The RADAR Test Methodology: Evaluating a Multi-Task Machine Learning System with Humans in the Loop

Aaron Steinfeld, Rachael Bennett, Kyle Cunningham, Matt Lahut, Pablo-Alejandro Quinones, Django Wexler, Dan Siewiorek, Paul Cohen[1], Julie Fitzgerald[2], Othar Hansson[3], Jordan Hayes[3], Mike Pool[4], and Mark Drummond[5]

October 2006
CMU-HCII-06-102

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

This report also appears as Computer Science Technical Report CMU-CS-06-125

Abstract

The RADAR project involves a collection of machine learning research thrusts that are integrated into a cognitive personal assistant. Progress is examined with a test developed to measure the impact of learning when used by a human user. Three conditions (conventional tools, Radar without learning, and Radar with learning) are evaluated in a large-scale, between-subjects study. This paper describes the RADAR Test with a focus on test design, test harness development, experiment execution, and analysis. Results for the 1.1 version of Radar illustrate the measurement and diagnostic capability of the test. General lessons on such efforts are also discussed.

[1] University of Southern California
[2] JSF Consulting
[3] Thinkbank, Inc
[4] IET, Inc (Formerly at)
[5] SRI International

This material is based upon work supported by the Defense Advanced Research Projects Agency (DARPA) under Contract No. NBCHD030010. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the DARPA or the Department of Interior-National Business Center (DOI-NBC).

Keywords: machine learning, human-computer interaction, artificial intelligence, multi-agent systems, evaluation, human subject experiments

1 Overview

The RADAR (Reflective Agents with Distributed Adaptive Reasoning) project[1] within the DARPA PAL (Personalized Assistant that Learns) program is centered on research and development towards a personal cognitive assistant. The underlying scientific advances within the project are predominantly within the realm of machine learning (ML). These ML approaches are varied and the resulting technologies are diverse. As such, the integration result of this research effort, a system called Radar, is a multi-task machine learning system.

Annual evaluation on the integrated system is a major theme for the RADAR project, and the PAL program as a whole. Furthermore, there is an explicit directive to keep the test consistent throughout the program. As such, considerable effort was devoted towards designing, implementing, and executing the evaluation. This document describes this process, protocol, and some of the results for the Radar 1.1 test. Note that this document is not centered on Radar features or the actual machine learning methods used.

It is also important to note that the RADAR project differs from the bulk of its predecessors and its companion PAL program project, CALO[2], in that humans are in the loop for both the learning and evaluation steps. Radar is trained by junior members of the team who are largely unfamiliar with ML methods. Generic human subjects are then recruited to use Radar while handling a simulated crisis in a conference planning domain. This allows concrete measurement of performance using a human-ML system view – a measurement that closely approximates real world measurement of ML benefit.

1.1 The Radar system

Radar is specifically designed to assist with a suite of white-collar tasks. In most cases, the specific technologies are designed to be domain agnostic (e.g., email categorizing, resource scrounging, etc). However, for the purposes of the test, the base data present in Radar and used for learning is centric to the domain of *conference planning*. As such, certain components appear to be domain-specific but their underlying technologies are not (e.g., conference-related email categories, room finding, etc).

While testing was performed on several Radar 1.x versions, they generally contained the same machine learning components (Table 1). The major variations were due to engineering and user interaction improvements in a number of components and the removal of the Briefing Assistant for engineering reasons. Again, the individual ML technologies will not be described in detail here – the focus of this document is the methods and materials used for the multi-task ML evaluation.

An important distinction is whether a ML component “learns in the wild” or requires special interaction to gain knowledge. Learning in the wild (LITW) is the primary mission of the RADAR project and is specific to learning that occurs through the course of daily use. Brute

[1] http://www.radar.cs.cmu.edu/
[2] http://caloproject.sri.com/
Correspondence: steinfeld@cmu.edu

Table 1. Radar 1.x releases

Radar		LITW	ML Component
1.0	*1.1*		
X	X		Annotations Database (AnnoDB): email parsing and related natural language processing (NLP)
X		X	Briefing Assistant (BA): summarization of activity
X	X	X	CMRadar-Rooms (Room Finder): resource scrounging by learning room reservation owner responses
X	X	X	Email Classifier: assign task-oriented labels to email messages
X	X		Scone: knowledge representation support for the AnnoDB
X	X	X	Space-Time Planner Elicitor (STP Elicitor): elicitation of facts about the world in order to do better optimizations
X	X	X	Virtual Information Officer (VIO): classification and extraction for specific information updates on websites
X	X	X	Workflow by Example (WbE): batch website updates from training on comma separated value input files

force spoon-feeding and code-driven knowledge representation is not LITW. To count as LITW, learning must occur through regular user interaction and user interfaces present in Radar.

An example of brute force encoding would be asking someone to copy the campus building specifications into Radar all at once. However, learning is LITW if Radar decides knowing the capacity of a certain room is really important, Radar asks the user for the capacity, and the user looks it up and enters the specific value.

1.2 Test conditions and overall hypotheses

In order to show the specific influence of learning on overall performance, there are two Radar conditions – one with learning (+L) and one without (-L). In the context of the evaluation test, learning is only LITW. Learning acquired through knowledge engineering by a programmer or through brute force encoding would be available in both the +L and -L Radar conditions.

To the user, Radar is essentially a system layered into Outlook (Figure 1). The components in Table 1 are either behind the scenes (e.g., Scone, AnnoDB) or visible as modified Outlook views (e.g., Email Classifier, VIO) or separate GUI windows (e.g., STP). In many ways, the user interaction development aspect of Radar lagged behind the learning components. This was largely due to limitations in Outlook and user interaction is targeted for improvement in the next version of Radar.

A third condition where subjects utilize conventional off the shelf tools (COTS) allows estimates to be made on the overall benefit of integration, optimization, engineered knowledge, and improvements in user interaction as compared to the current state of the art. For this application, this toolset consists of an unaltered version of Outlook, the schedule in an Excel spreadsheet instead of the STP GUI, a web portal to the room reservation system, and the conference website which could be manually updated. A future condition in the test will be a “human + assistant”

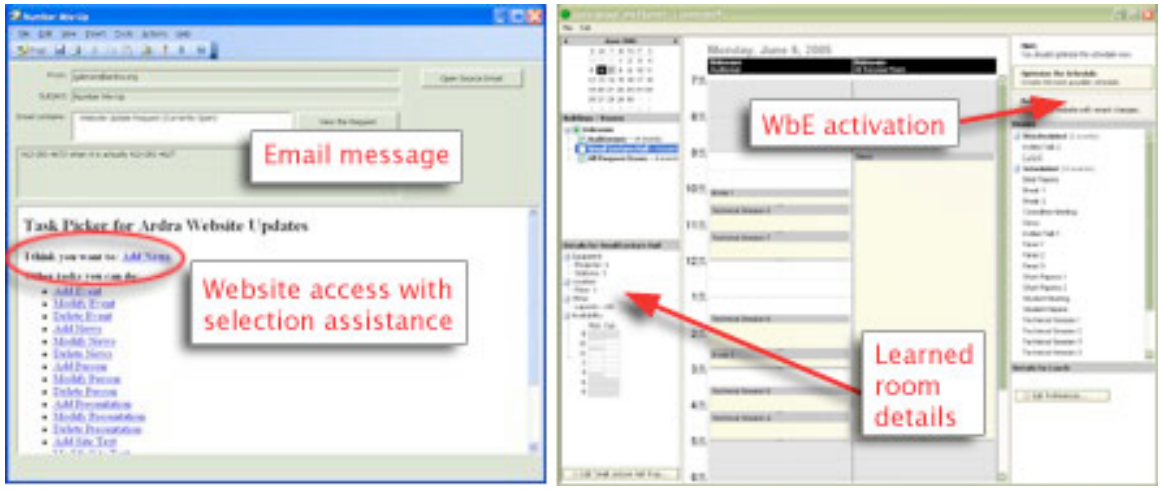

Email message
Task Picker for Ardra Website Updates
Website access with selection assistance
WbE activation
Learned room details

Figure 1. Example Radar screenshots showing VIO embedded in Outlook (left) and the STP stand-alone GUI (right)

case, currently termed COTS+A which will be used to see how well Radar magnifies an individual's abilities to complete tasks.

The primary mission of the evaluation test was to examine two top-level hypotheses. The hypotheses are that subject data will show that:

1. Radar with learning (+L) beats Radar without learning (-L)
2. Radar beats conventional tools (COTS)

The comparison in Hypothesis 1 is commonly called the Learning Delta. These two hypotheses are supplemented by a time axis where subject data will show that:

3. Each year, both Radar conditions improve over the condition's performance in the prior year
4. The year 4 version of Radar with learning (+L) beats COTS+A

1.3 Motivating evaluation factors

A key requirement for the annual evaluation test was repeatability and a consistent level of difficulty so that performance improvements can be measured across years. At a fundamental level, this is nearly impossible to achieve in a complex test of this nature. As such, the goal was to start with a test scenario that was challenging enough to accommodate advances and new research directions for the out-years. A common condition, working the problem with conventional off the shelf tools (COTS) is run for each test, thus permitting benchmarking of small changes to the protocol and each test's stimulus package (e.g., specific crisis, additional tasks, etc). Furthermore, the stimulus package for the test is bound by parameters that are broad enough to prevent training to the test, but narrow enough to ensure that the stimulus package will measure the ML technologies present in the version of Radar being tested. Note that some of the features in the test are fixed due to the requirement that the test be the same from year to year and other contractual requirements.

A recurring theme is that the goal is to <u>test a system consisting of Radar and a human</u>. At a high level this means that human subjects may need, or be required, to perform specific tasks manually. In fact, the utilization of a COTS condition where there are no Radar tools makes the ability for full manual execution a test requirement. This nuance also allows for tasks and stimuli that are currently difficult for strictly software tools to complete autonomously. Removal of manual control is allowed for Radar conditions as long as Radar technology replaces the manual inputs. For example, a GUI that allows subjects to manually scrounge for resources can be removed if a Radar component can be used to perform this task.

1.4 Related work

There have been past attempts at creating digital assistants to aid users in the performance of complex activities. Possibly the most memorable and infamous of example of these is the animated paperclip accompanying Microsoft Word. Agents such as these are usually most valuable to a novice, as opposed to an experienced user. Well intentioned though they may be, their knowledge bases are preprogrammed and the systems behind these agents are unable to learn. Thus, their usefulness is short-lived and their assistance rapidly becomes inappropriate.

On the opposite end of the spectrum of assistants, we can find those that are human. While human assistants are malleable, intuitive, accommodating, and are able to expand their knowledge, they lack certain characteristics present in an ideal digital assistant. Humans assistants lack perfect recall, incur latencies on time critical tasks, cannot rapidly compute optimizations and execute other taxing algorithms, are more susceptible to periodic performance losses due to turnover and constrained availability, and cannot operate continuously. Furthermore, human assistants do not scale well – providing an assistant to every human in an organization is cost prohibitive on several metrics.

RADAR is an attempt to achieve the best of both worlds by focusing on a cognitive digital assistant. The presence of learning is the main distinction when using the prefix “cognitive.” The knowledge it obtains can be used to automate and prep tasks, thus providing the assistance of a human without the limitations of a human and making digital assistance more adaptable and suitable for the user. As previously mentioned, this is a multi-task ML system and therefore requires a complex scenario for rigorous evaluation. Unfortunately, research utilizing human subjects to evaluate cognitive digital assistants with demanding tasks of this nature is limited, and so few comparison cases are available.

A lot of research on machine learning is focused on being able to automate tasks so that systems can perform functions on their own. As such, these systems are autonomous agents – rather than providing assistance to a user, they can replace the user for the task. For example, Clymer and Harrison (2002) designed a system that is able to simulate airplanes entering the approach corridor and the queue airplanes form when they intend to land. Their system contains both fuzzy and crisp rules in the classifier block, which is designed for two types of learning, rule induction and reinforcement learning. The evaluation of this system involved the simulation of airplane queues at differing levels of congestion and seeing how the system compared to two other methods of automation. Clymer also simulated a traffic control network in a major city (Clymer, 2002). Again, this evaluation was done only via simulation.

Examples of evaluations of machine learning can also be found in medicine and the biological sciences. Zhang et al (2005) developed a system for clinical diagnosis using fMRI data. Their test involved the use of fMRI data from drug addicts and control subjects and showed that the system was able to distinguish between the two groups.

Machine learning has also been used in other contexts, such as information extraction. Hu et al (2005) performed experiments on using title extraction for document retrieval, conducted on a corpus of documents. Subjects were asked to provide judgments of the degree of document relevance. The results show that the machine learning approach performs well. Unfortunately, this work's use of human subjects to evaluate ML is not common in the literature.

It is difficult to find examples of ML systems that have been rigorously evaluated. Furthermore, these evaluations are largely based on computer simulations and not human subject data. It is quite possible that this is generally the result of the kind of system that is built – something that is not meant to be an assistant but, rather, is designed to perform a task that has specific rules. An assistance system, when designed and evaluated, should be tested alongside the humans that it would help. This has been a common practice for PAL and prior work done by the authors (e.g., RADAR component experiments; CALO tests; Schrag, et al, 2002). As far as the rest of literature is concerned, the use of human subjects to perform rigorous evaluation tasks with ML systems is sparse, at best.

Yoo, et al (2003) designed an adaptive stock tracker called Stock Trader. This was an adaptive user interface that recommended stocks based on the trading profile of the user. It was tested with twelve human users of varying stock trading expertise. Information collected included the acceptance rates of the agent's advice and the time taken for transactions. The authors also performed an experiment with two hundred simulated subjects by creating profiles and measuring the resulting simulated behavior.

TaskPredictor (Shen 2006) is a machine learning system that attempted to predict users' current activities using two parts, one based on the windows in focus and the other based on email. Evaluation of the system involved training the system for a number of days and deploying it within the research group to a total of 9 test subjects. This work tested three predictive models to see which were more accurate.

When broadening to the wider category, there are many examples of digital assistants that do not learn. For example, Sutcliffe et el (2005) designed an advisor tool to help users in multimedia user interface design. While it does perform assistance functions, it does not learn over time, that is, it can be considered a digitial assistant with static knowledge. iAPERAS was a personal assistant for athletes designed by Verlic et al (2005), which was able to provide nutritional and health information to athletes.

The problem still remains that there is relatively little literature on evaluation results of cognitive digital assistants. This may be because most of assistants of this nature are design exercises, not evaluated with humans in the loop, and/or proprietary and unpublished.

1.5 Test storyline

The general scenario for the RADAR Test is that the subject is filling in for a conference planner, who is indisposed, to resolve a crisis in the current conference plan. This crisis is major enough to require a major shuffling of the conference schedule and room assignments. A number of secondary tasks are triggered as a result of this. These include supporting plans (e.g., shifting catering, keynote and VIP handling, AV equipment delivery, adjusting room configuration, etc), reporting (e.g., make changes to the website, issue a daily briefing, etc), and customer handling (e.g., "here is the campus map"). Noise stimuli are also present in the form of unrelated email, unusable rooms, unrelated web pages, and other clutter content.

2 Materials

It was clear from the initial phases of the evaluation effort that the process would require considerable development of materials and their infrastructure. There is a long-term goal to release certain materials for external use in order to remove a barrier for future research. Such releases will occur under the umbrella name Airspace and will occur periodically on a dedicated website[3].

2.1 Test & Radar

The Test Harness (TH) consists of email content, world details, static websites, and an ecommerce vendor portal. The "corpus" consists of the email and world state content. The latter consists of facts about the world (e.g., characteristics of Stever's Dowd Auditorium) and conference (e.g., characteristics of Panel A1).

The corpus initialization includes:

- The predecessor's conference plan in the format of the condition toolset
- Other world state information – e.g., room reservation schedule
- Stored e-mail from the original conference planner, including noise messages and initial vendor orders
- A web-based vendor database and order forms
- Injected e-mail, including details of the crisis, new tasks, and noise (e.g., Table 2)
- E-mail generated from vendor portal – e.g., quotes and confirmations detailing selections made in the portal

2.2 Simulated world

The simulated world and the initial conference were designed to provide clear boundaries on the types of tasks subjects would need to complete, yet also permit large-scale information gathering, high resolution on learned fact variation, and the opportunity to induce a substantial crisis workload. The conference itself was a 4-day, multi-track technical conference complete with social events, an exhibit hall, poster sessions, tutorials, workshops, plenary talks, and a keynote address. The conference was populated with over 130 talks/posters, each with a designated speaker and title. All characters were provided with email addresses and phone numbers. Many were also given fax numbers, website addresses, and organizations.

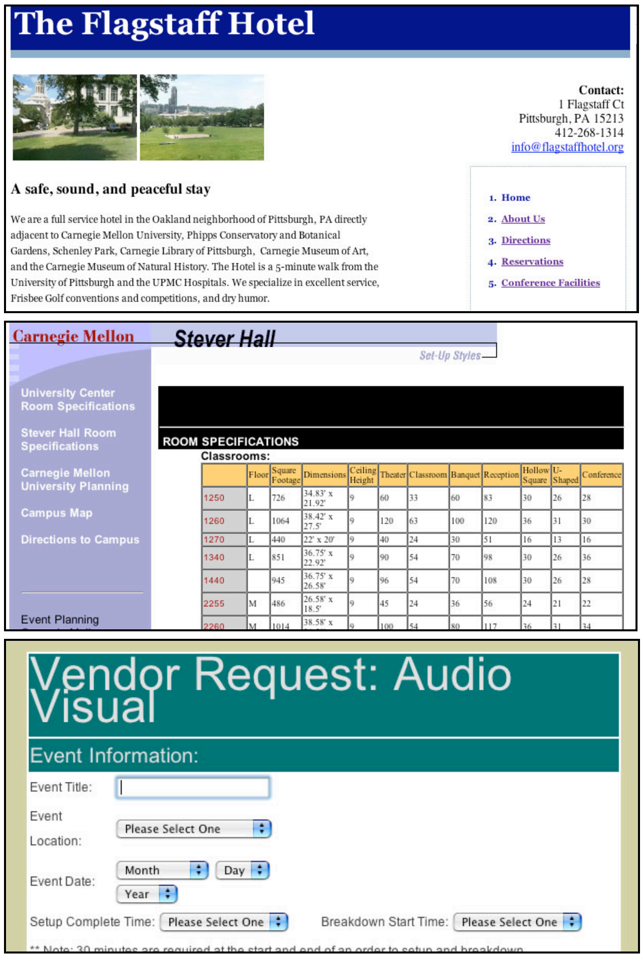
The physical space was a modification and extension of the Carnegie Mellon campus. In addition to modifying the University Center (student union), two academic buildings and a hotel were created and populated. These latter three buildings were instantiated to protect against Carnegie Mellon campus entry knowledge in the subject pool. This information was presented to the

[3] http://www.cs.cmu.edu/~airspace

subject in the form of static web pages easily accessible from the subject's home page (Figure 2, top and middle).

Other static web content included a conference planning manual (complete with documentation of standard task constraints), a PDF of the original schedule, and manuals for the tools used by the subjects (e.g., Excel, Radar, etc).

Subjects were also given access to a "university approved" vendor portal where goods and services could be ordered for the conference (Figure 2, bottom). These included audio-visual equipment, catering, security, floral arrangements, and general equipment rentals. Email receipts, complete with hyperlinks to modification/cancellation pages, were provided for the original vendor orders in support of the conference. All vendor interactions were via web forms since automatic or Wizard of Oz handling of subject e-mails can lead to problems with stimulus consistency and realism. This has face validity since many real-life counterparts are web-based, including the subject signup website used during recruitment.



The Flagstaff Hotel

Contact:
1 Flagstaff Ct
Pittsburgh, PA 15213
412-268-1314
info@flagstaffhotel.org

A safe, sound, and peaceful stay

We are a full service hotel in the Oakland neighborhood of Pittsburgh, PA directly adjacent to Carnegie Mellon University, Phipps Conservatory and Botanical Gardens, Schenley Park, Carnegie Library of Pittsburgh, Carnegie Museum of Art, and the Carnegie Museum of Natural History. The Hotel is a 5-minute walk from the University of Pittsburgh and the UPMC Hospitals. We specialize in excellent service, Frisbee Golf conventions and competitions, and dry humor.

1. Home
2. About Us
3. Directions
4. Reservations
5. Conference Facilities

Carnegie Mellon Stever Hall
Set-Up Styles

University Center Room Specifications
Stever Hall Room Specifications
Carnegie Mellon University Planning
Campus Map
Directions to Campus
Event Planning

ROOM SPECIFICATIONS
Classrooms:

	Floor	Square Footage	Dimensions	Ceiling Height	Theater	Classroom	Banquet	Reception	Hollow Square	U-Shaped	Conference
1250	L	726	34.83' x 21.92'	9	60	33	60	83	30	26	28
1260	L	1064	38.42' x 27.5'	9	120	63	100	120	36	31	30
1270	L	440	22' x 20'	9	40	24	30	51	16	13	16
1340	L	851	36.75' x 22.92'	9	90	54	70	98	30	26	36
1440		945	36.75' x 26.58'	9	96	54	70	108	30	26	28
2255	M	486	26.58' x 18.5'	9	45	24	36	56	24	21	22
2260	M	1014	38.58' x	9	100	54	80	117	36	31	34

Vendor Request: Audio Visual

Event Information:

Event Title:
Event Location: Please Select One
Event Date: Month Day Year
Setup Complete Time: Please Select One
Breakdown Start Time: Please Select One

** Note: 30 minutes are required at the start and end of an order to setup and breakdown.

Figure 2. Static web and vendor portal examples

Table 2. Sample email messages

Signal Message	**Noise Message**
From: jpsontag@ardra.org To: bor@cs.cmu.edu Subject: Lucia di Lamermoor I hate to be a pest, but I finally got tickets to the opera, Lucia di Lamermoor for my wife on our aniversary. It is wednesday night. I want the whole day to ourselves, so I can avoid crashing out plans, that would be great! Let me know. The other days are fine. Thank! J.P.	From: var="kimMail" To: bor@cs.cmu.edu Subject: Hey Uncle Blake! I have a favor to ask you--Mom and Dad's anniversary is coming up, and I wanted to do something special for them, especially since they've been so supportive of the whole wedding concept. I was thinking about getting them tickets to go see "The Phantom of the Opera" when the Broadway Series came to Pittsburgh. I know that sometimes you can get cheaper tickets through work, so I was wondering if that was possible for this show. Please let me know asap so that I can make arrangements! Thanks, you're the best! Kim

2.3 Email corpus

The corpus is a constructed corpus but occasionally utilizes anonymized real content where appropriate (e.g., noise messages). There were initial attempts to acquire an existing email corpus centric to a conference planning activity but this posed significant challenges in the realm of IRB approval due to the need to anonymize all content – including subtle cues that would reveal authors or described individuals. Prior attempts within the Radar project to perform such a step produced haphazard results where entity anonymization was not sufficient.

Even a real conference planning email corpus free of IRB constraints would not be entirely adequate. A real corpus would still require considerable alignment with simulated world entities (e.g., websites, rooms, etc.) and would not necessarily match the ML technologies present in the software release of Radar. For example, the corpus for the real conference may completely lack website update tasks and focus heavily on who to invite as keynote and plenary speakers.

This early investigation led to the determination that the corpus should be fabricated with an eye towards realism and the ML being tested. A team of undergraduate English majors was employed to create a backstory corpus, independent messages detailing one or more tasks, and noise messages. The students were given a series of story arcs, guidelines, and a handful of characters with some specific assigned personalities (e.g., formal, annoying, etc). This effort included a directive to the email authors to let natural errors occur in their writing (e.g., signal message in Table 2). Some characters were assigned personality types that would also lead to different writing styles and email body structure (e.g., terse, bad spelling, etc). Other directives included the utilization of event, paper, and room descriptor variations (e.g., “Dowd in Stever”). Resulting content was screened for fit to the specifications and template syntax adherence.

All email corpus content was in a structure which supported date shifting and variable substitution (e.g., sender in noise message in Table 2). Date anchors and variables were stored in a separate file. These allowed for easy modification of key values by the external evaluators and time shifting of the corpus for experiment execution. Currently, the extent of variable use is rather limited. The intent is to convert certain hard coded content over to variables as part of an ongoing test harness improvement effort.

2.4 Backstory modification, crisis parameterization, and boundaries

Specific crisis vectors and boundaries were negotiated in advance of the test. These provided a large territory for stimulus variability, but also constrained the test stimulus in order to (a) ensure measurement of Radar performance claims and expected research directions, (b) reduce integration mismatches, and (c) reduce mismatches to performance measurement methods.

2.5 Tasks to accomplish

Three types of tasks occurred during the test. The first were incoming queries that needed to be handled and responded to. Examples include queries from attendees, documentation of changes to the world state (e.g., room A seats 30 when configured as a classroom, session B needs a projector), and follow-up action items (e.g., speaker C needs their contact information changed on the conference website).

The second category of tasks were spawned as a result of subject actions. For example, when a subject moved an event to a new room they needed to determine if the event had an existing vendor order or subsequently needed a new one. In the first case, they would need to determine how to change the order, or possibly cancel it, due to the characteristics of the new room and/or the new time slot. Likewise, if the prior room had a feature that the event needed (e.g., a built-in projector) and the new room did not, they would need to order the feature from a vendor.

The third and most important type was artifact production. Artifacts included the schedule, website, and the daily briefing. During the test the subject needed to complete, or make progress on, certain things before they left so that scores could be computed. In reality, some subjects made no headway on specific artifacts (e.g., submitting a briefing) and received a zero or base score for a specific score component.

2.6 Interactions

It is important to identify what information passed between humans and the entity they interacted with. For this study, there were numerous places where a subject could interact with the test harness and/or Radar. Interaction that flowed to the subject is shown in Table 3 while Table 4 shows interaction that flowed from the subject.

3 Metrics

As with the need for new materials, there was also a need for new ways to measure system performance. This was largely achieved through an evaluation score designed and developed by the external evaluators (aka "Final_Score"). Additional metrics were collected via end of session surveys and extensive data logging.

Table 3. Output from test harness and/or Radar to subject

Name	Type	Content	Notes
Base e-mail	Static e-mail	A set of messages collected by original conference planner, including vendor order quotes/confirmations	Present in IMAP collection. The subject may never read some of this material.
Injected e-mail	Injected static e-mail	Incoming requests, constraints, queries, and noise	Timestamps and order were pre-set
Scripted e-mail	Dynamic e-mail	Responses as a result of subject actions	Mostly from web forms (e.g., vendor order confirmations)
Static web	Static web	Vendor portal, initial website, manuals, etc	Content and web forms that do not change
Radar UI	Radar output	Radar outputs not present in e-mail or web UIs	A function of the Radar release components and UI, not present for COTS
COTS tools	COTS output	Budget worksheet for all conditions, schedule for COTS	COTS schedule file has details which match engineered knowledge present in -L (e.g., default STP attendance and room capacity values, etc)

Table 4. Input from subject to test harness and/or Radar

Name	Type	Content	Notes
Sent e-mail	Freeform e-mail	Briefings, information request answers, etc.	
Web forms	Structured field entries	Requests and queries for vendor quotes, web changes, etc	
Radar UI	Radar input	Radar inputs other than e-mail or web UIs	A function of the Radar release components and UI, not present for COTS
COTS tools	COTS input	Manual changes to file contents, equivalent web forms for actions performed by Radar (e.g., finding rooms)	Subjects occasionally create new files - e.g., to do list in Word

3.1 Evaluation score

This complex score function was developed for the purposes of representing performance in a single value. It summarizes overall performance into a single 0.000 to 1.000 objective score. Performance is in terms of points collected by satisfying certain conditions coupled with penalties for specific costs. These include quality of conference schedule (e.g., constraints met, special requests handled, etc), adequate briefing to conference chair, accurate adjustment of the website (e.g., changes to contact information, copying the schedule to the website, etc), and costs incurred while developing schedule. Such costs included both the budget and how often subjects asked fictional characters to give up their room reservations (aka “bumps”). Additional detail on scoring will be deferred to other documents[4].

At the top level, the score coefficients were 2/3rd for the schedule (including penalties for costs incurred) and 1/6th for website updating as needed, and 1/6th for briefing quality.

3.2 General comments about how "Component X" will affect the results

While certain components are clearly central to certain score components, many components have an indirect result on the points collected through automation, facilitation, surfacing of key information, and improved efficiency. This is partly intentional in order to accommodate performance improvements in the out years by components that cannot be imagined. Just because a component is not explicitly tied to points does not mean its impact is minimal. It is important to remember that the test is on the integrated human-Radar system as a whole and not specific component performance. Examination of this issue as it relates to the evaluation score is presented in the results section (§4.4). Some high level notions on how specific Radar components will impact the overall score are presented in the appendix (§9).

3.3 Tasks which will support good score performance

There are a number of tasks a subject can accomplish to achieve good performance on the score function. Furthermore, certain secondary tasks can lead to cascading problems and inefficiencies

[4] For more information on this function, contact author JF (julie.fitzgerald@gmail.com)

if not completed. The subjects were instructed on tool use (e.g., the use of a particular tool to update the website) but many of the details for accurate and rapid performance of these actions were present in email messages, tool manuals, and the conference planning manual. It was up to the subject (and Radar) to find the important constraints, specifications, and action items (e.g., the need to order necessary AV equipment).

The key tasks are:

- Find a conference related message
- Determine what task(s) needs to be done for a conference related message
- Add/modify a schedule constraint
- Make suitable changes to room characteristics
- Find a new room slot
- Realize the schedule needs to be adjusted/optimized
- Adjust/optimize the schedule
- Realize the schedule is adequate
- Update the schedule to the web
- Realize that an event needs a vendor order
- Find existing vendor orders
- Change existing vendor orders
- Make new vendor orders
- Make a single point change to the website
- Complete a briefing
- For Radar +L, create or find an existing WbE rule and commit the result to the website

Some of these tasks have specific sequences or periods when they should be completed. For example, a Radar subject should optimize after applying changes to rooms and/or events. Likewise, the briefing should be done in the waning portion of the session or it will not adequately capture the subject's activity.

3.4 Survey

As previously mentioned, subjects were asked to fill out an end of session survey. The questions, and their respective categories, are shown below. All ratings were a 7-point scale with anchors at 1, 4, and 7 (Strongly agree, Neutral, Strongly disagree). Categories were not revealed to the subjects.

General

1. I am confident I completed the task well.
2. The task was difficult to complete.
3. I could have done as good of a job without the software tools.

Ease of Use

4. Learning to use the software was easy.
5. Becoming skillful at using the software was easy.
6. The software was easy to navigate.

Usefulness

7. Using similar software would improve my performance in my work.
8. Using similar software in my work would increase my productivity.

9. I would find similar software useful in my work.

Collaboration

10. I disagreed with the way tasks were divided between me and the computer.
11. Tasks were clearly assigned. I knew what I was supposed to do.
12. The software did exactly what I wanted it to do.
13. I found myself duplicating work done by the software.
14. I could trust the software.
15. The software kept track of details for me.
16. The software was assisting me.
17. I was assisting the software.

Disorientation

18. I felt like I was going around in circles.
19. It was difficult to find material that I had previously viewed.
20. Navigating between items was a problem.
21. I felt disoriented.
22. After working for a while I had no idea where to go next.

Flow

23. I thought about other things.
24. I was aware of other problems.
25. Time seemed to pass more quickly.
26. I knew the right things to do.
27. I felt like I received a lot of direct feedback.
28. I felt in control of myself.

Questions in the Ease of Use, Usefulness, Disorientation, and Flow categories were drawn from surveys validated in other fields (van Schaik & Ling, 2003 & 2005). Questions 10, 11, and 13 in the Collaboration section were adapted from surveys validated in computer supported cooperative work research (Kraut, et al, review; Fussell, et al, 1998).

At the end of the survey a full page was provided for responses to the following open-ended query:

- Please provide any suggestions on how the software tools could be improved

For the purposes of analysis, responses to each question within each category were flipped to have the same positive/negative direction and averaged as a group. This category level rating is referred to as a macro (e.g., Ease of Use Macro). The exception is the General category – these are not designed to measure a common metric, so they are left independent.

Questions 16 and 17 were specifically designed to examine how the specific mixture of user interaction, ML, and automation affected perceived relationships within collaboration. Ideally, a good mixture will lead to a low score for Question 16 and a higher score for Question 17. This would mean the system was perceived as behaving as an assistant, rather than a taskmaster.

3.5 Other metrics

While there was extensive data logging during test execution, the team is still exploring analyses that use these measures. Much of the logging was ad hoc and is missing critical details with respect to desired analyses. This avenue of research will expand as better logging is incorporated into Radar. Even with these weaknesses, there are already efforts to use this data to simulate performance for new ML components in the areas of task management, task relationships, summarization, and activity monitoring.

4 Test Procedure

4.1 Subject recruitment and utilization

Subjects were recruited via a Carnegie Mellon human subjects recruitment website. The recruitment notice included a requirement that subjects:

- Have not already participated in any version of this study,
- Are between the ages of 18 and 65,
- Do not require computer modifications,
- Are fluent in English, AND
- Are not affiliated with or working on the Radar project.

Due to limitations in the recruitment website, subjects were manually screened for attempts to participate in the study more than once. Payment was calculated per half hour and, per IRB regulations, subjects could leave early. Recruitment and retention for such a long study was addressed by utilizing a high pay rate.

Each subject was run through approximately 3 hours of testing. Each cohort of subjects for a particular session was run on a single condition (COTS, Radar-L, Radar+L). When possible, cohorts were balanced over the week and time of day to prevent session start time bias. Follow-up analyses on this issue revealed no apparent bias. The nominal cohort size was 15 but was often lower due to dropouts, no-shows, and other subject losses (e.g., catastrophic software crash).

Motivation was handled through supplemental payments for milestone completion (e.g., the conference plan at the end of the session satisfies the constraints provided). Subjects were given general milestone descriptions but not explicit targets. These milestones roughly correspond to the top-level coefficients in the score function. Milestone payments ($20 max) were defined to subjects as:

- $12 for schedule at end of the session:
    - All crisis rooms must be empty, all events must be scheduled, and scheduling conflicts and event needs must be adequately resolved
- $4 for issuing end of session briefing
- $4 for updating website as needed

These required manual checking of, respectively:

- The crisis region for events and if there were unscheduled events
- Presence of a briefing to conference chair
- Alteration of the website database by examining the database from a remote shell

Cohorts were run to accumulate approximately 30 subjects per condition. Due to the long nature of subject involvement and the realistic risk of software problems, the team allocated additional time and funds for spare subjects.

4.2 Experiment session

Sessions were planned to consume a total of 3 hours. Subjects were not compelled to avoid leaving the room during the test period (e.g., bathroom break, etc). Only a small number of subjects asked for such permission. The specific session schedule, complete with sub-activities, is shown in Table 5.

4.3 Post-test activities

Due to the complexity of the test and scoring process, certain steps occurred in advance of data analysis efforts. All steps were consistent across the conditions with one exception – the schedule scoring software was designed to accept the Space-Time Planner file format and COTS subjects did not use the Space-Time Planner. As such, two experimenters used the STP GUI to manually position events to match the Excel spreadsheet versions of the schedule produced by the subjects. This step included a few encoding rules designed to provide uniformity during encoding (e.g., if a subject scheduled two copies of an event in the spreadsheet, the first one is used). After the two experimenters encoded the files, their work was compared for discrepancies. If differences were not due to errors or incorrect interpretations of the encoding rules, the external evaluators decided which version should be used. This rarely occurred.

The consistent steps from this point forward were:

- Data for scoring was extracted from subject's schedule and web state snapshot and loaded into the external evaluator scoring code.
- Experimenters extracted and obscured the condition for completed briefings. These were passed to a human panel for scoring. The panel selected 4 item bullets in each briefing and scored each bullet for quality. The bullet scores were then summed and divided by 16 to obtain a score in the 0 to 1 range.
- The entire subject directory (database, IMAP folders, conference website, etc) were archived to 2 locations on different machines. One of these served as an archive copy and was included in the regular SCS back-up schedule.
- Access to the archive copy was limited and not utilized except for cases of emergency.
- Data for analysis was disseminated to the developer teams by the experimenters in a controlled manner. All individuals who were given access to the data were required to complete the Carnegie Mellon IRB training and provide such documentation.
- Some data had accidental documentation of subject identity (e.g., email signatures, etc). These data were restricted to a narrow set of team members. In some cases, "clean" anonymous master documents were generated and disseminated. Briefings were screened during the condition obscuration step.

4.4 Excluded subjects

As mentioned in Table 5, experimenters identified subjects during the session for possible exclusion. Some decisions were made outright during the session due to extremely obvious actions (e.g., sleeping). However, it was often necessary to review subject actions to determine if suspicions were correct and/or if observed issues actually led to biased results.

Table 5. Experiment session schedule (3 hours)

Duration (h:min)	Activity
Prior	Computers prepared • Cohort condition assigned • Tools loaded accordingly • Subject numbers set based on condition, laptop, and cohort numbers • Machines loaded with base test world and external evaluator stimulus package
0:05	IRB • Consent and other IRB documentation • Assignment of subject number based on laptop number
0:30	Instruction on tools only • Instruction on tools available within condition • Tool manuals shown to subjects • Subset of task-oriented video demos shown to subjects • Instruction quiz distributed and taken
1:00	Test • Activity starts
0:10	Break • Activity paused • Subjects instructed to not discuss experiment during the break • Subjects asked to leave room • Machine state snapshots taken
1:00	Test • Resume activity
0:15	Survey/Pay • Activity halted and surveys distributed • Milestones determined while surveys are taken • Surveys collected • Payment for session
Throughout	Observation by experimenters • Detect software crashes and bugs • Very tightly bounded response to subjects questions (e.g., "that information is available to you" ... "I cannot tell you where") • Identification of subjects who should be excluded (e.g., sleeping, leaving early, major software bug, etc.) • Identification of subjects who should be considered for exclusion (e.g., periods of inactivity, suspicion of poor fluency, software bug, etc.)
Post	Archive • Machine state snapshots taken

For example, some subjects clearly lacked verbal fluency in English during subject-experimenter communication. However, this is no guarantee that writing and reading fluency is inadequate for participation. Analysis results for these subjects were reviewed with the external evaluator, who then rendered a judgment on inclusion.

Some excluded subjects were discovered during analysis. The typical situation involved an undetected software problem that led to missing or blatantly biased scores. In one instance, a subject was an outlier due to extreme, inexplicable actions. A more detailed review of this subject's data revealed that this subject was being intentionally disruptive. This subject was promptly excluded.

5 Example Results

5.1 Data source for this example

There were several test windows during the RADAR Year 2 effort. The example shown here is for the final Year 2 evaluation test. This corresponds to COTS and Radar 1.1 tested with a stimulus package of 107 messages, 42 of which were noise. The crisis for this package was a loss of the bulk of the conference rooms for 1.5 days (out of 4 total). A variety of other small perturbations rounded out the task set. These included changes to room details, speaker availability, session preferences, and website details. This stimulus package was designed by the external evaluators and is occasionally referred to as *Crisis 1*.

The subject pool used for analysis, after exclusions and dropouts, was 29, 34, and 32 (COTS, Radar without learning, and Radar with learning). As such, this test accumulated 64 cumulative hours worth of time on task by subjects with a multi-ML system.

Scheduling and scoring for the conditions shown here was not in parallel. COTS data was collected in the fall of 2005 and scored by the external evaluators (IET, Inc[5]). The Radar data was collected in the spring of 2006 under the same stimuli and protocol and scored with IET supplied code (version 0.9) by the Test group with oversight by the current external evaluators[6].

5.2 Score function

Figure 3 shows between subject performance across the three conditions in the Y2 Test. The Learning Delta (the difference due to the inclusion of machine learning) is 0.099, which is approximately 78% of the Overall Delta (improvement over COTS). This suggests that machine learning was the prime contributor to the performance gains.

In this graph, all condition differences are significant and in the expected direction for the initial hypotheses (Table 6). Note that the 95 subjects in this analysis were adequate to show significance between conditions, thus suggesting that cell population sizes of in the range of 30 should be adequate for revealing differences between conditions in future experiments.

As previously mentioned, gains due to specific learning components cannot be singled out explicitly due to the interwoven nature of the scoring function and the considerable interaction within tasks by the different components. Such an analysis is theoretically feasible, but would require running the study multiple times with various learning components reverted to their

[5] Led by author MP
[6] Authors JF and PC

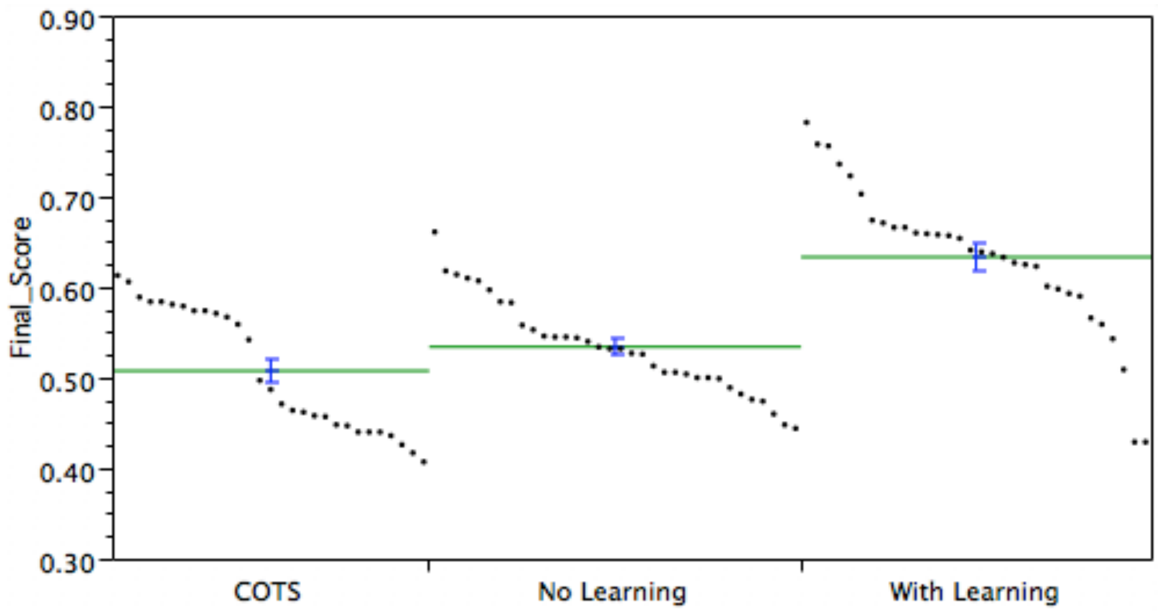


Figure 3. Radar 1.1 results on the initial crisis

Table 6. RADAR 1.1 means and t-test comparisons

Condition	Mean	Comparison	p-value
COTS	0.507	Overall Delta (With Learning > COTS)	<0.0001
No Learning (-L)	0.534	Learning Delta (With Learning > No Learning)	<0.0001
With Learning (+L)	0.633	Nonlearning Delta (No Learning > COTS)	<0.088

nonlearning state. The combinations of this and the effort behind an instance of the test make such an activity unrealistic.

Having said this, it is possible to examine the impact of groups of components on the results by examining the high level score function components. Figure 4 shows that the bulk of Learning and Overall deltas came from machine learning contributions in scheduling and publishing the schedule to the website.

Gains due to publishing the schedule to the website can be tied explicitly back to WbE, but is not the only place where WbE can contribute/detract from overall performance (Table 7). Note that while the Email Classifier contributes to many factors of the score function, its role is to surface the task and not to assist with the completion of the task itself. As such, the negative Learning Delta for the briefing component is not solely due to a deficiency of this learning component. In fact, this difference is more likely due to human decision making related to task allocation – almost twice as many subjects in the nonlearning condition as in the learning condition compiled a briefing (56% vs. 28%). Task identification is not the same as task prioritization.

5.3 Survey findings

Differences between the conditions on the survey macros (§3.4) were largely not significant when examined with t-tests. The exceptions being that COTS was considered easier to use than

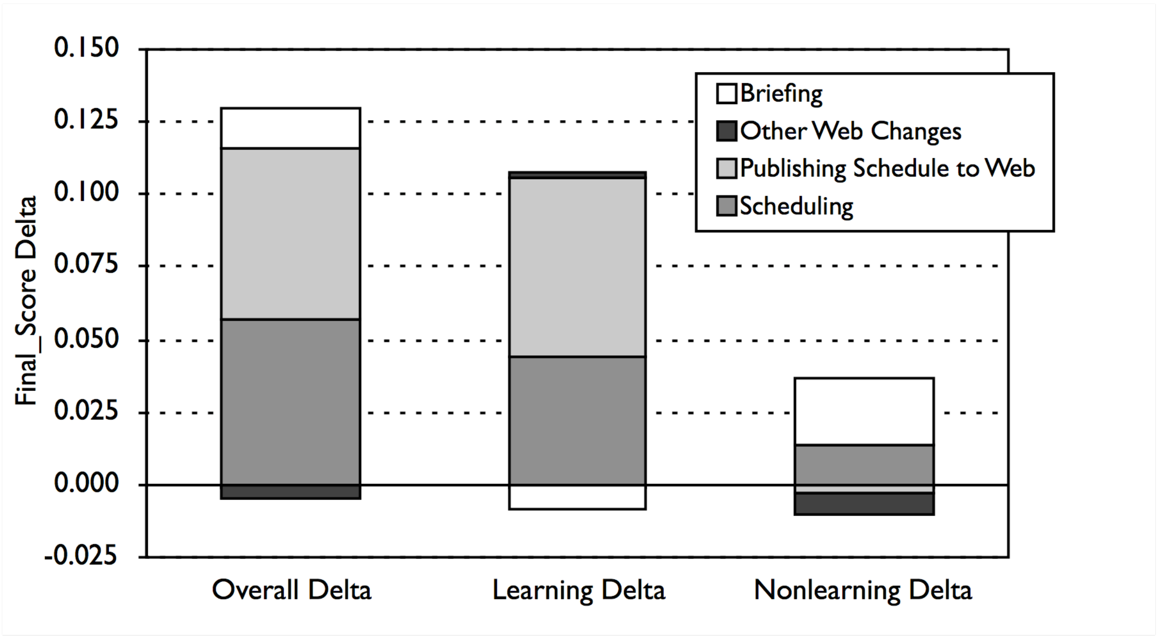


Figure 4. Score component impacts on the overall score

Table 7. Learning component contributors to score function component

Score Component	Learning Technology Contributors
Scheduling	STP Elicitor, CMRadar-Rooms, Email Classifier
Publishing Schedule to Web	WbE
Other Web Changes	VIO, WbE, Email Classifier
Briefing	Email Classifier

either Radar condition and that COTS was reported as more useful than Radar -L (Table 8). These are not surprising in that Radar 1.1 had known usability flaws. There were no significant differences across the conditions for the general questions.

In general, the macros performed reasonably well when tested for measurement reliability (Table 9). Only the Flow macro was markedly below the 0.7 reliability acceptance threshold used in the literature. Collaboration was right on the edge.

Table 8. Significant differences in macro survey results for macros

Macro	**Mean (1-7, lower is better)**		
	COTS	*No Learning*	*Learning*
Ease of Use	3.6	4.6	4.6
Usefulness	3.2	4.0	3.4
Comparison	**p-value**		
Ease of Use: COTS vs. Radar -L	<0.005		
Ease of Use: COTS vs. Radar +L	<0.004		
Usefulness: COTS vs. Radar -L	<0.017		

Using Questions 16 and 17 to examine assistance vs. taskmaster behavior of the system was inconclusive with no significant differences. However, there is a visible shift towards assistance when the Radar conditions are compared to COTS. For Figure 5, the metric Assistant is Q16 – Q17. As with the other survey measures, lower values correspond to stronger agreement.

Table 9. Reliability results for the survey macros

Survey Macro	Cronbach's alpha
Ease of Use	0.87
Usefulness	0.94
Collaboration	0.69
Disorientation	0.81
Flow	0.57

6 Discussion

6.1 Test effectiveness

The initial concern at the start of this endeavor was that the methods and materials would not be adequately sensitive to measure ML technologies that were still being formulated. Much of these methods and materials were designed and developed without full knowledge of what ML components would be developed by the rest of the team. This concern is still valid in that there are new ML components being developed for the next version of Radar. The decision to measure at the top human-Radar system level was an attempt to be robust to unknown ML technologies. As can be seen by the results, this approach clearly captures high level benefits arising from human in the loop multi-task ML.

While not shown here, there have been other human subjects tests with other versions of Radar (§1.1) and the protocol. These have shown changes in performance due to variations in ML, HCI, engineering, crisis difficulty, and human training. As such, the test method and materials have also been shown to suitable for measuring shifts in performance due to a variety of system and scenario effects.

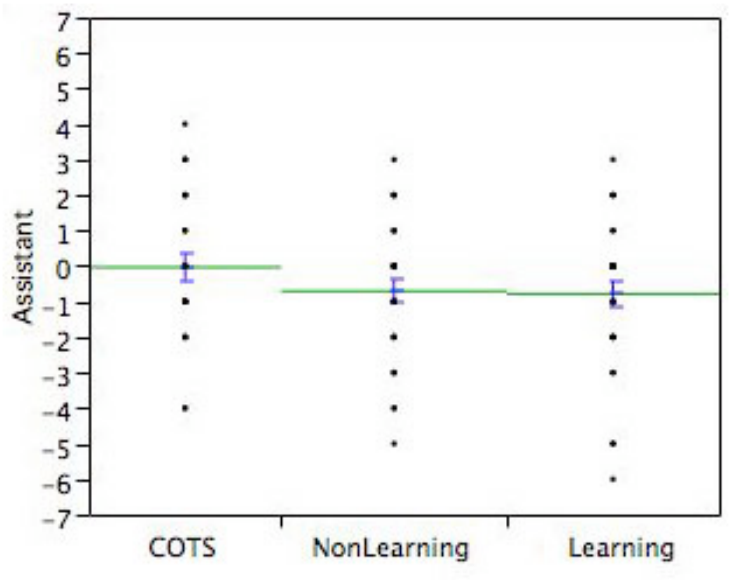


Figure 5. Perception of the toolset as an assistant (lower corresponds to stronger agreement)

6.2 Hypotheses results

The results clearly show that Hypothesis 1 (ML helps) holds true. Likewise, Hypothesis 2 (Radar is better than COTS) is also true. Data for Hypotheses 3 and 4 has not been collected yet, but other tests with earlier versions of Radar suggest that Hypothesis 3 (Radar will improve) will hold.

6.3 Quality assurance and diagnostic capability

The test methodology also revealed a useful byproduct in that the demands of human subject testing revealed a number of unknown flaws in the system due to bugs, mismatches, and other problems. The complexity of the test methods and materials are both a blessing and a curse in this respect. On one hand, they introduce a greater risk of initial state mismatches but on the other hand they exercise more of the system, thus increasing the test's value as an informal quality assurance exercise. On a related note, this complexity also permits sensitivity analysis on various parameters.

6.4 External benefit

Cost is a major barrier for experimental research and a large portion is attributable to stimuli and artifact development. The Test group has made the commitment to provide much of the stimuli and supporting content described here to external parties. As previously mentioned, this will occur through the Airspace website (§2).

Besides the obvious benefit to other researchers in the ML and HCI community, researchers examining decision making under stress, teamwork, and other non-computer science topics could also use this material. This content could even be utilized as a rehabilitation tool for persons with cognitive impairments or accessibility problems with computers and leveraged to instruct new entries into the white-collar workforce.

7 Acknowledgements

Many members of the RADAR project provided ongoing and detailed feedback regarding specific nuances of the test. Jaime Carbonell and Scott Fahlman provided regular advice on ML evaluation approaches and insight into Radar ML technologies. Anthony Tomasic assisted with development of the stimulus boundaries (§2.4). Alex Rudnicky provided insight on briefing scoring techniques and Paul Bennett managed the panel during testing of Radar 1.1 (§4.3). Robert Kraut provided advice on survey development (§3.4).

Valuable protocol suggestions, early experiment design input, and fabrication assistance with an early version of the simulated world (§2.2) was received from Aaron Spaulding. Email content authors included (§2.3): Sarah Barbour, Michelle Bova, Diane Keene, Stephanie Kimssy, and Jennifer Suh.

Dave Gunning was the DARPA program manager for both RADAR and the external evaluation.

Notes: Othar Hansson joined Google after contributing to this work. Both Julie Fitzgerald and Mike Pool left IET during the course of this work. Julie Fitzgerald subsequently acted as an external evaluator through a separate arrangement.

8 References

J. R. Clymer. Simulation of a vehicle traffic control network using a fuzzy classifier system. *Proceedings of the 35th Annual Simulation Symposium 2002*, 1-7, 2002.

J. R. Clymer, & V. Harrsion. Simulation of air traffic control at a VFR airport using OpEMCSS. *Digital Avionics Systems Conference 2002*, 3.B.1.1-8, 2002.

S. R. Fussell, R. E. Kraut, F. J. Lerch, W. L. Sherlis, M. McNally, & J. J. Cadiz.. Coordination, overload and team performance: effects of team communication strategies. In *Proc. of the ACM Conference on Computer Supported Cooperative Work (CSCW)*, 1998.

Y. Hu, Y. C. Hang Li, D. Meyerzon, & Q. Zheng. Automatic extraction of titles from general documents using machine learning. *JCDL 05*, 145-154, 2005.

R. E. Kraut, S. R. Fussell, F. J. Lerch, & A. Espinosa. Coordination in teams: Evidence from a simulated management game. *Journal of Applied Psychology*, under review.

P. van Schaik, & J. Ling. Using on-line surveys to measure three key constructs of the quality of human–computer interaction in web sites: psychometric properties and implications. *Int. J. Human-Computer Studies,* 59, 545–567, 2003.

P. van Schaik, & J. Ling. Five psychometric scales for online measurement of the quality of human-computer interaction in web sites. *Int. J. Human–Computer Interaction*, 18(3), 309–322, 2005.

R. Schrag, M. Pool, V. Chaudhri, R. Kahlert, J. Powers, P. Cohen, J. Fitzgerald, & S. Mishra. Experimental evaluation of subject matter expert-oriented knowledge base authoring tools. *Proc. NIST Performance Metrics for Intelligent Systems Workshop*, 2002.

J. Shen, L. Li, T. G. Dietterich, & J. L. Herlocker. A hybrid learning system for recognizing user tasks from desktop activities and email messages. *IUI*, 86-92, 2006.

A. G. Sutcliffe, S. Kurniawan, & J. Shin. A method and advisor tool for multimedia user interface design. *International Journal of Human-Computer Studies*, 64, 375-192, 2006.

M. Verlic, M. Zorman, & N. Nertik. iAPERAS – Intelligent athlete's personal assistant. *Proceedings of the 18th IEEE Symposium on Computer-Based Medical Systems*, 2005.

J. Yoo, M. Gervasio, & P. Langley. An adaptive stock tracker for personalized trading advice. *IUI*, 197-203, 2003.

L. Zhang, D. Samaras, D. Tomasi, N. Volkow, & R. Goldstein. Machine learning for clinical diagnosis from functional magnetic resonance imaging. In *Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 1211-1217, 2005.

9 Appendix: Expected impact of specific components on the learning delta

This is a description of which learning components were present in Radar 1.x and how they were expected to contribute to the learning delta at a system level. Likewise, certain components also improved performance when compared to COTS due to efficiency and user interface features. Exact measurement for each component is not feasible within the overall score function construct. Estimates for some are possible based on component-level analyses, but other components and user interface effects bias such measurements.

Individual components are shown in Table 10 using an order that roughly corresponds to typical user task sequence: a message with a task arrives, is processed, schedule or web actions are taken, and a status briefing is composed at the end of the session.

Table 10. Expected learning impact

Component	Impact on Learning Delta	Rationale
Classifier	Positive	Pre-categorization of email messages will help through highlighting tasks and reduced GUI actions.
AnnoDB	Neutral*	Extraction and other NLP will help through highlighting tasks and reduced GUI actions.
Scone (knowledgebase)	Neutral*	The kowledgebase will enhance Email NLP efforts through better entity association and understanding.
Space-Time Planner (STP)	Positive	The STP optimizer starts with default ranges for many simulated world facts. Elicitation of specific values will allow better schedule optimization.
CMRadar-Rooms (aka Room Finder)	Positive	Knowledge about simulated character decisions will lead to a better success rate when asking characters to give up their reserved slots. This may also obtain better rooms and will have better pre-bump request abort decision making for unobtainable requests.
Workflow by Example (WbE)	Positive	The STP optimizer inherently introduces a lot of schedule changes. WbE makes updating these to the website more efficient.
Virtual Information Office (VIO)	Positive	Prior work suggests this form of classification and extraction will produce performance improvements for individual website changes but that this may not have a large delta.
Briefing Assistant (BA)	Positive	Learned patterns will permit surfacing of key details to be included in the status briefing.

* It is important to note that these learners were engineered in Radar 1.x and therefore such assistance was present in both versions of Radar.