identify potential suggestions; the latter two exploit machine learning technology to develop recommendation models derived from prior tasklist executions. Each time the user adds or modifies a task, Task Assistant queries its libraries or learned models to see whether any of these three suggestions is warranted for the task.

Figure 5 shows examples of all three of these suggestions. First, note the light bulb icon to the right of the task "Staff Checklist", toward the bottom of the figure. The light bulb icon indicates that Task Assistant has a suggestion for the user. The figure shows the result of clicking on the light bulb: one or more options are listed for tasklists from the library that are similar to the highlighted task. This suggestion for inserting a tasklist provides value in at least two situations. The most common is when the user is starting to build a large group of tasks, e.g., a doctrinal step such as "Mission Analysis". Rather than requiring the user to type in all steps in this process, Task Assistant helps the user make use of the fact that someone else has already created this tasklist. Simply clicking on the "Insert Mission Analysis" menu item replaces the existing task with the full tasklist, which can then be customized. The second situation is when a user is delegated a task from a higher echelon. From the perspective of the higher echelon, it is a single task, but the task is a hierarchy of tasks from the perspective of the user. Task Assistant will help the user find the tasklist that corresponds to the high-level task. In both cases, the user can use the tasklist as is or customize it to suit the situation at hand. Currently, Task Assistant does not personalize these suggestions to a particular user, but we plan to add this capability.

Second, the task "Get HVT background data" in Figure 5 shows a light bulb icon below the summary text and to the left of a link called"Delegate to Intel_T". Clicking on this link will delegate this group of tasks to the Intel expert in the command center. (Without this suggestion, the user would have to right-click on the task, select `Delegate to ...` and then find the person desired.) If the user decides this

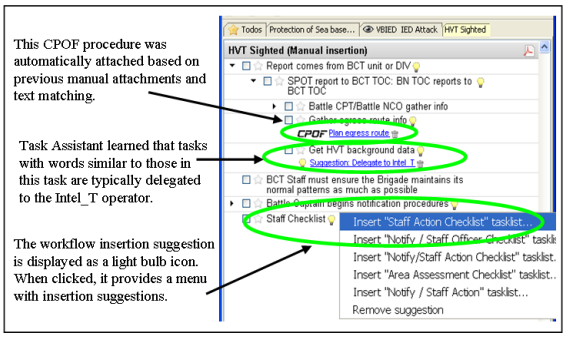

Figure 5: Task Assistant provides suggestions for inserting existing tasklists to expand a task; for whom to delegate tasks; and for which external procedures to link to tasks.

suggestion is poor, he can click the trash can icon to the right of the suggestion to remove it. This suggestion is based on a learned model that uses properties of each task to predict the relevance of each task to other known users. Using manual delegation actions and suggestion removal actions, Task Assistant learns which tasks to delegate and to whom.

Finally, note the CPOF procedure "Plan egress route" circled in Figure 5. Procedures from CPOF or other applications can be attached or removed using the right-click menu. However, Task Assistant automatically attaches a procedure when its learned model indicates that the procedure is appropriate for the task. Using both string matching and the manual attachment and removal of procedures, Task Assistant learns which procedures to attach to which tasks. In some cases, this simply saves the time it takes for a user to navigate the right-click menu and the library of procedures; given hundreds of procedures, this can be significant. Of greater value, perhaps, is when Task Assistant can help find procedures the user did not know existed, e.g., when a member is new to a team or is in training.

Learning Methods

As we will explain below, our target users do not give special credence to "AI features"; they evaluate technology mostly on ease of use and contribution toward critical tasks. This viewpoint imposed three requirements for our learning approach: (1) training the learners must require no additional effort, (2) "using" the learning must not interfere with the interaction or require additional steps, and (3) the system actions based on learning must be acceptable before any learning occurs. These requirements preclude initial information-gathering steps and explicit feedback requests to the user (e.g., "Was this a good suggestion?"). They also forced us to choose techniques in which the initial model would provide value, or at least not reduce value.

Thus, our initial ideas for more knowledge-intensive learning techniques were not pursued. In particular, we had considered requiring the user to enter relationships and profile information to aid in delegation learning. For procedure attachment learning, we considered interface elements that would allow the user to tag and rate automated procedures.

In the end, we chose a scheme that relied solely on the properties of the tasks and a single (optional) user profile entry: a list of the user's roles.

The techniques for learning to whom to delegate tasks and which procedures to attach to each task are similar. Both use what is essentially a Naive Bayes classifier that treats each word in the summary and description as a feature. We maintain a model for each possible prediction class (e.g., a person or a procedure). Each model consists of a set of weighted support objects, which are elements of tasks, including particular words and particular property values for each task. Each time the user creates a connection (e.g., delegates a task to a person) or removes a connection (e.g., removes a procedure from a task), the weights in the model of the class in question are updated. Thus, in keeping with the first requirement, learning is based on implicit feedback gleaned from essential user actions: no additional steps or clicks are needed to train the learners.

For example, if the user delegates to the Intel operator a task called "Get HVT background data" with description "Retrieve all data related to the HVT and associations", the model for class "Intel" is retrieved and the weights for each (nonfiller) word in the summary and description are increased. Conversely, if Task Assistant suggests that a task should be delegated to Intel but the user disagrees, the user *has the option* to delete the suggestion, thus decreasing the weights of that task's support objects for Intel. Over time, after many weight increases and decreases, the model for a class will reflect the types of words used in tasks delegated to Intel. The automatic appearance of the suggestion and the optional nature of using it or discarding it satisfies the second requirement listed above.

The model for each prediction class is initialized with a set of words that describe it. For automated procedures, we had access to its user-created name and thus automatically added those words to the model with significant initial weight. For delegation targets, we seeded the model with their names and with their list of roles (each client shares its roles with other clients).

To make predictions for a given task, the weights for support objects in the task are summed for each prediction class. If the largest score for all classes is above a threshold, the class with the largest score is suggested.

The mechanism for updating weights in each model was more art than science. Each training example was given a weight allocation that was different for positive and negative examples and different for each learning problem. The allocation was spread over the words extracted from the task. Thus, for tasks containing only a few words, each word was given greater weight than for tasks with many words.

The first result of this approach was that suggestions made prior to any learning made intuitive sense to the user: the words in the task matched the words in the procedure or the role of the delegatee. The second result was that the effect of learning was evident very quickly: after a user delegates the first task containing the words "get" and "info" to the intelligence officer, the very next task containing those words will have the suggestion "Delegate to Intel" directly below it. Together, these results satisfy the third requirement of our learning scheme: the system actions based on learning must be acceptable from the start.

This approach worked well in practice, although it was not thoroughly evaluated in a real-world setting. In our cases, a maximum of about ten users was available to accept delegations, and the library of possible procedures was in the tens, not hundreds that are possible in a full deployment.

Development Approach and Lessons Learned

Because our team did not have first-hand experience in Army environments, we developed Task Assistant in tight collaboration with Army personnel, current and retired, interacting with one or more SMEs about every three weeks on average. This interaction was crucial: most of our initial research agenda for Task Assistant was gradually supplanted by another through suggestions and critiques from the SMEs. In general, the advanced functionality we had planned would not have value until we tackled the problem of integrating Task Assistant into the work practices of the particular Army unit type, a Brigade Combat Team (BCT).

Initially, we simply demonstrated our baseline functionality (essentially, a personal task manager for office workers) to SMEs and asked them to brainstorm on how such a tool might be used. Given this feedback, we began adding the features that required little effort and mocking up more development-intensive features. Using the new version, we then trained the SMEs on Task Assistant and presented the mocked-up features and options for changing them. This proved extremely valuable: hands-on experience for the SMEs solidified their opinions, made them more excited about the possibilities, and uncovered aspects of the tool they found annoying or simply misaligned with how they do business (e.g., they did not want to know the time a task was due, they wanted to know how much time was left).

The difficulty arose when our SMEs had differing opinions. For example, one SME wanted tasks presented as text, whereas another preferred them on a timeline. Digging into these differences helped us to better understand the domain. In this case, the two SMEs saw Task Assistant being used for different purposes: one for planning a mission, the other for executing it. This was common in our experience. Given the flexibility of the tool, many SMEs intended it for different purposes; typically, the purpose for which they intended it matched their experience or role when on active duty.

User Response

Throughout the one-year development period, we had the opportunity to give demonstrations and receive feedback from numerous current and former Army personnel. We were met with near unanimous excitement for the tool, even before the AI-related features were demonstrated. This speaks to the need for the core functionality provided by Task Assistant. The AI features excited them as well, but only after being demonstrated in the domain of interest. Being unaccustomed to assistive or personalized features, they were skeptical of their value until it was demonstrated. The key aspect that won them over was that the personalization added value without requiring extra effort. They demanded

simplicity above efficiency of use, keyboard shortcuts, and having multiple ways to achieve the same result.

Most of our user involvement was in the context of preparing for a large evaluation in December 2008 of technologies within DARPA's Personalized Assistant that Learns (PAL) program. The evaluation compared the performance of two 7-person teams monitoring and commanding an Army brigade in realistic scenarios, operating with and without PAL-enabled technologies. Each team used their respective PAL or non-PAL technologies to coordinate actions for simulated battalions, to receive orders from a division commander, and to respond to a flurry of events injected into their systems. Although the event was focused on other PAL technologies, Task Assistant was installed on the players' workstations for the event and we were allocated a small amount of time to train the users on the tool.

In the nine months prior to this evalution, four Army SMEs worked closely with Task Assistant. Typically, they would begin to use the tool for a particular purpose, then stop, telling us what must be changed before they continued using it. Toward the end, they were using it to encode scenarios they were planning for the large evaluation. They were impressed and surprised by how quickly they could develop these scenarios using the tasklist insertion suggestions. Many Army standard operating procedures have similar substeps, so very often the SME would need only to enter the high-level steps of the tasklist, and then customize the substeps suggested by Task Assistant.

A few months prior to the evaluation, we took the opportunity to train the 14 users on Task Assistant and let them use it when they were operating with the other PAL technologies. The training spanned four weeks, but Task Assistant was allocated only three 3-hour sessions for training. Most users could attend only one or two sessions. Most users quickly learned the tool, but there was not sufficient time to work through use cases for how Task Assistant could be used effectively in the demonstration.

Although most users did not expend a lot of effort to learn Task Assistant (they were instructed to focus on another tool), those who did found it useful and saw great potential if fielded. During one planning phase of the event, one user claimed that Task Assistant gave him a great advantage when he was able to load a tasklist for a similar plan developed by another SME. The existing tasklist enabled him to leverage mental effort expended by the other SME, who developed the tasklist in a nonpressure situation. Users felt Task Assistant's value would be best demonstrated in real-world settings, where complexity and scope is much greater. Managing multiple tasks with an orchestration of many team members would make Task Assistant's value more evident.

Some of the key features of Task Assistant could not be evaluated in the event. For example, the collaboration and delegation capabilities were not as useful when staff members were so physically close together: they could simply ask a teammate to perform a task. The event gave us another iteration of valuable feedback, but was not sufficient to evaluate the utility of Task Assistant for their environment.

Beyond the PAL evaluation, we also demonstrated Task Assistant to other Army personnel at a number of bases. The response was enthusiastic. Surprisingly, we had several requests for permission to install the software on their office computers for noncombat uses, such as managing other on-base procedures that typically involved a substantial amount of email.

Based on the initial enthusiastic response, we are finalizing plans to deploy the technology to an Army unit this year. The deployment will allow more extensive training and a more formal evaluation. We plan to first evaluate the AI technologies separately through a controlled experiment to quantitatively determine how well the procedure attachment and delegation suggestions adapt to user input. Then, prior to deployment, we hope to conduct a user-satisfaction evaluation with an Army unit after significant use.

Lessons Learned

Throughout the experience, we learned several lessons with respect to deploying AI technology.

First, users do not distinguish between AI features and "normal" features—they are all just features that help or get in the way. While we were most interested in the AI features (the learning and advanced workflow management), the users were more interested in what we would consider mundane aspects, such as whether the date is shown after "checking off" a task or knowing when someone else has modified a task. Thus, to let the AI features shine, equal or more effort needs to be put into the non-AI features.

Second, the mundane aspects of the software must fit well into the user's mode of operations: users do not have the will to "get past" a deficiency to gain the benefits of the AI technology. Users will assume that if the simple things are not perfect, then the advanced features cannot be trusted. An iterative design process is key to avoiding this pitfall. Getting feedback often is critical, and partially implementing features to get feedback speeds the process.

Third, task managers and other personal assistants are secondary tools, meaning that they support a user in accomplishing the primary task. Therefore, it is difficult for a user to adopt them, because her focus is on another tool. In fact, a user of a single tool typically prefers to have secondary tools built in. One way to address this is to ensure that the tool has at least one feature users cannot live without—doing so will motivate learning and use of the remainder of the tool.

Finally, we learned the value of having multiple SMEs from a variety of backgrounds. It is difficult to understand a domain when one has not "lived it" for an extended period. Multiple SMEs speed the learning curve by explaining situations in varying ways and introducing you to different ideas and different aspects of the domain.

Related Work

The Army has produced at least one other system with the intent of encoding standard operating procedures. Named the Military Decision Making Process Assistant, it lacks Task Assistant's flexibility. Soldiers cannot modify tasklists, which means that the SOPs are not tailored to the soldiers' work styles and environment, and therefore is little used.

One of the first distributed task management tools for office environments was Task Manager (Kreifelts, Hinrichs,

and Woetzel 1993), which contained many of the office-oriented features of Task Assistant, including collaborative work on tasks, tasklist templates, and attaching resources to tasks. It added an email capability that let users create email threads based on a particular task. This is the closest system to Task Assistant, but it does not have any learning capabilities or support for linking to automated procedures.

Many of the more recent task management tools have focused on the integration of email and task management. Ethnolographic studies have shown that many use their email clients as their primary task management tools (Ducheneaut and Bellotti 2001), mostly because many tasks either arrive through email or are discussed there. The tools TaskMaster (Bellotti et al. 2003), TaskVista (Bellotti et al. 2004), and TimeStore (Yiu et al. 1997) are email clients specifically designed to support task management. In a BCT environment, email is not used: communication is via voice, instant messaging, and shared information products. Therefore, these insights do not transfer to our military environment.

A variety of commercial task management tools exists, including Accomplice (Accomplice 2009), a PC-based tool specializing in attaching context to tasks (email messages, web pages, etc.), and Remember the Milk (Remember the Milk 2009), an online To Do manager. Like the research systems, they focus on individuals in office environments, and consequently are not suitable for our problem.

Other forms of task management outside the office environment include tasking manufacturing plants (Lau, Tso, and Ho 1998), coordinating flight crew management (Schutte and Trujillo 1996), and computer process or thread management (Adya et al. 2002).

Conclusion

We have described the process of adapting a personalized office To Do manager to fit the needs of a military environment, which differ in three important ways: shorter time scales, the focus on the team instead of the individual, and the focus on tasklists instead of tasks. We described the Task Assistant system and our approach to the first phase of deployment. We listed several lessons we learned from the experience, which we believe may generalize to help others attempting to deploy AI technology to new environments.

The enthusiastic response of SMEs and current army personnel has led to current and future work in increasing the role of learning in Task Assistant as well as further hardening and additional deployments in the coming year.

Acknowledgments

We thank Tom Garvey for many of the initial ideas for this work. We also thank Hal Dick and Kevin Vallandingham, Army subject matter experts who helped us align the tool with Army practices and provided many suggestions for improvement. This work was supported by the Defense Advanced Research Projects Agency (DARPA) under Contract No. FA8750-07-D-0185/0004. Any opinions, findings, and conclusions or recommendations expressed are those of the authors and do not necessarily reflect the views of DARPA or the Air Force Research Laboratory.

References

Accomplice. 2009. accomplice.vertabase.com/.

Adya, A.; Howell, J.; Theimer, M.; Bolosky, W. J.; and Douceur, J. R. 2002. Cooperative task management without manual stack management. In *ATEC '02: Proc. General Track of the Annual Conference on USENIX*, 289–302.

Bellotti, V.; Ducheneaut, N.; Howard, M.; and Smith, I. 2003. Taking email to task: the design and evaluation of a task management centered email tool. In *CHI '03: Proc. SIGCHI Conference on Human Factors in Computing Systems*, 345–352.

Bellotti, V.; Dalal, B.; Good, N.; Flynn, P.; Bobrow, D. G.; and Ducheneaut, N. 2004. What a to-do: Studies of task management towards the design of a personal task list manager. In *CHI '04: Proc. SIGCHI Conference on Human Factors in Computing Systems*, 735–742.

Conley, K., and Carpenter, J. 2007. Towel: Towards an intelligent to-do list. In *Proc. of the AAAI Spring Symposium on Interaction Challenges for Intelligent Assistants*, 26–32.

Ducheneaut, N., and Bellotti, V. 2001. E-mail as habitat: An exploration of embedded personal information management. *Interactions* 8(5):30–38.

Garvey, T.; Gervasio, M.; Lee, T.; Myers, K.; Angiolillo, C.; Gaston, M.; Knittel, J.; and Kolojejchick, J. 2009. Learning by demonstration to support military planning and decision making. In *Proc. of the 21st Conference on Innovative Applications of Artificial Intelligence*.

Haynes, A. B.; Weiser, T. G.; Berry, W. R.; Lipsitz, S. R.; Breizat, A.-H. S.; Dellinger, E. P.; Herbosa, T.; Joseph, S.; Kibatala, P. L.; Lapitan, M. C. M.; Merry, A. F.; Moorthy, K.; Reznick, R. K.; Taylor, B.; Gawande, A. A.; and the Safe Surgery Saves Lives Study Group. 2009. A surgical safety checklist to reduce morbidity and mortality in a global population. *New Engl Jrnl of Medicine* (to appear).

Kreifelts, T.; Hinrichs, E.; and Woetzel, G. 1993. Sharing to-do lists with a distributed task manager. In *Proc. Third European Conference on Computer-Supported Cooperative Work*.

Lau, H.; Tso, S.; and Ho, J. 1998. Development of an intelligent task management system in a manufacturing information network. *Expert Systems with Applications* 15:165–179(15).

Pronovost, P.; Needham, D.; Berenholtz, S.; Sinopoli, D.; Chu, H.; Cosgrove, S.; Sexton, B.; Hyzy, R.; Welsh, R.; Roth, G.; Bander, J.; Kepros, J.; and Goeschel, C. 2006. An intervention to decrease catheter-related bloodstream infections in the ICU. *New Eng Jrnl of Medicine* 355(26):2725–32.

Remember the Milk. 2009. www.rememberthemilk.com/.

Schutte, P. C., and Trujillo, A. C. 1996. Flight crew task management in non-normal situations. Technical report, NASA Langley.

Yiu, K.; Baecker, R.; Silver, N.; and Long, B. 1997. A time-based interface for electronic mail and task management. In *Proc. HCI International*, volume 97, 19–22.