Research Paper

Needs Analysis and Technology Evaluation: Evaluation Case Study of Alternative Software for Controller Planning Work—Part 2

Dorrit Billman, **Lucia Arsintescu,** San Jose State University @ NASA Ames Research Center, **Michael Feary**, NASA Ames Research Center, **Jessica Lee**, and **Rachna Tiwary**, San Jose State University @ NASA Ames Research Center

A NASA Mission Control group provided us the opportunity to conduct a needs analysis, produce software guided by the needs analysis, and evaluate the software prototype. This paper reports our discriminative evaluation of new prototype software for Attitude Determination and Control Officers, who plan and execute maneuvers of the International Space Station with their Russian counterparts. On a specific, pragmatic level, our evaluation provided evidence of large performance improvement. Relative to legacy software, the new software reduced time and errors by half in a laboratory experiment using valid tasks identified from the needs analysis. Our discriminative evaluation also isolated specific benefits attributable to specific improvements in alignment of technology to the work. Our discriminative evaluation isolated contributing factors affecting performance by using item sets designed to differentially engage software components. This approach for identifying specific factors contributing to performance improvement may be reusable in many situations, particularly where it is not feasible to develop and test many software versions, each differing in just a single factor of interest.

Keywords: cognitive engineering, discriminative evaluation, needs analysis, software design, planning, aerospace

Address correspondence to Dorrit Billman, San Jose State University, NASA Ames Research Center, Mail Stop 262-4, Bldg 262, Rm 177, P.O. Box 1, Moffett Field, CA 94035-0001, Dorrit.billman@nasa.gov.

Journal of Cognitive Engineering and Decision Making
201X, Volume XX, Number X, Month 2015, pp. 1–22
DOI: 10.1177/1555343414567775

Practitioners and researchers in cognitive engineering often have multiple evaluation goals: They want to evaluate the quality of a specific design, and they want to understand the basis for high quality or for improvement over a previous design so that effective elements or methods can be reused in other projects. Although the balance between the goals of specific improvement and general knowledge will vary, often both evaluation goals are in play.

We faced the challenge of multiple evaluation goals in a case study of National Aeronautics and Space Administration (NASA) planning by Attitude Determination and Control Officers (ADCO). ADCO is the NASA Mission Control group responsible for controlling the trajectory of the International Space Station (ISS) and planning those operations. On the pragmatic side, our goal was to evaluate possible benefits of a specific new software prototype, relative to legacy planning software. On the research side, our goal was to evaluate the importance of aligning technology with the work it is intended to support as a factor contributing to improved performance, and the value of the needs analysis methods used to improve alignment. The value of ensuring that work technology is fit-for-purpose may be intuitive, yet there are many forces that can lead to technology that is poorly aligned, particularly in complex domains. These include an assumption that the needs will be obvious to designers or engineers, that a legacy design got it right, or that exploiting new technical capabilities should be the primary driver of design; difficulty identifying needs due to partial, distributed, implicit, or out-of-date knowledge

sources; or assessment that the costs versus benefits of need-driven design do not justify it. In the preceding companion paper (Billman, Feary, Schreckengost, & Sherry, this issue), we presented our needs analysis method, our software redesign informed by that analysis, and how the new software design improved alignment relative to the legacy design.

In this paper, we present our evaluation experiment, which advanced both the specific, pragmatic goal and the general, research goal. It directly evaluated performance using the redesigned versus legacy software, while also providing indirect but useful information about contributors to that benefit. Discriminative analysis aims to identify factors responsible for performance differences between systems, even if those systems differ in multiple ways (Roth & Eggleston, 2013). Broadly, our experimental logic was to create discriminative items that differentially depend on a key feature of the improved software, which in turn is based on a key result of our needs analysis method. We believe the experimental logic used in our study can be reused and extended for other situations and thus is a contribution to discriminative evaluation. After summarizing the results of our needs analysis and the design of new prototype software (detailed in Billman et al., this issue), we report our discriminative evaluation of the revised software and the role of needs analysis in discriminative evaluation.

KEY RESULTS OF OUR NEEDS ANALYSIS FOR ADCO PLANNING

The ADCO group is part of NASA Mission Control for the ISS, in Houston. ADCO, in partnership with their Russian counterparts, determine and control the orientation and trajectory of the ISS. These operators build advance plans specifying what is to be done and when. A plan goes through many refinements and revisions, exchanged between Russian and American controllers. The preceding companion paper reports our needs analysis of a subset of their planning work and describes the prototype planning software designed to be better aligned with the structure of work as identified by that needs analysis.

The primary ADCO planning work is event scheduling and construction of a plan with appropriate concurrence among multiple groups. Of greatest relevance to the evaluation, we used product-document analysis to identify the high-level requirements for a successful product, specifically the structure of ADCO plans. ADCO plans proved to be organized around units of human planning and execution. Note this is not the only possible plan structure but contrasts, for example, with plan structure organized to support allocation of resource streams (such as energy, space, or money) and constraint satisfaction among such continuous-valued limited resources. ADCO plans are organized temporally and hierarchically, with three nested levels of planning units: Increment, Activity, and Action. Increments are the largest unit of planning, defined by the interval between crew changes, and typically last months or weeks. Activities are carried out by ADCO (and its Russian counterparts) to support a high-level goal, such as docking an arriving vehicle or testing thrusters, and typically last hours. Actions are individually commanded changes to carry out an Activity that are sent to the ISS, affecting its control status, orientation, or trajectory, and typically last minutes or hours. Activities and Actions each have several types. Product-document analysis of versions also showed that time revisions were by far the most frequent, with revisions to the ISS attitude values (yaw, pitch, or roll) also frequent.

Some aspects of plan structure were already explicitly known, documented, and labeled. For other aspects, knowledge was more implicit, undocumented, and referred to indirectly with multiple overlapping terms. Some aspects of plan structure were very poorly supported in the legacy software used to build the plan product documents. For example, temporal relations, part-whole relations, and Activity level of events were poorly represented, nor did legacy software offer support for different event types. The legacy software was a form-based editor, providing editing operations on individual actions. Figure 1 illustrates the legacy system.

An existing planning software platform developed at NASA Ames was selected (SPIFe; McCurdy, 2009; McCurdy, Ludowise, Marquez, & Li, 2009) and prototype software developed to align better with the domain structure than had the legacy software. The prototype software provided better representations of and operations

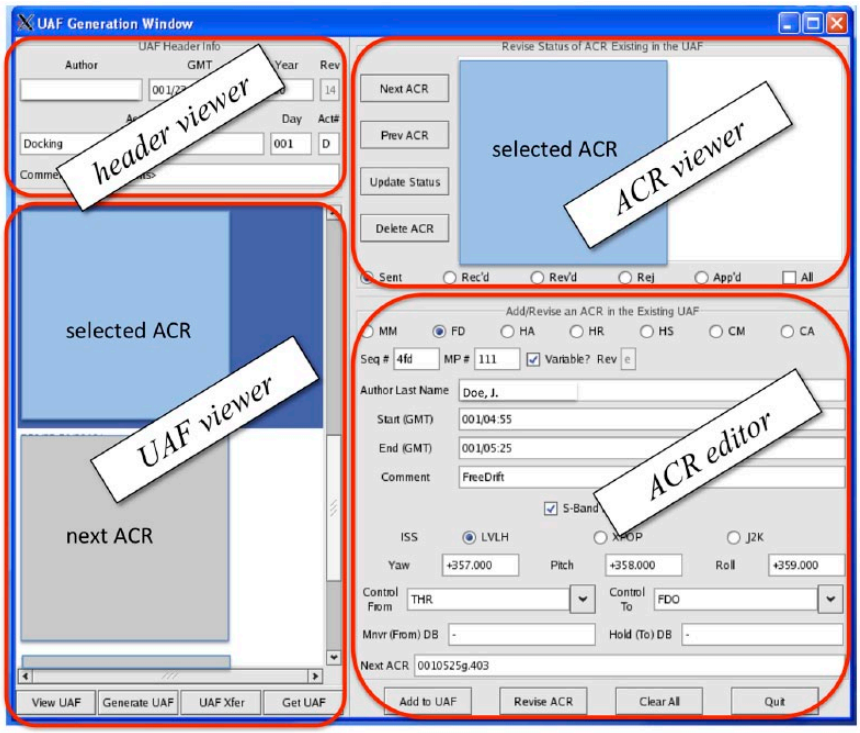

Figure 1. Screenshot of Legacy system showing the four main function panels for editing plans. The Actions attribute values shown here are invented, do not reflect a real event, but illustrate the types of individually possible values. Formatted content is occluded.

on (1) temporal relations, (2) part-whole relations, (3) the event levels and types, and (4) event attributes. Figure 2 illustrates the new system. The preceding companion paper shows the aspects of plan structure supported or neglected by the legacy and by the new prototype software, and the specific points where support differs; screenshots of the two systems are also shown. Overall, the new prototype software was better aligned with domain structure than the legacy software had been. This is schematically illustrated in Figure 3.

Because a key goal was improving software as much as feasible, many changes were introduced; many but not all of these were directly linked to improved alignment. Some changes (inherited from the software platform we selected), such as use of color, form of scrolling, or provision of an undo function, had little direct relation to improving alignment with the work structure. Key changes did directly improve alignment to the structure of work. These included representations and operations for temporal and part-whole relations and for the Activity level of events. The new software represented temporal order on a timeline and part-whole relations among events as graphically nested in an integrated timeline view. This change was global and would plausibly affect performance on most common uses of the software. Local improvements were also introduced. Specifically, the prototype system improved alignment of the representations and operations for Activities more dramatically than for Actions, because the legacy planning software provided essentially no representation of or operations on Activities. Activities were loosely and implicitly represented as files, which could be opened or closed at the level of the operating system, but no operations were available in the planning software, such as rescheduling or changing other values of an Activity. Differential improvement for Activities versus Actions proved valuable for discriminative evaluation.

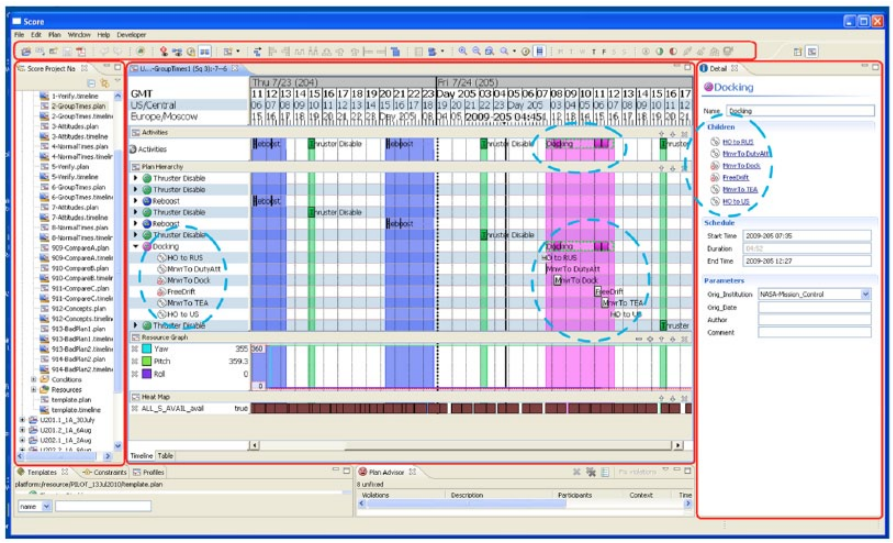

Figure 2. Screenshot of NEW system, showing the four main function panels for editing plans outlined in red. The example shows one activity expanded and selected. Dotted circles show four of the five possible repesentations of an Activity available in this design.

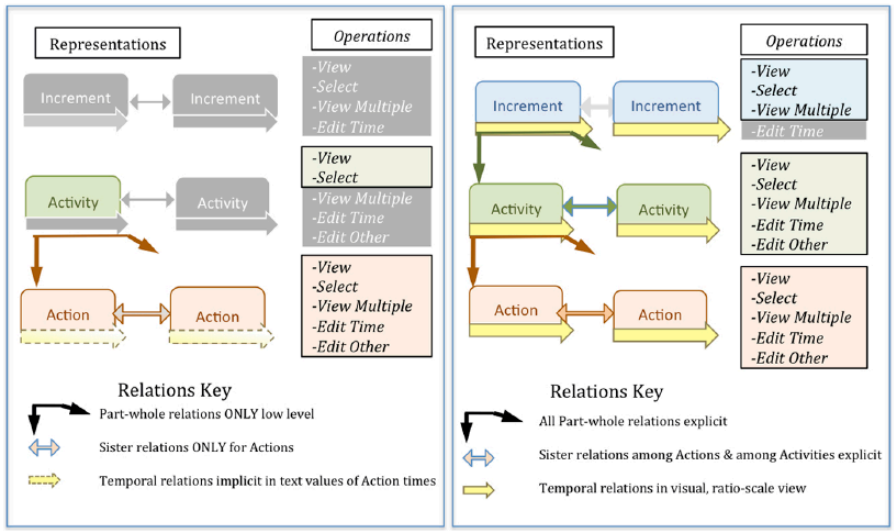

Figure 3. The left panel illustrates the Legacy Software and the right panel the New (SPIFe-based) Software. Shaded representations, relations, and operations indicate aspects of the domain structure that are not expressed in the software. Differences in relations are annotated in the key. Alignment of the two systems differs on the components grayed out in one but not the other. The redesigned prototype aligns much better with the domain structure (fewer gray components).

EMPIRICAL EVALUATION: GOALS, HYPOTHESES, AND THE ROLE OF NEEDS ANALYSIS

Our evaluation compared performance in Legacy versus New Software Conditions. First, we wanted to assess experimentally whether the New software in fact improved performance over the Legacy software, in a valid evaluation context. This addressed whether the prototype was a good design foundation for building operational ADCO software. Needs analysis helped us ensure validity of our experimental evaluation: The plan structure and the plan-revision tasks were similar to actual plans and actual tasks as identified in the needs analysis.

Second, we wanted to assess whether improved alignment of software to work structure was a factor contributing to performance improvement from Legacy to New software, should that be found. If so, this would in turn support the value of needs analysis methods that identify work structure and allow evaluation of alignment. Note that a global performance benefit of New over Legacy software provides only suggestive support for the importance of alignment, because the two designs differed in other respects as well. Therefore, we needed a more discriminative test to identify a specific impact of improved alignment. To do this, we constructed test items that differed in whether or not a specific increase in alignment affected performance on the item. If condition advantage is greater for the items benefiting from a specific increase in alignment more than does an otherwise comparable item that is not affected, such an interaction is attributable to the differential change in alignment. Increase in alignment from Legacy to New is much greater for Activities than for Actions. Thus, where comparable operations can be done on either an Activity or an Action, we should see larger performance improvements from Legacy to New when that operation is carried out with respect to an Activity than an Action. We identified two situations that allow this highly discriminative comparison and predict specific Software Condition × Item Type interactions. One concerns revision times, and the other concerns navigation: (a) Revising the time of an activity should show greater advantage in New versus Legacy condition than revising the time of an action. Further, in the New Condition, rescheduling an activity should be faster than rescheduling a proper subset of its actions, but in the Legacy Condition, rescheduling a proper subset of actions within an activity should be faster than rescheduling the activity. (b) Tasks that require changing focus to another action in a different activity should show greater condition advantage of New over Legacy than tasks that require changing focus to another action within the same activity.

We evaluated three hypotheses, detailed in the Methods section and summarized in Table 1.

Hypothesis 1: Faster (and, secondarily, more accurate) performance when working with the New Software versus the Legacy Software Condition, specifically in making revisions to plans, measured as main effects of software condition on response time (and error rates if sufficiently sensitive).

Hypothesis 2: Interaction between Condition and Item Type, such that items where alignment was most improved show greater condition benefit, measured as interaction effects between Software Condition and Item Type, on response time (and secondarily on errors); this will occur (a) for revision items that operate on Activity versus Action and (b) for revision items that require navigation across rather than within an Activity.

Hypothesis 3: Faster learning and better retention of New versus Legacy software, as a general benefit analogous to improved performance.

EXPERIMENT 1A: INITIAL LEARNING AND PERFORMANCE

Method

Design. The primary independent variable was the software system used (New versus Legacy Condition), varying between subjects. The experiment included four primary tasks assessing performance on editing tasks and several exploratory tasks assessing conceptual understanding. Within each of the editing tasks was a second independent variable of item type; these item-type variables are detailed after

TABLE 1: Overview of Hypotheses and Empirical Assessment Plan

	Hyp1: Performance Benefit From New Versus Legacy	Hyp2: Alignment a Factor in Benefit. Condition × Type Interactions.	Hyp3: Better Learning and Retention in New. Day × Condition Interaction.
Experiment 1A			
Revision: Verify	*Primary*	Secondary: Navigation Hyp 2b	Tertiary
Revision: Attitude	*Primary*	Secondary: Revision units Hyp 2a; Navigation Hyp 2b	Tertiary
Revision: Normal Times	*Primary*	Secondary: Revision units Hyp 2a; Navigation Hyp 2b	Tertiary
Revision: Group Times	Secondary	*Primary: Revision units Hyp 2a*	Tertiary
Experiment 1B			
All Revision Tasks	Secondary	Secondary	*Primary Retention*

introducing the tasks. The third independent variable was repetition Block (4 levels: 2 item sets repeated on 2 days); we also grouped the two blocks into Day 1 and Day 2 to assess retention. Item Type and repetition Block were within-subject factors. The primary dependent variables were response times and error rates. Condition is crossed with Item Type, for each task.

Participants. Graduate students and upper level undergraduates majoring in aeronautics/astronautics or in physical sciences/engineering were recruited from two local universities and from students interning at NASA. This participant population is broadly analogous to the incoming population of ADCO trainees. We allocated participants between conditions to balance level and department of degree. Nine participants ran in the Legacy and nine in the New Condition. Data from one New Condition participant was overwritten, reducing the condition's data sets to eight. Participants were recruited intensively within the time available for data collection; we had originally hoped to recruit 10 per condition.

There were four revision tasks, which tested our hypotheses, and several exploratory, conceptual tasks.

Four revision tasks. All revision tasks required entering changes to a plan or verifying plan information. The types of items for each of the four tasks are summarized in Table 2. Three tasks were designed primarily for validity to compare Legacy with New software in realistic tasks and to test Hypothesis 1 of overall advantage: (1) Normal Time Revision, (2) Attribute Revision, and (3) Verify. These tasks also provided supplementary data relevant to Hypothesis 2. Items that required operations on or navigation across activities should produce particularly large condition differences in performance due to greater improvement in alignment provided by the New software. The fourth task, Group Times Revision, was designed primarily to be discriminative, assessing whether increased alignment is a contributing factor to improved performance (should it be found). Group Times items differed in how much the group of events to reschedule capitalized on improved alignment, providing a test of Hypothesis 2a.

Items for each revision task were designed in two matched Sets, with users doing both sets in each day, thus not repeating the same item in a day. As appropriate for the task, we controlled or counterbalanced the matched sets on the following: (1) the navigation path from the entity in the prior item to the target entity to be revised, (2)

TABLE 2: Item Types Use in the Revision Tasks

Revision Tasks	Item Types in Task
Verify	
	Navigate to a different property within the same Action
	Navigate to a different Action within the same Activity
	Navigate to a different Activity
Attitudes Revision	
	Complex-Change Activity
	Complex-Multiple Values
	Simple-Diff Activity
	Simple-Diff Action
	Simple-Diff Action
Normal Times Revision	
	Activity-Shift
	Action-Shift: in-Different-Activity
	Action-Shift: in-Same-Action-Same-Activity
	Action-Shift: in-Different-Action-Same-Activity
	Context
Group Times Revision	
	Action Alone
	Activity
	Actions Within Activity
	Actions Spanning Activities

the nature of the revised entities (individual or variously grouped actions), (3) the magnitude and direction of the time change for temporal revisions, and (4) the specific property changed for revising nontemporal properties. Across all four tasks for one Set, users completed 61 revision problems; thus, our data came from 122 different revision items across the two Sets; both Sets were repeated from Day 1 to Day 2. Materials are described below for a single set.

Verify. The Verify Task of 21 items per Set included three practice items, was presented first, and sampled a variety of nontemporal attributes. The participant clicked a *match* or *no match* button, indicating whether the attribute value specified in the prompt matched the value in the plan. Item sequences were constructed to test the effects of finding, or navigating to, the value to be verified. The three Verification types differed in the required navigation from the previous item:

(v-1) Navigate to a different property within the same Action (3 items/block),
(v-2) Navigate to a different Action within the same Activity (9 items/block), or
(v-3) Navigate to a different Activity (6 items/block).

We predicted navigating to a new Activity would show the largest difference between conditions.

Attitude Revision. In each set, 13 Attitude Revision items consisted of four complex items (grouped into two types) and nine simple items (grouped into three types) that required changing the yaw, pitch, or roll. The two complex types changed multiple entities or values and

always required navigating to a new activity; the three simple types changed just one component of attitude (yaw, pitch, or roll) for one just one action, but required different navigation:

(ar-1) *Complex-Change Activity*: navigate to a different Activity, change one value for all Actions in the Activity (e.g., "change Pitch for all actions to 359.2");
(ar-2) *Complex-Multiple Values*: navigate to a different Activity, change multiple values;
(ar-3) *Simple-Diff Activity*: navigate to a different Activity, change a value;
(ar-4) *Simple-Diff Action*: navigate to a different Action in the same Activity, change a value; and
(ar-5) *Simple-Same Action*: navigate to a different attitude component within the same Action, change a value.

The Attitude Revision task primarily tests for general benefit of the prototype software (Hypothesis 1) in representative tasks and also for interaction effects of different navigation demands (Hypothesis 2b).

Normal Times Revision. In each set, the Normal Times Task consisted of 12 items revising times and three Context items revising nontemporal properties, used to change user focus and thus control the navigation needed. The four types of time revision items differed in the entity to be rescheduled, either an Activity (Type 1) or an Action (Types 2, 3, and 4). They also differed in changing focus to a new Activity (Type 1, 2, and 5), to a new Action with the same Activity (Type 4), or keeping focus on the same Action (Type 3).

(nt-1) The two *Activity-Shift* items shifted the time of an activity; only two items were included because piloting showed this took a long time in the Legacy Condition.
(nt-2) The four *Action-Shift: in-Different-Activity* items shifted the time of an action that also required navigating to a different Activity from the prior item.
(nt-3) The three *Action-Shift: in-Same-Action-Same-Activity* items shifted the time of an action already in focus from the prior item; requiring focusing on a different attribute, but remaining within the same Action.
(nt-4) The three *Action-Shift: in-Different-Action-Same Activity* items changed the time of a different Action within the Activity used in the prior item, requiring a focus change to a new Action but not new Activity.
(nt-5) The three *Context* items changed a nontemporal property and moved the focus to a particular Activity for use by the following temporal item.

The Normal Times Revision task primarily tested for general benefit of the prototype software (Hypothesis 1) and secondarily for interaction effects (Hypothesis 2). We predicted Type × Condition interaction showing larger condition differences on types that required navigating to a new Activity and larger condition differences for changing the time of an Activity versus an Action.

Group Times. In each set, there were 12 items, of four item types. Types differed in the extent that the items drew on improved alignment of the New Software. Types 1 and 2 operated on a single, meaningful unit, either an Action or Activity (analogous to Types 1 and 2 in the Normal Times Task). Types 3 and 4 operated on ad hoc collections of contiguous actions, either within one or spanning across two Activities. The four types, presented in the order listed, asked the user to shift the time of:

(gt-1) one action (*Action Alone*);
(gt-2) one *Activity*;
(gt-3) a collection of actions within one Activity (*Actions Within Activity)*; and
(gt-4) a collection of actions across two activities (*Actions Spanning Activities).*

Additional aspects of variation as plausibly contributing to difficulty were controlled: (a) All Group Times items targeted events that were in a different activity than in the prior item, so all required similar navigation; (b) within each of the three multi-action item types, we varied the number of actions affected (between 2 and 8) but balanced the average number of actions across types as closely as feasible, as shown in

the Appendix; and (c) the amount of time shifted (5 to 120 minutes) and earlier versus later direction of shift were evenly distributed.

The Group Times task was designed primarily to test whether alignment is a predictor of item difficulty, between and within conditions. Hypothesis 2a predicts specific interactions. Viewed as within-condition patterns, in the New Condition, Activity revisions will be similar to Action revisions (software is aligned with both meaningful units) and much easier than revisions to ad hoc collections of actions (within or spanning across an activities). In the Legacy Condition, Activity revisions will be much harder than Action revisions and similar to or slightly more difficult than rescheduling *Actions-Within-Activity* (because *Activity* Items had slightly more actions than did the *Actions-Within-Activity* items; see Appendix). Viewed as between-condition patterns, condition differences will be greatest for Activity revisions, smallest for Action revisions, and intermediate for Actions-within-Activity and Actions-spanning-Activity.

The conceptual tasks were both an opportunity for participant learning, and an assessment of participants' knowledge about ADCO plans. All conceptual tasks were exploratory, investigating whether knowledge and tasks beyond plan revision were affected by the software used.

Compare Plans. Participants used the software to compare pairs of plans. Questions focused on increment level (e.g., activities in common between two plans), on Activity (e.g., commonalities across Thruster Disable activities), on Action (identify actions that differ between two activities), or on Attribute (e.g., where does the Yaw/Pitch/Roll change). Questions included identifying where an activity could be inserted in a plan, rescheduling actions that did not have the required communication coverage, identifying the points where mass properties changed, or actions that had a particular approval status. Participants typed answers into a text file questionnaire. Analogous sets were used for Day 1 and Day 2.

Plan Reference. Participants viewed a plan with the software to answer questions that required checking and reasoning across multiple aspects of a plan. This task is not scored and results are not reported.

Problem Finding. Problem Finding assessed recognizing problems in plans presented in the software. Participants were asked to find and note all aspects of a plan that were errors, anomalies, or in some way discrepant. This task was given on Day 2 only, as the last task using the software, and had two parts based on two plans. Each plan had 10 errors. The first plan introduced erroneous attribute values, and the second had errors in the order, spacing, or inclusion of actions in activities. Participants described each problem they found on a document numbered 1 to 10. Roughly half way through the experiment, we began limiting users' time on each part to 10 minutes, to ensure sufficient time for remaining tasks.

Plan Knowledge. Participants answered questions about how plans should be organized, without using planning software or referring to any particular plan. Questions included what action should be done in a particular situation, which attributes would change values, and also why an action should be taken or the reason for a difference between action types.

Card Sorting. Six card-sort problems were used, three on Day 1 and all six on Day 2. The first asked participants to group 31 cards (labeling 14 Actions, 8 Activities, 9 other ADCO terms) into separate piles for Actions, Activities, and other terms. The second and third asked the user to move action cards under one of three activities and to sequence the actions correctly. The last three card-sorts asked users to produce the correct actions, sequence, and relative separation for a single activity. Participants selected and ordered action cards for each activity.

Procedure

Participants were scheduled for 7 hours on 2 days a week apart. The order of tasks is shown in Table 3. Morning, midday, and afternoon breaks were provided and hearty snacks offered. All tasks but orientation, usability interview, and debriefing took place on computer, and interaction was captured as screen video using Morae software.

TABLE 3: Sequence of tasks on Day 1 and Day 2

Tasks	Day 1	Day 2 (Week Later)
Introduction and Training		
Welcome, consent, and orientation	x	
Domain Tutorial with Questions	x	
Hands-On Training on software	x	
Revision Tasks {Verify, Group Times Revision, Attitudes Revision, Normal Times Revision}		
Set 1	x	x
Set 2	x	x
Conceptual Tasks		
Compare Plans	x	x
Plan Reference	x	x
Problem Finding		x
Plan Knowledge	x	x
Card Sorting [as time permitted]	x (1-3)	X (1-6)
Usability Interview		x
Debriefing about the study		x

Revision Tasks	Item Types in Task
Verify	
	Navigate to a different property within the same Action
	Navigate to a different Action within the same Activity
	Navigate to a different Activity
Attitudes Revision	
	Complex-Change Activity
	Complex-Multiple Values
	Simple-Diff Activity
	Simple-Diff Action
	Simple-Diff Action
Normal Times Revision	
	Activity-Shift
	Action-Shift: in-Different-Activity
	Action-Shift: in-Same-Action-Same-Activity
	Action-Shift: in-Different-Action-Same Activity
	Context
Group Times Revision	
	Action Alone
	Activity
	Actions Within Activity
	Actions Spanning Activities

Participants were asked to read general information about the ISS and the ADCO group prior to coming to the lab. Participants signed informed consent forms. The in-lab domain training began with review of the advance material, text, and graphics viewed on computer in six sections. At the end of each section, the participant wrote answers concerning control of the

ISS and ADCO plans for doing this; when the participant finished a question set, the experimenter provided corrections or elaborations as needed. Training on either tool introduced all functions of that tool to be used in the experiment. Teaching each aspect concluded with "now you do it" exercises. Introducing the New system required eight exercises versus five for Legacy.

The four Revision Tasks posed items and collected response times through MatLab with the tasks done using the planning software (Legacy or New). After the Revision Tasks, the participant did the exploratory Conceptual Tasks intended to develop as well as to assess conceptual knowledge about ADCO planning. We wanted to give our users a richer, more interesting set of tasks than just the revision actions, both to make the overall activity more similar to the actual work environment and to make it more engaging. Also, the additional tasks were a potential source of "on the job learning." These tasks were not extensively piloted, and many proved quite difficult.

The first three Conceptual Tasks (Compare Plans, Plan Reference, and Problem Finding) were done viewing plans in the software and writing answers in a text file. The General Knowledge and Card Sort tasks assessed knowledge without use of the planning software. For the Card Sort tasks, participants grouped electronic "sticky notes" on the computer screen.

The experiment finished with an interview and debriefing. The experimenter asked about the software, the tasks done in the experiment, and the strategies used. In debriefing, the experimenter explained the purpose of the experiment, discussed the project as dictated by participant interest, and completed payment paperwork.

Results

Analysis plan. Our Revision Tasks data sets had a number of characteristics that made analysis challenging, including nonsphericity, nonnormality, missing data due to high error rates, and outliers. We used the repeated measure approach for item Type within a task (rather than treating the types as multiple variables). For each task, we conducted a planned Condition × Type × Block mixed MANOVA. We use Greenhouse-Geisser adjusted values throughout our results (which adjusts for violations of sphericity) for a conservative, consistent approach. Thus, *df* (degrees of freedom) is always adjusted based on degree of nonsphericity. We supplemented this planned analysis as appropriate. SPSS was used for analysis, and R was used for some graphics.

Revision Tasks: Verify. Figure 4 shows results of the Verification Task. Verification is two-thirds the speed and has a 10th the errors in the New Condition (M = 13.0 seconds, SD = 2.87, errors = 0.2%) relative to the Legacy Condition (M = 21.2 seconds, SD = 8.66, error % = 2.0). A 2 (Condition: New vs. Legacy) × 3 (Type of Navigation: Navigate to a New Attribute within the Same Action, Navigate to a New Action within the Same Activity, Navigate to a Different Activity) × 4 (Block: 1, 2, 3, and 4) mixed MANOVA on correct responses showed significant effects of Condition, $F(1, 15) = 6.61$ $p = .021$, $\eta_p^2 = .306$; Type, $F(1.97, 29.59) = 55.86$, $p < .001$, $\eta_p^2 = .79$; and Block, $F(1.33, 20.01) = 19.70$, $p < .001$, $\eta_p^2 = .57$; significant interactions of Type × Block, $F(2.08, 31.19) = 3.82$, $p = .03$, $\eta_p^2 = .20$, and Condition × Type, $F(1.97, 29.59) = 9.49$, $p = .001$, $\eta_p^2 = .39$; but Block × Condition as well as Block × Condition × Type interactions were not significant.

Variability was greater and distributions positively skewed in the Legacy Condition. Supplementary tests (aggregating, partitioning, and transforming data, and using nonparametric methods) confirmed and extended the findings and showed Conditions differed at all time points for Navigation Type 3 (Navigate to a New Activity), at Blocks 2 and 3 for Navigation Type 2 (Navigate to a New Action within the Same Activity), but Conditions did not differ for Type 1 (Navigate to a New Attribute within the Same Action).

In sum, the primary source of the robust overall condition differences came from items requiring navigation to a new activity. Performance on items requiring verification of a different property within the same action, thus locating information already visible on the screen and close to a prior point of attention, was similar in both conditions.

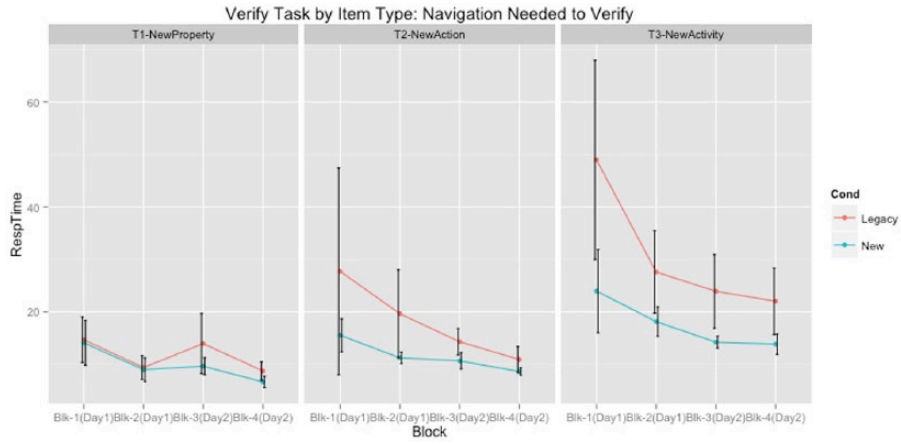

Figure 4. New versus Legacy comparison for the Verify Task across four Blocks. Average time of correct responses for the three Verify types. Panel a: Navigate to a New Attribute within the Same Action; Panel b: Navigate to a New Action within the Same Activity; Panel c: Navigate to a Different Activity. Error bars are 95% confidence intervals with normality assumed. Slower response times for Legacy than New Condition support Hypothesis 1. Larger condition difference for Type 3 than Type 2 supports Hypothesis 2b.

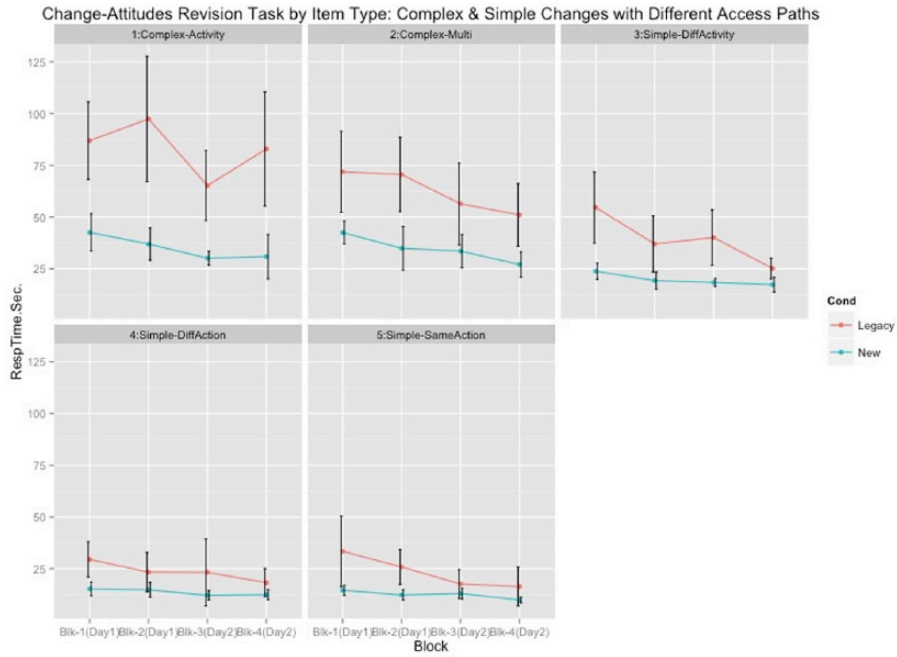

Figure 5. New versus Legacy comparison for the Attitude Revision Task across four Blocks. Average time of correct responses for the five Attitude types. Types 1, 2, and 3 require navigation to a new Activity, Type 4 to a new Action, and Type 5 only to a different property within the same Action. Slower response times for Legacy than New Condition support Hypothesis 1. Larger condition differences for Types 1, 2, and 3 versus Types 4 and 5 support Hypothesis 2b. Error bars are 95% confidence intervals with normality assumed.

Revision Tasks: Attitude. Revising attitude values with New (M = 21.3 seconds, SD = 2.77, errors = 2.6%) takes half the time and produces half the errors as with Legacy (M = 42.3 seconds, SD = 14.5, errors = 6.0%), as illustrated in Figure 5. A 2 (Condition) × 5 (Type of Attitude Change:

Complex-Change Activity, Complex-Change Values, Simple-New Activity, Simple-New Action, and Simple-Old Action) × 4 (Block) mixed MANOVA on correct responses showed significant effects of Condition, $F(1, 15) = 17.63$, $p = .001$, $\eta_p^2 = .54$; Type, $F(2.22, 33.25) = 112.15$, $p < .001$, $\eta_p^2 = .88$; Block, $F(1.84, 27.58) = 15.81$, $p < .001$, $\eta_p^2 = .51$; and significant interactions of Condition × Type, $F(2.22, 33.25) = 20.21$, $p < .001$, $\eta_p^2 = .57$; Type × Block, $F(4.61, 69.16) = 2.67$, $p = .033$, $\eta_p^2 = .15$; with Block × Condition marginal, $\eta_p^2 = .15$, and Condition × Type × Block, *ns*. Supplementary analyses supported and extended these findings. New was faster than Legacy across all item types. The difference was largest for the three types that require navigating to a new activity to make the revision (Types 1, 2, and 3) and for the Complex revisions (Type 1: revising an attitude value for an entire Activity, and Type 2: revising multiple values).

Revision Tasks: Normal Times. Performance with New ($M = 19.3$ seconds, $SD = 4.56$) takes half the time as with Legacy Condition ($M = 39.07$, $SD = 17.86$), as illustrated in Figure 6. Concerning errors, participants in the New Condition produced a third fewer errors (1.7%) than did Legacy participants (errors > 5.3%) with one error-prone Legacy user excluded; if included, the Legacy error rate would increase to 7.6%. Note that for the Legacy Condition, changing the time of an activity is much *slower* than changing time of an action, whereas for New the times are similar but changing an activity is slightly faster.

The 2 (Condition) × 5(Type: *Activity-Shift; Action-Shift -New-Activity; Change-New Action-in-Same Activity; Change-Same Action-in-Same Activity; Context*) × 4 (Block) mixed MANOVA on correct responses with one Legacy Condition user dropped due to missing data (a Block × Type cell with no correct responses) showed significant effects of Condition, $F(1, 14) = 10.15$, $p = .007$, $\eta_p^2 = .42$; Type, $F(1.16, 16.22) = 14.55$, $p = .001$, $\eta_p^2 = .51$; Block, $F(1.17, 16.35) = 10.24$, $p = .004$, $\eta_p^2 = .42$; and significant interactions of Condition × Type, $F(1.16, 16.22) = 12.16$, $p = .002$, $\eta_p^2 = .47$; and Condition × Block, $F(1.17, 16.35) = 5.67$, $p = .026$, $\eta_p^2 = .29$ (but only marginal, $p < .1$, Type × Block, $\eta_p^2 = .21$, and Condition × Type × Block, $\eta_p^2 = .18$). Note that dropping the excluded user has the effect of improving RT for the Legacy Condition. Consistent effects were found in a variety of supplementary analyses: contrasts, excluding the Context Item Type, dropping two Legacy users identified as outliers, aggregating across blocks, and nonparametric statistics.

These finding show large differences favoring the New over Legacy Condition (supporting Hypothesis 1) and the pattern of advantage reflected the points where New improved alignment with the plan structure relative to Legacy. The largest condition differences occurred when users needed to reschedule an activity (Type 1) rather than an action (Type 2), supporting Hypothesis 2a, and when revision required shifting focus to an activity different from the prior item (Types 1, 2, and 5) rather than staying with the activity (Types 3 and 4), supporting Hypothesis 2b.

Revision Tasks: Group Times. Overall, times for correctly completed Group Times items for New averaged 50.6 seconds ($SD = 8.51$), half that of the Legacy mean of 106.9 seconds ($SD = 32.3$). Condition distributions are shown in Figure 7. The Group Times task was even more difficult than we expected, producing high error rates: 11.5% for New and 26.4% for Legacy. Throughout, slower responses went with higher error rates, a speed/accuracy association not tradeoff. However, differential error rates complicated the analysis, as omitting times on the error trials selectively dropped the slower responses in the more difficult conditions; further, some users made errors on all three items of a type within a block, thus producing a response category with missing data for that user. Errors primarily consisted of selecting an incorrect event (or set of events) to reschedule or of rescheduling the time to an incorrect value. To address these data characteristics, we used a large ensemble of analysis methods, with highly convergent results.

Our first suite of analyses included the planned MANOVA for effects of Condition and interaction with Type, for correct responses and also for errors. Performance over time is shown in Figure 8. The 2 (Condition) × 4 (Type: *Action Alone;*

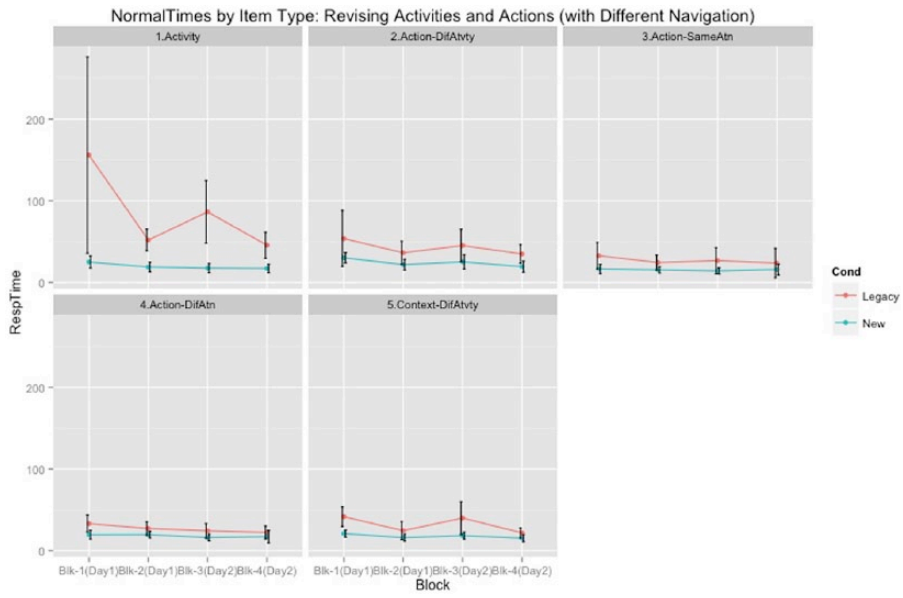

Figure 6. New versus Legacy comparison for the Normal Times Revision Task across four Blocks. Average time of correct responses. Type 1 items change time of a whole Activity and Types 2, 3, and 4 items change time of one Action. Types 1, 2, and 5 require shifting focus to a different Activity. Types 3 and 4 keep focus within the same Activity but shift focus to time from a different property of the same Action (Type 3) or shift focus to a different Action (Type 4). Error bars are 95% confidence intervals with normality assumed. Slower response times for Legacy than New Condition support Hypothesis 1. Larger condition differences for Type 1 versus Type 2 support Hypothesis 2a; larger condition differences for Types 1, 2, and 5 versus Types 3 and 4 support Hypothesis 2b.

Activity; Actions Within Activity; Actions Spanning Activities) × 4 (Block) mixed MANOVA for response time was based on only the six New and four Legacy Condition users, who had correct responses for each Type × Block level. Nevertheless, it found significant effects for Condition, $F(1, 8) = 11.28$, $p = .01$, $\eta_p^2 = .585$; Type, $F(1.73, 13.80) = 28.60$, $p < .001$, $\eta_p^2 = .781$; Block, $F(1.70, 13.60) = 16.47$, $p < .001$, $\eta_p^2 = .673$; and significant interaction of Condition × Type, $F(1.73, 13.80) = 6.62$, $p = .012$, $\eta_p^2 = .453$, with no other significant interactions. We analyzed effect on errors of 2 (Condition) × 4 (Type), finding significant effects of Condition, $F(1, 15) = 5.53$, $p < .038$, $\eta_p^2 = .272$; Type, $F(2.30, 32.2) = 8.97$, $p < .001$, $\eta_p^2 = .39$, and the Type × Condition interaction, $F(2.30, 32.2) = 3.84$, $p = .027$, $\eta_p^2 = .215$. Activity revisions were initially extremely time consuming for Legacy users. Supplementary analyses (aggregating across Block, partitioning by Type, nonparametric, analysis of all response times including incorrect responses) supported and extended these findings. Viewed from the perspective of within-subject differences, the pattern of relative item difficulty changed as predicted: Activity and Action revisions were similar in the New Condition, but Activity and Actions-within-Activity were similar in the Legacy Condition. The overall benefit of New over Legacy software is large and significant (main effects of condition, Hypothesis 1), and improved alignment contributes to these benefits (interaction patterns significant, Hypothesis 2a).

A second suite of analysis further tested the specific interaction predictions of Hypothesis 2a. We predicted that the improvement in performance from the Legacy to New software would be greater for the *Activity* items than for the *Action-Alone* items or for the *Actions-Within-*

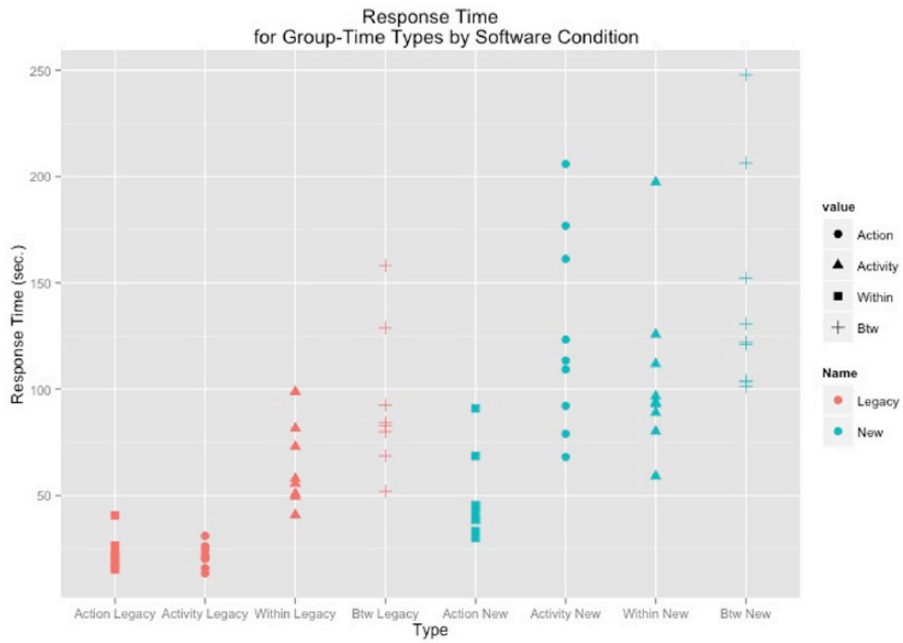

Figure 7. Response times of correct responses to Group Times items, showing differential patterns of difficulty between conditions. In the New Condition, revising an activity is similar to (and slightly faster than) revising an Action, with both much faster than revising a subset of Actions within an Activity. In the Legacy condition, revising an activity takes much longer than revising an Action, and longer than revising a subset of actions within an Activity. This differential pattern is a signature of differential alignment of software: For Legacy, operating on actions forming an activity are no better supported than operating on ad hoc collections of actions (Hypothesis 2a).

Activity items. We tested these difference-of-Condition-differences with two paired *t* tests, using the within-subject difference between the two types being compared. The New Software provided greater benefit for Activity items than for Action-Alone, $t(8.3) = 6.52$, $p < .001$, and for Activity items than for *Actions-Within-Activity* items, $t(15) = 4.28$, $p = .001$. The New Software also provided greater benefit for Activity items than for Actions-Spanning-Activity, $t(15) = 3.20$, $p = .006$. Thus, the relative benefit of the New software is predicted by the relative improvement in alignment, on a test where other factors were controlled to a high degree. These tests are the most direct and controlled assessment of the impact of improved alignment on performance: greatest benefit where greatest improvement in alignment.

A third analysis approach points to the different strategies available in the two software conditions, which underlie the pattern of interaction. The experimental design varied and counterbalanced the number of actions to be changed across the three multiaction items. For each individual, we correlated number of actions to be changed with response time for that item. Number of actions was a very strong predictor of response time for individuals in the Legacy Condition ($M\ r = .64$), much more than in the New Condition ($M\ r = .30$), $t(15) = 9.46$, $p < .001$; if only the three complex item types are included and Action-Alone Items dropped, the Legacy correlation is $r = .43$ versus New $r = .10$. This confirms that the number of actions the user must handle contributes much less to the overall item difficulty in the New versus Legacy Condition: for New but not Legacy users, a set of any number of actions that make up an activity can be operated on as one unit.

Revision Tasks: Learning and stability of performance. Hypothesis 3 predicted faster

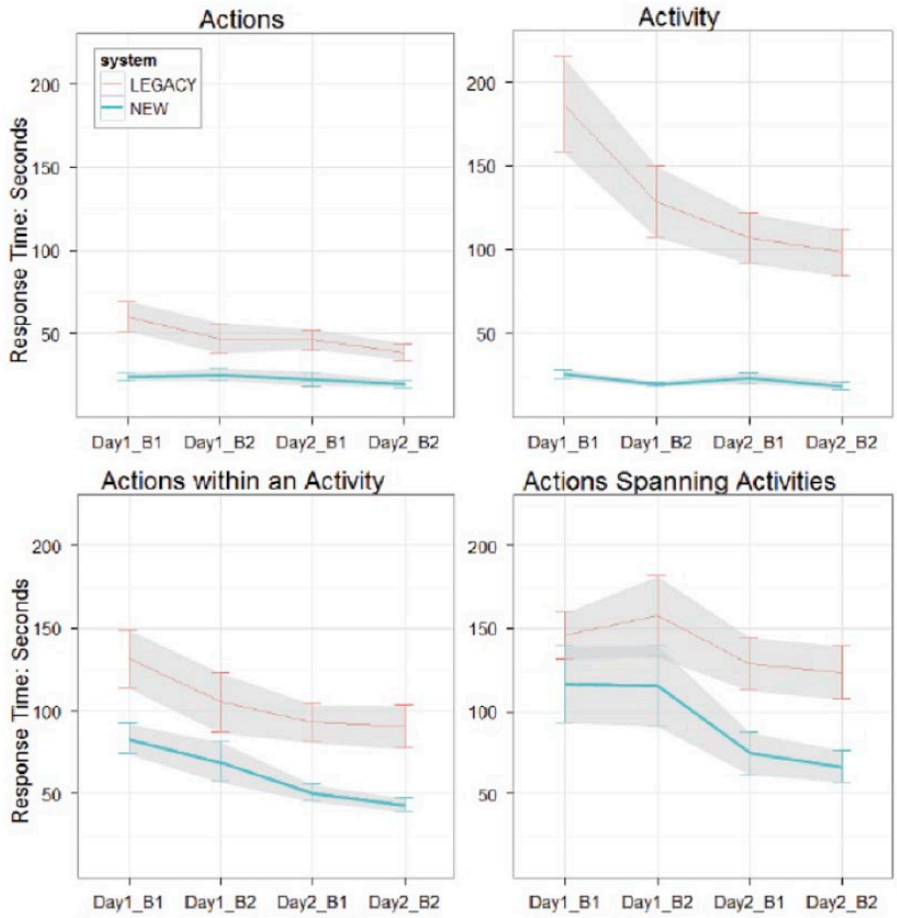

Figure 8. New versus Legacy comparison for the Group Times Revision Task across four Blocks. Average time of correct responses for the types. Panel a: 1-Action Alone; Panel b: 2-Activity; Panel c: 3-Actions Within Activity; Panel d: 4-Actions Spanning Activities. Error bars are 95% confidence intervals with normality assumed.

learning and better retention of the New rather than Legacy Software. Rather than showing a steeper learning curve in the New Condition, New participants did well from the first block. Whether this is because there was less to learn or because they learned within the first block, we cannot tell. Thus, we have no evidence for our prediction of faster learning for New rather than Legacy software. There seems to be more that must be learned in the Legacy Condition, resulting in large changes from Block 1 to Block 2 particularly for item types involving Activities.

To provide a direct and integrated test for learning or retention differences between conditions, we tested change from Day 1 to Day 2. For each of the four revision tasks, we derived an average correct response time by Day (weighting items not types equally). This eliminates any effects of Sets (which might differ in difficulty, despite our counterbalancing) and minimizes impact of missing data. The Day × Condition interaction, an overall measure of differential speed-up, was significant, $F(1, 15) = 5.12$, $p = .039$, $\eta_p^2 = .25$, in the 2 (Condition) × 2 (Day) × 4 (Tasks: Verify, Attitude, Normal Times, Group Times) mixed MANOVA, showing more speed-up for Legacy. The main effects of Condition, Day, and Task (Fs > 16, $p < .001$, $\eta_p^2 \geq .5$) and the interactions of Task × Condition ($p = .001$, $\eta_p^2 = .53$) and Task × Day ($p < .01$, $\eta_p^2 > .34$)

TABLE 4: Speed-Up in Seconds From Day 1 to Day 2 for Revisions Tasks

Type	Legacy	New
Verify	10.4	3.6
Attitude Revisions	13	4.5
Normal Time Revisions	6.5	2.5
Group Time Revisions	22.4	15.7

were also significant. Table 4 shows the average, within-individual speed-up from Day 1 to Day 2, illustrating the greater speed-up for Legacy than for New users. Our data do not provide evidence of forgetting or retention difficulties over the intervening week. The nature of change and stability of performance is addressed further in Experiment 1B, when several users returned after about 2 months of disuse.

Conceptual Tasks: Compare Plans. Performance was scored as the sum of positive points for correct information (24.5 and 25.5 points possible for Day 1 and Day 2, respectively) and negative points for incorrect information. Users in the New Condition were better able to make accurate comparisons between plans, scoring higher (average score = 20.12, SD = 2.40) than did those in the Legacy Condition (average score = 17.36, SD = 2.01). A 2 (Condition) × 2 (Day) mixed MANOVA found significant effects of Condition, $F(1, 15) = 6.68$, $p = .021$, $\eta_p^2 = .31$; Day, $F(1, 15) = 6.73$, $p = .02$, $\eta_p^2 = .31$; and their interaction, $F(1, 15) = 13.97$, $p = .002$, $\eta_p^2 = .48$. Condition differences emerged in Day 2, and t tests found effects for both intrusions and correct information as well as overall score. On Day 2, performance was impaired by users actively proposing inaccurate information, not simply failure to identify correct information; the Legacy Condition users averaged 4.9 intrusions compared to 1.0 in the New Condition.

Users in the New Condition completed the task in an average time of 19 minutes, compared to 27 minutes in the Legacy Condition. A 2 (Condition) × 2 (Day) mixed MANOVA on time-on-task found effects of Condition, $F(1, 14) = 6.12$, $p = .027$, $\eta_p^2 = .30$, but not Day or Condition × Day interaction. Time for one Legacy User on Day 2 was not recorded.

Conceptual Tasks: Problem Finding. Most responses reported 1 of the 20 intended problems across the two plans, and users identified about half. Some users identified additional incorrect details that we had not intended, occurring at a level we had not expected to be viewed. These were included as correct detections. Users produced some intrusions, either stating that an aspect of the plan was incorrect when it was unambiguously correct or stating a rule or regularity that was clearly false. Our overall score summed the number of problems correctly reported and subtracted the number of intrusions; other responses such as ambiguous comments were not included in the scoring and were infrequent. Scoring was done with condition identification removed, though the content of some reports provided information sufficient to identify the user's condition.

Users in the New Condition scored higher than did those in the Legacy Condition with a mean of 12.9 (SD = 2.95) versus 9.7 (SD = 2.43), $t(13) = 2.24$, $p = .043$, $\eta^2 = .28$. The t test values should be treated as a heuristic guide to identifying the more reliable patterns. Two users who did not do the second plan were not included in the t test. The pattern of better New Condition performance held across finer partitions of the data (by plan or by type), but only the aggregated score was significant.

Several qualitative patterns emerged that may show the impact of the system used. First, five "false rules" were stated by users in the Legacy Condition, but none in the New Condition, hinting that Legacy Condition users might be developing fragments of inaccurate domain models. Second, both groups produced statements that were not clearly wrong, but were observations about patterns judged unusual. Within this type of assessment, New Condition users provided eight varied observations about expected sequences of Activities, whereas Legacy Condition users provided only two observations about Activity relations, both of the same pattern; perhaps, the timeline view made relations among

activities more prominent to users in the New Condition. Finally, in the New Condition, the timeline was displayed showing points where S-band communication was needed but unavailable (these were the basis for detecting the additional correct, but unanticipated errors). Seven users reported these as plan errors. This inadvertently illustrated the value of this visualization for plan checking; the items had been designed for display with constraint checking off, but when on, users noted the constraint violations.

Conceptual tasks performed without using the software: Plan Knowledge and Card Sort. Neither the Plan Knowledge Task nor the Card Sort tasks showed any effect of Condition, and performance was variable and generally poor. Overall performance on the Plan Knowledge Task was 50% correct, with the Legacy Condition scoring insignificantly better, averaging 15 points versus 13 points in the New Condition ($p < .3$). Condition scores remained very similar and did not differ when data were divided by day or question type. Time taken was very similar as well: 16 min for Legacy and 14 min for New.

Performance on the Card Sorts 1-3 was very similar between conditions but improved from Day 1 to Day 2 (from a score of 15.79 to 21.52). A 2 (Condition) × 2 (Days) × 3 (Sort Problem: Terms, Sequence 1, Sequence 2) mixed measure MANOVA found significant effects of Day, $F(1, 14) = 20.38$, $p < .001$, $\eta_p^2 = .59$, but no effect of Condition or its interaction with Day or Sort Problem. Sorts 4-6 (Day 2 only) were not analyzed statistically because all scorings were very similar and several subjects did not have time to complete these sorts.

EXPERIMENT 1B: RETENTION AND CROSS-TRAINING

Hypotheses

We predicted (1) continued performance differences between the Legacy and New software, after an extended retention period; (2) decreased performance after this retention interval; and (3) less decrease for the New than Legacy Condition. We also predicted the same software advantage would be found within-subject.

Method

After an average of 8 weeks, four Legacy users and three New Condition users who did relatively well in Experiment 1A returned for additional tasks on "Day 3." The users began working with the familiar system and then were crossed-trained on and worked with the software unfamiliar to them. Cross-training also provided information on transfer patterns, but these are not addressed here.

For the core revision tasks, participants used the familiar software to complete Set 1 Normal Time Revisions, then Set 2 Group Time Revisions, and then a set of a new Mixed Revisions Task. Following this, they were trained on the unfamiliar system and worked with that system to complete Set 2 Normal Time Revisions, Set 1 Group Time Revisions, and a repetition of the same Mixed Revisions set. The Mixed Revisions task consisted of 12 items changing event times and 9 items changing nontemporal attributes of events. Because we had such a small number of users, we expected that our analysis would be primarily descriptive. However, within-subject comparisons and continued large differences proved to produce statistically reliable effects.

Results

Continued effects of Condition and Item type. For the seven returning users on Day 3 (four used Legacy on Days 1 and 2; three used New), the responses to the revision tasks continued to be faster with the New than Legacy software: for Group Times, 37.0 versus 89.0 seconds; for Normal Times, 23.0 versus 59.3 seconds; and for Mixed Block, 29.6 versus 70.1 seconds. For Day 3, Software Condition is a within-subject factor: each user contributed for Legacy and for New, one block of data for each task (Group Times, Normal Times, and Mixed Block). Day 3 data for the Group Times task is shown on the right side of Figure 9, with each user's response on Days 1 and 2. For each task, a Software Condition × Type mixed MANOVA showed significant effects of Condition, Type, and Condition × Type interaction. Large benefits

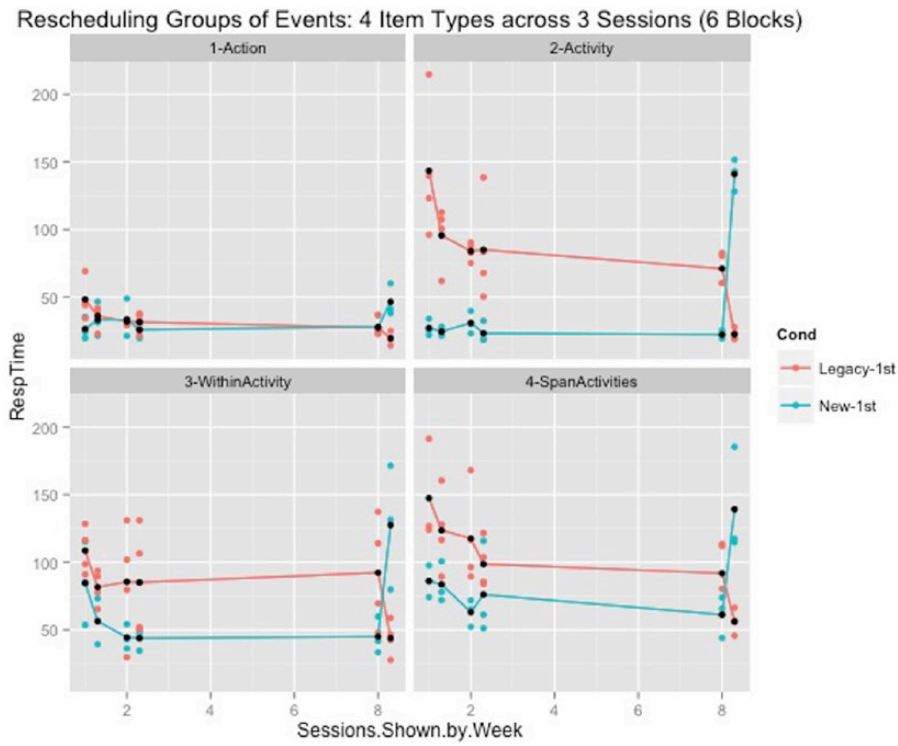

Figure 9. Correct response times for the seven users who returned 8 weeks after the initial two sessions, with group means in black. On the third session, users began with the system they had learned initially and then switched to the alternative system. The crossover at 8 weeks shows the within-subject comparison as users switched from New to Legacy or from Legacy to New. Users switching to the New system perform comparably to experienced New users, but users switching to the Legacy system perform worse than experienced Legacy users (Legacy remains difficult despite experience with New).

of the New software remained after additional retention time; the Condition × Type interactions for the three tasks followed the patterns of Experiment 1A—namely, greater benefit of the New software for items requiring operations on Activities and navigation across Activity boundaries.

Performance after retention. Performance on the Group Time Items allowed a clean comparison between performance using the familiar software on the identical set, in the first block of Day 3 and in the last block of Day 2, 8 weeks earlier. There was no evidence of forgetting in either Condition, with both groups producing nonsignificant improvements, dropping from 42.2 to 39.0 seconds in the New Condition and from 75.0 to 70.6 seconds in the Legacy Condition. We found no evidence of poorer retention in either condition.

DISCUSSION

Key Evaluation Results

Our first evaluation goal was to assist ADCO as much as feasible by assessing benefits of the new planning prototype for the target scope of work. Our first hypothesis, that the New software would support more effective performance than would the Legacy software, was dramatically supported at the level of general evaluation. The New software reduced completion times and errors by half. Supported by our findings of dramatic improvement, the ADCO

group and its management were able to argue successfully for development funding. Much more extensive planning software has been developed and is recently operational.

Our second goal was designing a discriminative evaluation to assess whether improved alignment of software with the domain structure was a contributing factor to improved performance. Because the systems we compared varied on many factors, discrimination required more specific predictions and findings than overall benefit, as in Hypotheses 2a and 2b. We designed items that differentially capitalized on increased alignment of the New Software, specifically (a) revisions to Activities versus Action and (b) navigation across Activity boundaries in carrying our plan revisions. We predicted and found strong interactions, with greater advantage of the New versus Legacy system for items that operated on Activities as a unit and for items that required navigating across Activity boundaries as well as corresponding differences in relative item difficulty within each condition.

The differences we found were substantial. However, investigating the same issues with larger sample sizes and with different technology will be important for establishing the generality—across systems and users—of the patterns of performance found here.

Implications for Needs Analysis and for Evaluation Methods

Evidence that a better-aligned system produces much better performance is important. Such demonstrations increase evidence for the benefits of needs analysis. In turn, an improved cost/benefit ratio further argues for the value of conducting a needs analysis. The two systems we compared differed in many ways, and thus the overall contrast is not a clean measure of the benefits of alignment, as are the interaction effects around operations on Activity units. However, because so much of the difference between Legacy and New software did concern differences in alignment, we think it likely to be a major contributor to overall differences in performance.

Methods for discriminative evaluation are important. Identifying a contributor to design outcomes is useful because the identified factor may then be tried in other contexts. The most direct test of whether a particular factor impacts performance is comparison of two systems differing only in that factor; a single improvement might be added to a legacy system, or a "reduced" version of the improved system might subtract out this improvement. Either way, development resources must be spent to implement the comparison system, and a different comparison system is needed to evaluate each feature of interest; further, additional evaluation conditions would be needed for each feature.

We believe that it is possible to tease apart and identify contributing factors when comparing systems that differ in multiple ways, when these factors can be selectively engaged and discriminative performance found across the systems being compared. Such comparisons between systems that differ in many respects are often important because it is infeasible to develop and test the many possible designs of interest (Roth & Eggleston, 2013). We developed sets of test cases, most explicitly for the Group Times Task, which selectively engage a factor of interest. In our case, the factor of interest was increased alignment of the New system with elements of the work domain identified in the Product-Document part of our needs analysis.

Our experiment provides suggestive but indirect support for our product-document analysis method. Knowing the contribution to good outcomes of a method (e.g., for representing work needs) is of even broader value than knowing the contribution to good outcomes of a particular feature. However, direct assessment of a method is even more difficult and expensive than direct assessment of a software feature. A direct test of the relative effectiveness of a target method would require comparing the merits of many designs relying on that method with the merits of many designs relying on an alternative method, a very resource-intensive approach. Because resources are always limited, initial support for a method can come from a case study that applies a method and produces a successful outcome. Also of interest are evaluations of intermediate outcomes of alternative analysis methods. Jamieson, Miller, Ho, and Vicente (2007) identified differences in requirements identified by work domain analysis and by task analyses.

CONCLUSIONS AND FUTURE RESEARCH

Our study provides preliminary evidence that improved alignment of software with the work it is intended to support increases its effectiveness and that needs analysis methods including product-document analysis are useful for our domain. More direct assessment of the impact of improved alignment of a socio-technical system with the work it is intended to support will be very important, as well as development of more standard representations of work, at a level suited for both design and evaluation. Generalization across work domains, types of work functions, and assessment methods will be needed. For example, assessment of the impact of alignment on ease of learning, and the relation between performance by system novices and system experts, each with domain expertise, will be very valuable (for related investigation, see Burns, Warren, & Rudisill, 1986). Our findings raise questions for future research about the conditions where improved alignment will substantially benefit performance. In addition, future research should explore what needs analysis methods, in what conditions, best support improving alignment.

APPENDIX: Group Times Items

	Activity			Actions Within Activity			Action			Actions across Activities		
Trial	1	2	3	4	5	6	7	8	9	10	11	12
Block 1	6	2	5	4	2	3	1	1	1	6	5	3
Block 2	5	2	8	4	2	4	1	1	1	5	4	5
Ave #Act		4.67			3.17			1			4.67	

Group Times Items: *Activity* trials shift an activity. *Actions-in-Activity* shift a subset of actions in an activity. *Action* shifts one action. *Actions-across-Activities* shift consecutive actions running across activities. Cell values are number of actions in each item to be shifted, with average for each type of trial listed below. Trial shows item order.

ACKNOWLEDGMENTS

Parts of this research project were reported in conference proceedings (Billman et al., 2011; Billman, Feary, Schreckenghost, & Sherry, 2010). This research was supported by funding from NASA Human Research Program: Space Human Factors Engineering (466199.02.01). We thank Erin Reed and Tarik Ward, the ADCO experts who worked with us, taught us, and checked materials; Steven Hillenius, Melissa Ludowise, Mike McCurdy, and the SPIFe development team for help with the software; and Emilie Roth and Gudela Grote for comments on an earlier report of this work. Author order after the first is alphabetical.

REFERENCES

Billman, D., Arsintescu, L., Feary, M., Lee, J. C., Smith, A., & Tiwary, R. (2011). Benefits of matching domain structure for planning software: The right stuff. In *Proceedings of the SIGCHI conference on human factors in computing systems* (pp. 2521-2530). New York: ACM Press.

Billman, D., Feary, M., Schreckenghost, D., & Sherry, L. (2010). Needs analysis: The case of flexible constraints and mutable boundaries. In *Proceedings of the SIGCHI conference on Human factors in computing systems* (pp. 4597-4612). New York: ACM.

Billman, D., Feary, M., Schreckengost, D., & Sherry, L. (this issue). Needs analysis and technology alignment method: A case study of planning work in an International Space Station Controller Group. *Journal of Cognitive Engineering and Decision Making*.

Burns, M. J., Warren, D. L., & Rudisill, M. (1986). Formatting space-related displays to optimize expert and nonexpert user performance. In *Proceedings of the SIGCHI conference on human factors in computing systems* (pp. 274-280). New York: ACM Press.

Jamieson, G. A., Miller, C. A., Ho, W. H., & Vicente, K. J. (2007). Integrating task- and work domain-based work analyses in ecological interface design: A process control case study. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, *37*(6), 887-905. doi:10.1109/TSMCA.2007.904736

McCurdy, M. (2009). Planning tools for Mars surface operations: Human-computer interaction lessons learned. *IEEE*, 1-12. doi:10.1109/AERO.2009.4839639

McCurdy, M., Ludowise, M., Marquez, J., & Li, J. (2009). *Space human factors engineering report: Crew scheduling lessons learned*. Moffett Field, CA: NASA Ames Research Center.

Roth, E. M., & Eggleston, R. G. (2013). Forging new evaluation paradigms: Beyond statistical generalization. In E. S. Patterson & J. E. Miller (Eds.), *Macrocognition metrics and scenarios design and evaluation for real-world teams*. Surrey, UK: Ashgate Publishing.

Dorrit Billman (PhD, Psychology, University of Michigan) is a research scientist with San Jose State University at NASA Ames Research Center, investigating how software and automation support complex cognitive work. Before working at NASA she was on the faculty at Georgia Institute of Technology and the University of Pennsylvania.

Lucia Arsintescu is a senior research psychologist with San Jose State University in the area of research on the effectiveness of preventative interventions and countermeasures that enhance human capability, and she is part of the Fatigue Countermeasures Group at NASA Ames. She received her medical degree from School of Medicine, Bucharest, Romania in 1996 and her master's degree in Experimental Psychology from San Jose State University in 2003. For the past several years, she has been engaged in research investigating the performance effects of sleep deprivation in operational environments.

Dr. Michael Feary is a research scientist in the Human-Systems Integration division at NASA Ames Research Center. He received his PhD in Human Factors Engineering from Cranfield University, UK (2006). He has focused on the development of tools to support design and analysis of Human-Automation Interaction in complex, safety critical systems. He has more than 45 publications on this topic.

Jessica Lee (BA, Stanford University, 2010) was a Research Associate with the San Jose State University at NASA Ames Research Center. She has built cognitive models and designed complex planning software for ISS mission operations. She most recently worked at an e-commerce startup in Indonesia implementing agile software practices and building out the product management department.

Rachna Tiwary (MDes, Product Design, Indian Institute of Science, Bangalore, India 2007) was a Research Associate with San Jose State at NASA. She worked on a comparative usability assessment of complex planning software designed for regulating the movement and orientation of the ISS. She currently works as User Researcher for a consultancy firm conducting qualitative research for enterprise- and customer-facing applications, with emphasis on mobile applications.