Research Paper

Needs Analysis and Technology Alignment Method: A Case Study of Planning Work in an International Space Station Controller Group–Part 1

Dorrit Billman, San Jose State University @ NASA Ames Research Center, **Michael Feary**, NASA Ames Research Center, **Debra Schreckenghost**, TRACLabs, and **Lance Sherry**, George Mason University

Our research (reported in two parts) improved software for a NASA Mission Controller group for the International Space Station and provided evidence for a key factor we believed contributed to the expected improvement. This factor is the degree of alignment of the technology to the structure of the work it is intended to support, or its fitness-for-purpose. This paper, Part 1, reports our needs analysis and software redesign, which provide specific and more general contributions. The specific contribution was new prototype software for planning work of the Attitude Determination and Control Officer group, who schedules trajectory and orientation changes of the International Space Station, with its Russian counterparts. The general contribution was a new needs analysis method, *product-document analysis*, a general design-process benefit. Product document analysis is a method complementary to task analysis and work domain analysis. Our needs analysis method characterized the high-level structure required of acceptable plans in terms of the plan components and their relations and constraints. The redesigned software was better aligned with the structure of work, as captured by needs analysis. We discuss conditions when product-document analysis may be useful.

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. 1–17
DOI: 10.1177/1555343414567774

Good design is the foundation of effective systems, but many systems lack good design. Indeed, poor design, particularly of human-system integration, is a key contributor to many accidents in high stakes and safety critical domains (Ellis, 2000; FAA Human Factors Team, 1996; French Inquiry Commission, 1994; Institute of Medicine: Committee on Quality of Health Care in America, 2000; Leveson, 1995; Maris, Dulac, & Leveson, 2004; Sarsfield, Stanley, Lebow, Ettedgui, & Henning, 2000). Good design depends on an accurate characterization of the purposes that the designed system is intended to support. Without this, there is little chance that the system is fit-for-purpose—that is, that the right problem has been addressed. Although we do not know of studies directly assessing inadequate characterization of the needs that the system should address, there have been studies of the impact of poor requirements. Poor requirement specification is a key contributor to poor design in software, the domain where this has been most studied (Charette, 2005; Emam & Koru, 2008; Hofmann & Lehner, 2001; Leffingwell, 1997; McLeod & MacDonell, 2011; Nasir & Sahibuddin, 2011; Procaccino, Verner, & Lorenzet, 2006; Savolainen, Ahonen, & Richardson, 2012). The value of understanding the needs and context of work for designing Human-System Integration has also been noted by the National Research Council (Pew & Mavor, 2007). Efficient and effective needs analysis methods are very valuable.

Our work extends and develops needs analysis methods to guide design and evaluation. Several established approaches are in use, some with many variations. Task analysis focuses on the steps in a process for accomplishing a task

(Kirwan & Ainsworth, 1992). Where a redesign or novel design aims to transform the way work is done, task analysis may be unduly tied to current procedures, and this issue was one motivation for developing Work Domain Analysis. Work Domain Analysis (within Cognitive Work Analysis) focuses on enduring constraints and relations in the work environment (Vicente, 1999). It has primarily been applied to process-control domains, such as operation of chemical or electrical plants, where the boundaries of work and its constraints are fixed. Contextual Inquiry has been primarily applied to information work and relies on an analyst's observation of users at work (Beyer & Holtzblatt, 1997). It provides a collection of work representations to capture an analyst's observations in a form that can be shared with users and used to guide design. Work Centered Design (WCD) refers to integrative approaches (Butler & Zhang, 2009; Eggleston, 2003) that have also been applied to information work. WCD approaches explicitly represent the entities, relations, and structure of the information work (in ontologies).

Our key motivator is to guide design to ensure fitness-for-purpose, or alignment of technology with the structure of work to be accomplished. Structure of work includes the components of what needs to be done and the organization of these components.

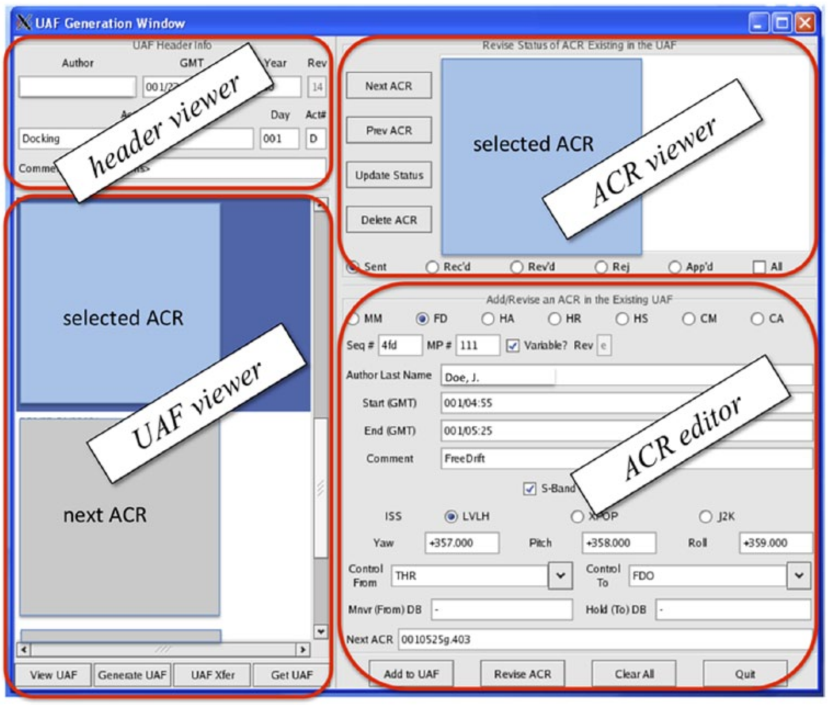
We did not find a method that exactly matched our conditions and domain (characterized below). Accordingly, after applying a form of task analysis, we developed a method that analyzes the products of information work, *product-document analysis*. We present this method in the context of our application. Product-document analysis is suitable for complex, technical, information work, where the important structure of work may not be obvious from observation and where access to experts is limited. We use *structure of work* primarily to refer to the core entities, relations, operations, and constraints in the work domain; constraints and operations bear on work *process*, but we intend "work structure" in broad contrast to detailed or arbitrary aspects of work flow or specific choices of how to get things done—that is, "work process." Specifically, we hope to pick out those stable aspects of work that should endure across (the relevant degree of) technology change.

This paper reports the first part of our case study of planning work for a NASA group in Mission Control, the Attitude Determination and Control Officers (ADCO). Overall, the case included needs analysis, redesign, and evaluation. The study gave us the opportunity to apply and develop needs analysis methods and to select and develop a new prototype software that was better aligned with ADCO planning needs. We had both a specific, pragmatic goal (to develop and test a prototype for the Mission Control of the International Space Station [ISS]) and an abstract, methodological goal. On the specific level, we aimed to conduct a needs analysis that led to a better design for ADCO software. On the general level, we aimed to develop needs analysis methods that produced useful results without requiring a great deal of time, particularly time of the domain expert. We aimed to develop methods that increase the degree of alignment of the technology to the structure of the work it is intended to support. Although the importance of alignment may be intuitive, how work should be analyzed for this purpose is not well understood.

This paper is organized into the following sections: (1) an overview of the ADCO work domain to orient the reader, (2) our needs analysis method and resulting analysis of part of ADCO's planning work, (3) the software prototype informed by the needs analysis, and (4) the circumstances where our method may be helpful, its possible benefits and limitations, and future directions. The second, companion paper (Billman, Arsintescu, Feary, Lee, & Tiwary, this issue) provides the results of a discriminative evaluation comparing performance of the prototype versus legacy software. The results from the experiment found that the prototype provided large improvements to accuracy and speed, and we analyze some sources of the benefit.

ADCO IN NASA MISSION CONTROL

We provide, as an orientation for the reader, a broad description of ADCO work; some parts of this description emerged as the result of our needs analysis and some we understood initially. The ADCO group is part of the



UAF Header Info

Author	GMT	Year	Rev
	001/22		14

	Day	Act#
Docking	001	D

Comme

Revise Status of ACR Existing in the UAF

[✓ Sent] Rec'd Rev'd Rej App'd All

Add/Revise an ACR in the Existing UAF

MM [✓ FD] HA HR HS CM CA

Seq # 4fd MP # 111 [✓ Variable?] Rev e

Author Last Name: Doe, J.

Start (GMT): 001/04:55

End (GMT): 001/05:25

Comment: FreeDrift

[✓ S-Band]

ISS [✓ LVLH] XPOP J2K

Yaw +357.000 Pitch +358.000 Roll +359.000

Control From: THR Control To: FDO

Mnvr (From) DB: - Hold (To) DB: -

Next ACR: 0010525g.403

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.

NASA Mission Control for the ISS, in Houston. ADCO, in partnership with its Russian counterparts, determine and control the orientation and trajectory of the ISS. Most frequently, they change the way the ISS is oriented in its orbit (the attitude), but they also are responsible for orbit changes, such as *reboosting* the ISS back up into a higher orbit. American and Russian controllers work in tight collaboration, as the mechanisms for changing position and attitude are split between Russian systems (the thrusters) and American systems (gyroscope-controlled rotational momentum management). They draw on intensive support from engineering groups for modeling and deriving technical specifications. ADCO operators work "on console" for commanding the software on-board the ISS to make attitude changes, and ADCO also builds its own plans for attitude changes, which guide the real-time execution. ADCO flight controllers do extensive planning, collaborating internally within the ADCO team and across teams of other flight-controller groups in Houston and in Moscow. For example, support for the arrival and docking of the Space Shuttle required a number of actions to check and align the ISS. One component of this was applying torque with the Momentum Management System, for a specific duration at a specific time, to achieve a specific new yaw, pitch, and roll setting of the ISS in orbit. The initial specification and every change to every component of an action require approval by both Houston and Moscow controllers. Plans are detailed and modified over time, so planning typically requires scheduling and rescheduling each event multiple times. The "ACR editor" panel in Figure 1 shows fields for "Start (GMT)" and "End (GMT)" where a revised time would be entered. ADCO integrates and coordinates input from and distributes notifications to multiple groups. The ADCO specialists characterize their job as consisting of on-console execution of real-time operations, of training other operators, and of operations planning. Our research addressed a subset of their planning functions.

ADCO flight controllers felt the software they used in planning could be improved to better support distributed planning. In particular, a key tool used in exchanging and revising plans to gain concurrence with the Russians was a form-based text editor with formatting support (the Legacy Software). Figure 1 provides an annotated screen shot. This editing software was experienced as very inefficient and error-inducing. Specially formatted text files, called UAFs, or Unified ACR (Attitude Change Request) Files, were passed back and forth as plan refinements or revisions were made, and each action specified in the UAF went through a multistep checking and approval process. Controllers used the form editor to carry out this planning and scheduling activity. Figure 1 shows how ADCO would view and revise values in a "Free Drift" event, for example, by changing the system in control at the beginning of the event ("Control From") to "THR," indicating Russian thrusters are responsible for maintaining orientation.

Members of the ADCO group wanted to improve the software and had begun specifying interface improvements. An ADCO controller sought out human factors support from our research group, which produced the opportunity for our case study. We began by explicitly identifying work needs. This led to planning software with new functionality and architecture, not just updating the interface of the existing system.

Figure 2 illustrates part of the plan structure identified in our analysis. This illustration is a snippet of a screen shot from the new software we developed, based on our product document analysis and presented in Part 2. The illustration shows a docking event made up of several events including handovers, which change locus of control, and maneuvers, which move or reorient the ISS. ADCOs schedule different types of events in accord with different requirements for duration and for separation or contiguity with other events; this Maneuver to Torque Equilibrium Attitude requires a half-hour and begins at the end of Free Drift. Planners build into the plan details of engineering parameters such as which system is controlling the ISS position at the beginning and end of an event and what the values are for yaw, pitch, and roll at beginning and end. Planners also track and enter approval and authoring status as a plan is revised.

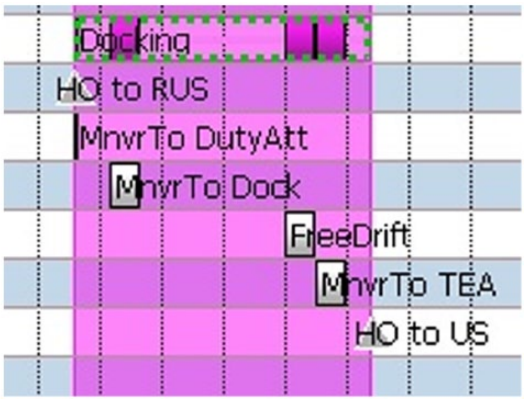


Figure 2. Illustrations of the structure of a plan fragment, showing a larger docking event and and its components. (Taken from a screen shot of the new software.)

NEEDS ANALYSIS PROCESS AND RESULTS

Overview

Our needs analysis adapted and extended existing methods, to suit the resources and limitations of our situation. Throughout, our more structured methods were supported by vital, though limited, informal interactions with experts. Our needs analysis drew on methods and representations from the task analysis (Diaper & Stanton, 2004; Kirwan & Ainsworth, 1992; Schraagen, Chipman, & Shalin, 2000) and contextual inquiry traditions (Beyer & Holtzblatt, 1997).

We also developed *product document analysis*, a novel method for analyzing the structure (and content) of work products to discover and characterize what the outcome of work should be. Our approach also draws ideas from Cognitive Work Analysis and WCD (Eggleston, 2003; Naikar, Hopcroft, & Moylan, 2005; Roth, 2008; Vicente, 1999) and from distributional analysis of language corpora (Brill & Marcus, 1992; Harris, 1951). Product analysis provides an analysis *method* as well as an analysis *result* and would be one way of discovering the type of *product ontology* proposed by WCD (Butler & Zhang, 2009). We shifted from an initial focus on understanding the current tasks and workflow of planning to understanding the resulting product of planning—namely, the structure of sound plans. Additional details are provided in Billman, Feary, Schreckenghost, and Sherry (2010).

Product analysis is relevant for a body of cognitive work where the primary output of work is semistructured documents such as reports, budgets, or plans. Product documents are often used by the producers of the documents, but typically these are also exported to others, and this helps define their importance and status as products. Product documents often describe or represent important entities, relations, and structures in the work domain, and agreements about what information needs to be communicated. Product analysis identifies the elements, relations, and structures that occur in the document corpora. A specification of the way output documents are structured and the patterns or constrains on the content provides a characterization of what must be accomplished for successful work. The requirements on output are likely to provide a more stable and accurate account of work needs, than would a focus on process; new technology should certainly change and improve process, yet the goal and outcome of successful work stays much the same. Product document analysis focuses on what should be stable across technology change and can provide a fruitful support to designing such technology change.

Two very helpful domain experts supported the needs analysis. They advocated for the project, introduced us to the domain, demonstrated current software and procedures, shared their thoughts on problems and solutions, provided access, and helped build and respond to the evolving representations of the work needs. However, only limited opportunity was available for face-to-face interaction or for observation. We had three opportunities to observe on-console execution of plans by several operators. This provided understanding of the context and constraints in the larger setting and provided insight into how plans are used in execution, but on-console was not the primary context for building plans. Observation of planning proved difficult to arrange. Rather than observation of normal workflow, we watched as experts demonstrated how they did a very small sample of tasks. Further, it was difficult to understand from observation what was being done or why, and there was limited time either for simultaneous explanations or for playback and explanation afterward. Driven in part by limited access to experts and observation, we accessed and analyzed a substantial range of documents. We acquired both (1) documentation, such as training materials, operating procedures, “cheat sheets,” and diagrams, and also (2) work products—that is, the plans built by the ADCO.

Task-Focused Phase of Needs Analysis: Method and Results

The task-focused analysis began with a mid-level description (Human Computer Interaction Process Analysis [HCIPA] from Sherry, Medina, Feary, & Otiker, 2008). This preliminary interaction identified the tasks related to generating a UAF, their frequency, and their associated hazards. The analysis also documented the decision-making and analysis steps that were involved in generating a UAF. The subject matter experts, who were steeped in the current tools, used the terminology of the existing tool and its features to describe the process. This elicitation process was both time-consuming for experts to carry out and was anchored too much to the current system to provide an ideal guide for redesign. It served as a way to get domain experts to reflect on “what” they do and “how” they do it. The experts found this process challenging, because the UAF generation is a complex process that is steeped in complex organizational structure (i.e., U.S/Russian negotiations, coordination with other Mission Controllers), and the structure and limitation of the existing editor-style tools.

We then shifted to a higher level functional description (e.g., as advocated by Kieras, 1996) and developed the Mission Decomposition Matrix (MDM) to represent and analyze work. The MDM evolved from the Task Design Document (Sherry & Feary, 2004), as had HCIPA. It decomposes a larger mission into a sequence of high-level tasks (or functions) and, for each, specifies these aspects: (a) a name or description; (b) the information produced by the task (output); (c) the information, and its source, needed to do the task (input); (d) the trigger for and frequency of the task; and (e) significant subtasks, which can be represented in their own matrix. We built a complementary, graphical representation that showed relations among the tasks in the MDM. This diagram type, which can be represented in UML (Unified Modeling Language; Pinheiro da Silva & Paton, 2000), emphasized decision points, contingencies, and information flow. It provided information that might be conveyed in a Sequence Model and a Flow Model within the Contextual Inquiry approach (Beyer & Holtzblatt, 1997). Use of dual, differently framed (table vs. graph) representations was helpful; for example, as the expert reviewed the graph, he noticed missing elements not noticed from review of the matrix alone.

The results of the high-level task analysis gave us an understanding of the nature of the work, identified key pain points, and suggested a broad solution approach. A central part of ADCO planning work concerns verifying, scheduling, and coordinating aspects of the developing plan, among multiple ISS stakeholders. Although technical expertise is required to ensure that formulated plans are technically correct and minimally hazardous, a large amount of work concerns scheduling attitude changes to meet stakeholder requirements, and communication among stakeholders to gain schedule concurrence. For example, scheduling the events for a docking depends on the availability of crew, of the docking port, and of bandwidth on the correct communication channel, as well as confirmation that the configuration will not disrupt requirements of the groups managing thermal and power-generation needs; information from these stakeholders arrives and changes at various times. When asked what made a "bad day," the ADCO expert gave an example that concerned managing last-minute scheduling changes.

This task-focused analysis identified the following key, high-level need: improved support for schedule revision. Given this, we hypothesized that using scheduling software rather than trying to modify the legacy form-editing software might be much more productive. Though the analysis clearly showed coordination was important, our project resource limitations meant that we could not support actual communication tasks; we needed to address work of an individual, while being sensitive to the (asynchronous) communicative context in which individual work is embedded. We showed ADCO experts an existing planning and scheduling tool, developed by an HCI group at NASA Ames for other mission control groups (McCurdy, 2009; McCurdy, Ludowise, Marquez, & Li, 2009). The ADCO experts responded enthusiastically to this planning tool, developed for a different set of Mission Control functions, and we used this existing platform as our basis to enable rapid development of a new prototype for ADCO.

Although advancing our understanding, the high-level task analysis had shortcomings. On the one hand, it was hard to distinguish functions that were critical to accomplishing the work from unnecessary activity that might go away given better tools. On the other hand, we still lacked information that would be desirable to guide development of a planner, in particular what exactly the product of work was supposed to be. Thus, details we had gathered might not be relevant, but we lacked detail that clearly would be needed for design. Limitations of low-level task analyses and also task-focused analyses in general have been noted elsewhere, and we experienced some of these limitations (Kieras, 1996; Vicente, 1999). Given that the goal of this part of ADCO work is producing an agreed-upon plan of operations, what is needed for such a plan?

Product-Focused Needs Analysis: Motivation

Product analysis is relevant when the goal of work is generating some product, in our case a plan. We generated this product analysis method as we did the research, but discovered that WCD has a similar interest in information products (Butler et al., 2007). Product analysis characterizes what constitutes a satisfactory product and may be particularly valuable if the process is subject to change or variation, as through the introduction of new technology. Information work often produces documents as products. The product-documents from financial work might be several forms of monthly budget sheets for different customers, from

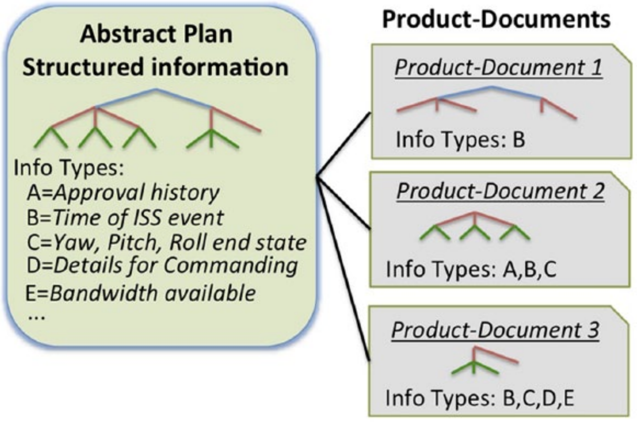


Figure 3. Schematic relation between an abstract plan and specific planning products. The content of the abstract plan can be inferred from product-documents that represent (parts of) the abstract plan.

architectural work might be various drawings for different subcontractors, and from planning work might be documents specifying the plan for different user groups. However, the information product may be more abstract than particular documents. Figure 3 schematically illustrates the relation between an abstract plan and product documents. When informational rather than physical entities are the work product, the core product may be an abstract information object that may be expressed in various documents. The structure of the abstract information product can be inferred from those documents, and the abstract product may be a more useful guide for redesign. In our case, the product is an abstract plan that is expressed in several product documents. This abstract plan includes timing, engineering parameters, and traces of authoring and approval history; and it provides the basis for coordinated action and advanced decision-making.

Although the focus of product analysis is on product not process, the product often encodes critical aspects of the process. In our case, ADCO plans included approval traces (assurance that the process followed requirements for safety) and revision history.

In domains where work products are expressed as documents, product analysis provides practical and conceptual benefits. On the practical side, once we gathered a collection of product documents, access was unrestricted, in contrast to our very restricted access to experts' time. A product-document analysis can consolidate and explicitly represent information that was implicit and distributed before. Domain experts should be able to recognize and vet the proposed structure and content identified in a document-analysis, even though they may not have time or resources to generate an analysis by reflection. On the conceptual side, product-analysis for information work can recover the structure and high-level content of the product-documents, based on the elements, properties of elements, relations among elements, and resulting organization. It can recover restrictions on what elements or properties can co-occur or what values are required in what conditions.

Document analysis can also provide information about the operations used to produce documents, for example, by analysis of revision histories. An analysis of the product provides valuable guidance for a relatively high level of design and evaluation. Of course, a good high-level design can certainly be defeated by poor choice of details, and design may also need to respond to process constraints not represented in the product. The information that a product-document analysis can provide may be necessary but is not sufficient for good design outcomes.

Analysis of product-documents can certainly be informed by documentation that explains the contents of the product-documents or how they should be written. ADCO documentation was very relevant for helping understand ADCO work generally, but little documentation was focused on planning.

Product-Focused Phase of Needs Analysis: Product-Document Analysis

Product-document analysis consists of five activities. They are temporally ordered but allow iteration; for example, our experts provided feedback at multiple points.

(1) Identify relevant product-document types. A product-document is the output of information work. Documents such as operations and training manuals for work are not its products, though they may provide input to work. Defining the set of product-documents defines the scope of analysis. The files exchanged with Russian Controllers, the UAFs (Unified ACR (Attitude Change Request) Files), were the core of our analysis as they were most closely linked to the "pain points" we aimed to relieve. Discussion with experts identified two additional plan document types with distinct but overlapping content, initially considered part of separate work activities. These were for different users: the *Attitude TimeLine (ATL)* for other NASA controller groups carried less detail overall and the *Ops Timeline* for ADCO internal use in execution with overall more technical detail.

(2) Gather the product-document samples. We gathered all the UAFs for two multimonth ADCO planning epochs, one consisting of 416 UAFs, the second of 209; the second was the focus of our analysis. Each collection had all the files sent between Houston and Moscow controllers, and individual files contain information about revision history. We had the ATLs for both periods and the Ops Timeline for the second. We also gathered manuals and "cheat sheets" that touched on planning or on the events being planned.

(3) Analyze detailed document structure and content. The domain experts provided a general tour of each document type. We manually analyzed the structure and content of UAFs and other file types, reviewing the 209 UAF files for one planning epoch and selectively reviewing other files. On the one hand, we developed an informal, semantic understanding of the units, operations, properties, and constraints by studying the product files, reading manuals, and talking with the experts. On the other, we developed a structured, systematic analysis based on syntactic relations such as co-occurrence and exclusion. We aimed to rely as much as possible on formal "syntactic" properties plus some semantics of temporal properties, rather than our open, semantic understanding, to push how far these "automatable" cues might go.
We identified events that formed the units of a plan. We identified constraints and requirements concerning combinations of units, combinations of properties allowed within one unit, and sequence requirements and we posed these for review. For example, if we found that a particular pattern (such as a combination of initial and final control mechanisms in the same event) never occurred in our sample, we asked if this was a necessary prohibition or an accident of our sample. Although our expert could answer such questions, sometimes the questions required reflection and the experts did not have accessible existing formulas for answers. We presented any violations of the patterns we had identified, for expert review; for example, in two cases, operators had sent an oddly structured plan document that did not, in fact, specify a plan but rather sent new parameter values (a creative use of the plan-exchange software that violated the normal structure in a UAF). Our focus was on plan structure, but we also identified operations on elements and relations needed to build and revise the plan structure.
The common content we found across the file types argues for a shared, underlying plan expressed differently in the different types of product documents. Revision histories provide further evidence for an abstract underlying plan: revisions in a document of one type (e.g., the UAF) propagated as revisions to the corresponding parts of the other document types.

(4) Abstract the high-level structure. From our informal understanding and our detailed analysis of particular events, we looked for sets of events that behaved similarly in terms of their relations to other events and identified event categories based on this. We used co-occurrence and exclusion patterns (an informal analog of more formal distributional analysis in linguis-

tics), but we also were guided by the meaning or content of the event. We abstracted the high-level structure of ADCO plans and the ISS events they represented. We identified classes of elements, relations, and structure. We mapped these onto terms in use by ADCOs or in ADCO documentation; sometimes there were multiple or indirect terms. There are several senses in which aspects of structure identified in our analysis were implicit. (a) Not all of the event classes identified had an explicit, consistent term used to refer to them; (b) elements of software and elements of a plan were conflated—sometimes terms for a software element (e.g., Unified ACR File) were used to refer to a type of plan element (Activity); (c) planning and plan content was learned informally by working with more experienced ADCO operators, although structured training was provided for using the software tools to build the document types; and (d) no one plan product represented the full plan, nor did any unified database provide a common information source.

(5) Vet proposed analysis by domain experts. We reviewed and discussed the components of our analysis incrementally with the ADCO expert who initially contacted us, incorporating his feedback as we progressed. After describing the abstract structure of the plans and the planning domain they represent, we conducted a structured interview with this individual and two additional ADCO operators to vet the content of the top-level analysis. They corrected, filled in details, or vetted the components of our analysis. For example, one expert added a low level of procedure "steps," which is important in execution but not visible in advanced planning (similar to level shifts in Vallacher & Wegner, 1987).

Product-Focused Needs Analysis: Product-Document Analysis Results

We only report our findings concerning the abstract structure, because these results guided our high-level design and our evaluation of the resulting prototype. An ADCO plan is a structured sequence of events, which are controlled and monitored in execution. Events have types characterized in terms of their relations to other events and their attribute values. Overall, we identified four key aspects of plan structure: (1) temporal relations, (2) part-whole relations, (3) event levels and types, and (4) event attributes. The structure of event types and their temporal and part-whole relations are illustrated in Figure 4 and summarized in Table 1.

(1) Plans are structured by *temporal relations* among events and temporal properties of an individual event. Each event has a duration (interval or point). Events at the same level are nonoverlapping. Events may be separated by intervals or transition without a gap. Events are ordered with respect to other events, particularly the next and previous. Events are specified in terms of absolute as well as relative time.
(2) Events have part-whole, as well as temporal relations. High-level events (longer duration) are composed of lower level, shorter duration events, bounded within the higher level event. Events at a given level may be of different types and have different attribute values. In this domain, events at the same level do not overlap.
(3) There are three levels of events important in planning within this part-whole structure: *Increment, Activity*, and *Action*. The fourth, lowest level (*Step*) is important in execution only and therefore fell outside our scope of analysis.

    (a) *Increment*. The unit of Increment is explicitly named and defined, as the period between ISS crew changes, typically weeks or months. Each increment is individually named (e.g., "Increment 22"); we do not know of any distinct Increment types. ADCO planning is organized around this unit, with each increment having one Increment Lead, responsible for planning and execution of ADCO functions.
    (b) *Activity*. The midlevel unit, Activity, is a meaningful collection of actions organized to support a common goal. This level of structure is more inconsistently named, less documented, and more implicitly represented than the Increment and Action levels. We identified eight activity types, such as a docking or thruster test. Each type has a canonical sequence of actions. We identified these types and their characteristics from product-document analysis

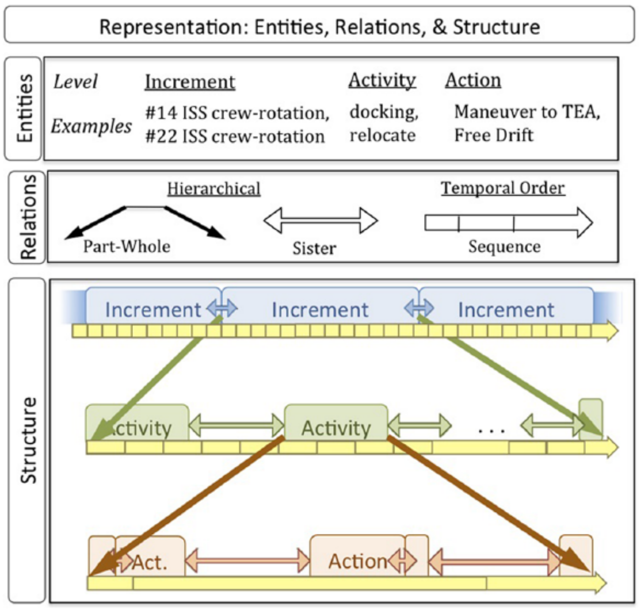


Figure 4. Structure of the ADCO planning domain. Key properties that guided our redesign are: temporal order, hierarchical relations, and three levels of entities including Activities. Left panel shows the representation entities, relations, and structure in the abstract ADCO plans; entities are events at three levels of analysis. Right panel shows the operations on plans used in plan revision. Note that construction of plans requires Add and Delete operations for each element. Content of plans is primarily specified by the types of Activity and types of Actions, with attribute values of the element modified as required.

TABLE 1: Summary of Plan Structure Identified by Product-Document Analysis

Type	Description	Examples
Relations		
Part-whole	Higher level, longer event types contain constituent events	Increments contain Activities. Activities contain Actions.
Temporal	Events at the same level are temporally ordered and do not overlap	
Event units		
Increment	Planning unit spanning period between a crew change.	"Increment 22," "Increment 14," etc.
Activity	Goal-directed unit planned and executed by ADCO and Russian counterparts.	Docking, Thruster Test
Action	A commanded change to ISS control status, orientation, or trajectory.	Handover US to Ru, Thruster Disable
Attributes	All events have temporal and meta-data attributes; actions also have engineering parameters.	Start Time, Control Type at End

and asking experts to confirm or clarify them. Activity types and characteristics are not documented, as Action types are. The value of our analysis results—specifically, identifying and codifying the Activity level—was suggested by an ADCO expert reviewing the training material we developed for use in the experiment reported in our companion paper. The ADCO expert commented that it was great to introduce the concept of activity directly and not even call it a UAF (which is a file type not a plan element), suggesting this analysis was novel and an improvement.

(c) *Actions*. These event units are specific operations controlled by ADCO or its Russian counterparts. Although terms that refer to Actions in general vary, the individual action types are explicitly named and key required attribute values for different types are documented. We analyzed the content of each Action type further, particularly what attributes could be changed by each Action type and their sequential dependencies.

(4) Events have *attributes*. All events have some temporal attributes, such as start and stop times, and some meta-data attributes, such as author or approval status. In addition, all Actions have engineering parameters, represented as values of attributes. There are about 14 key planning parameters, such as initial and final controller setting, frame of reference, and yaw, pitch, and roll; a larger set is relevant in execution. Planning involves very specific relations between actions (e.g., time needed to maneuver from the particular attitude of a prior action to the attitude of the following action).

In addition to information about structure, product-document analysis also provided information about *process*, specifically what plan-revision operations occurred. Examination of all tracked plan revisions in the UAFs for the increment studied showed that rescheduling times of events was by far the most frequent, and resetting the attitude values (yaw, pitch, or roll) was the other frequent revision.

SOFTWARE DESIGN GUIDED BY NEEDS ANALYSIS

The prototype redesign was based on results from our needs analysis. Needs analysis focused our redesign on replanning, particularly the routine work of entering and checking revisions. Identifying an existing planning software platform that could be modified to current needs, SPIFe (Scheduling and Planning System for Exploration; McCurdy, 2009; McCurdy et al., 2009), made it feasible to design and build an appropriate prototype. Primary changes within the SPIFe platform architecture were developing ways to represent the part-whole hierarchy.

Our prototype was designed to align with the temporal and part-whole aspects of work structure identified in the product-document analysis. Figure 5 illustrates the relation of the domain structure to the structure of software interaction and shows the differences in alignment between the *Legacy* versus *New* systems. The New but not Legacy system provides (1) explicit representation of Activity-level entities, presenting them in the context of their Increment; (2) explicit representation of temporal and part-whole relations, including a graphical timeline display of the relations; and (3) richer operations on the higher level entities of Activities and Increments, enabled by their explicit representation. In addition, the software included templates for Actions and Activities, providing required content in (editable) default values.

Difference in representation of Activities is a critical point of contrast between systems, and one investigated in our experimental evaluation in the companion paper (Billman et al., this issue). The New software provides operations on Activities, such as rescheduling an Activity as a unit. In contrast, Activities are not explicitly represented within the Legacy system; rather, they are only represented as files (i.e., at the level of the operating system), so the only operations at the Activity level are selecting and viewing; no operations internal to the software, such as editing, can be applied. Only one Activity can be opened at once within the Legacy application, so even simple operations such as visual comparison are not well supported, nor are any relations included among Activities or between

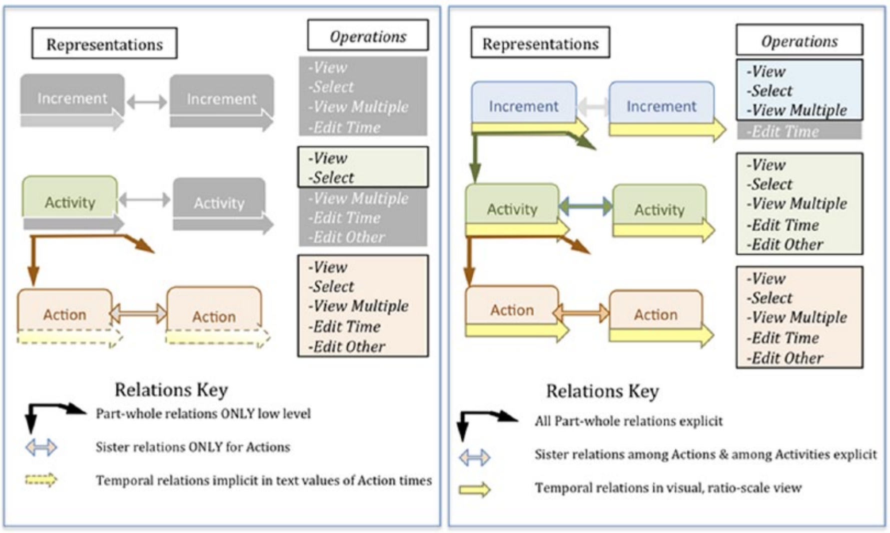

Representations
Operations
Increment
Activity
Action
-View
-Select
-View Multiple
-Edit Time
-Edit Other
Relations Key
Part-whole relations ONLY low level
Sister relations ONLY for Actions
Temporal relations implicit in text values of Action times
All Part-whole relations explicit
Sister relations among Actions & among Activities explicit
Temporal relations in visual, ratio-scale view

Figure 5. 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).

Activities and their Increment. In contrast, the New system shows Activities in relation to each other, their Increment, and their parts, and it allows operations on Activities, such as rescheduling an Activity as a unit.

The systems can be further compared by looking at their interfaces, as shown in Figures 1 and 6. Figure 1 shows the primary, four-panel window of the Legacy system, the form editor for UAFs. The upper left panel displays and edits metadata about the UAF. The lower left panel is a view-only, scrollable window, displaying the descriptions of about two actions (ACRs); the action being edited is highlighted. The upper right panel displays the action being edited, allowing the user to change the action's status and to move through the file by selecting the next or previous action. The bottom right panel is the primary edit space, allowing the user to set values for each of the action's attributes and add actions to the UAF. All editing is done at the level of an individual attribute value for an action.

Figure 6 shows the window of the New software. An increment is represented as a plan, activities are expandable hierarchical events in a plan, and actions are component events within an activity. (1) The top tool bar includes functions such as zoom and undo. (2) The left panel displays the available plans, selectable by clicking. (3) The central panel provides a scrollable timeline view of an increment. The top two colored timelines are relevant to the tasks reported here. The top, *Activities* timeline, summarizes events at the activity level. The second, *Plan Hierarchy* timeline, provides two ways of accessing actions from activities: (a) The list on the left expands the row to list the actions, and (b) the activity name within the timeline can be clicked to expand to show the component actions. The duration of activities and actions is indicated by their display size in the timeline. Actions and activities can be dragged and dropped to new times. (Implementation issues limited the ease with which events could be precisely dropped.) (4) The right, Details-Edit panel allows display and edit of the attribute values of a selected event. For a selected activity, its component actions, time information, and metadata are displayed. For a selected action, the engineering parameters (attitude, control, mass properties index, etc.), time, and metadata are displayed. In addition, collections of actions can also be selected and edited: If a constant value, say an updated Mass Properties Index, is needed for all selected

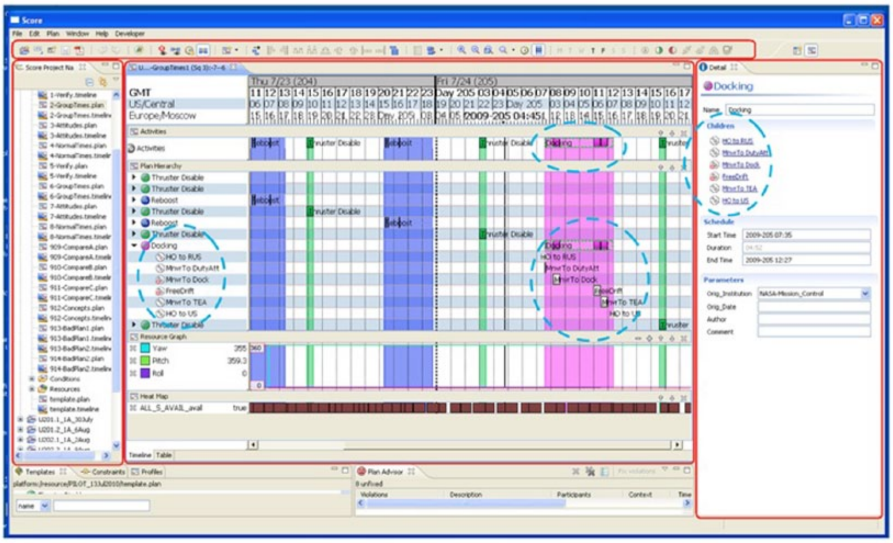


Figure 6. Screenshot of NEW system, showing the four main function panels for editing plans outlined in red. The example shows one activity expanded and selected. Dotted circles show four of the five possible repesentations an Activity available in this design.

actions, this can also be set through the Details Editor. Implications of differences in design for performance are explored via simulation (Lee & Billman, 2011) and via a laboratory experiment reported in the companion paper. Both evaluations found advantages of the New prototype over the Legacy planning software on plan-revision tasks. The empirical evaluation also tested for and found greater performance advantage where improvement to alignment was greater, for otherwise similar items.

DISCUSSION

Benefits of Needs Analysis

Our research made both ADCO-specific and more general contributions. Specific to ADCO, the needs analysis of part of ADCO work identified the core aspects of work. The product-document analysis characterized the structure of the work domain, enabling us to select and develop better aligned prototype software for ADCO work. This report details product analysis because it is the more novel aspect of our analysis. However, task-focused analysis (as well as informal observation and interaction) was also valuable, for example, in identifying pain points in current practice.

At a more general level, our research illustrates how needs analysis, and product-document analysis in particular, can identify needs of information work in terms of the requirements on the information products and thus contribute to design of better aligned software. Although the analysis uses a concrete collection of documents as input, the resulting output characterizes the abstract information structure. This can be used to guide design of the underlying information architecture of a system as well as the user interaction. Our work is part of the recent interest in identifying work characteristics such that they can play a role in design and evaluation as a system moves from conceptualization to operational status. For example, function allocation based on specifics of the work may be particularly useful compared to reliance on broad concepts such as level of automation (Defense Science Board Washington DC, 2012; Feigh & Pritchett, 2013).

Identifying the requirements on product documents for information work means identifying key constraints that must be met for work to be successful. These provide critical, high-level drivers guiding design of the software that supports production of those product documents. Although additional details about the work and additional decisions about the design will be needed, the structure of the product provides a critical framework for high-level design. If time and resources are limited, we suggest that there will be good payoff from prioritizing identification of product requirements. If correctly identified, these almost certainly will be more stable across changes in supporting technology or work context than will characteristics of the procedures for building the product.

Limitations of Analysis

Product-focused analysis is most relevant where the primary goal of work is generating products: for cognitive work domains, the products typically are (semi-)structured documents. These work domains are likely to be more reflective and less reactive, with the process more user-governed and less tightly coupled to the dynamics of a changing environment. Product-focused analysis will likely be less useful for domains that emphasize process rather than product, such as process control, system regulation, or piloting airplanes; nevertheless, such work may have subgoals of producing documents such as flight plans or work schedules that are needed to perform the more dynamic aspects of work. At an abstract level, identifying the requirements on the product documents for information work is analogous to identifying the characteristics of a safe and efficiently running chemical plant in process control. In both cases, the objective is to identify the fixed requirements or constraints in the work that will endure independently of the processes for producing "the product." In both cases, identifying the units and relations among units is foundational; however, the methods of identification will differ.

Our needs analysis prioritized identifying the structure of the work domain, or the structural constraints. In some planning domains, optimizing known continuous-valued parametric constraints on resources (energy, money, or access to an individual, supply, or tool) may be the key problem, and other forms of analyzing products may be more useful. We did not focus on parametric constraints for two reasons. First, specification of parametric constraints depends on specification of the entities, relations, and structures in the domain, which are constrained and constraining. Second, meeting parametric resource-allocation constraints did not seem to be a key driver in the ADCO's planning work; rather, much work consisted of incremental revisions to get agreement on exactly when and exactly which actions would be executed. (We did identify some resource-linked constraints, such as the need for communication bandwidth for a particular maneuver.) Plans were designed to be conservative and well within large safety boundaries, keeping planned actions far from safety constraints. ADCOs were supported by engineering analysts responsible for assessing the current safe (and sufficiently efficient) states, so much of the analysis of parametric constraints occurred elsewhere. Constraints often seemed to be "soft" in that, when pressed whether something was always the case, experts would produce exceptions: For example, though an event may "require" particular communication satellite availability, in context some communication gaps are feasible. Our analysis did not aim to identify all the relevant parametric constraints, nor use these as key drivers for design. Indeed, the software platform we use had well-established means for adding resource-based constraints; our challenge was how to represent the right entities and relationships.

Applicability of Product-Document Analysis

Needs analysis is valuable across work domains. However, different characteristics of work domains may lead to different applicability of analysis methods. Analysis of product-documents was useful in our domain and is likely to be useful in other work domains as well, when they share key characteristics true of our domain. Key characteristics include:

(1) Work is done by experts. Analysts and designers may share only a small part of the domain expert's knowledge.
(2) Cognitive activity by those experts is hard to understand from observation.
(3) Access to experts is very limited, posing a key constraint in conducting needs analysis. The "frozen" expertise in documentation or training materials may be of limited value for many reasons: lack of coverage, a focus on in-theory rather than in-practice operation, gaps for key aspects of work that are filled by training rather than documentation, and out-of-date documentation.

(4) The boundaries of the domain for information work are not clear-cut. Unlike operation of physical devices, boundaries of information work may not be unique, may be less visible, and may be more easily reconfigured by introduction of altered technology. Determining the scope of work to be supported may emerge as part of the analysis, not in initial definition.

(5) The work product is primarily informational and is expressed in concrete entities such as files or documents, but conveys an underlying, abstract information structure.

(6) The information products are used to communicate and coordinate, typically with others outside the boundary of the immediate work domain to be supported. This may encourage explicit and standard structure of product documents and thus facilitate product-document analysis.

Our work domain had multiple types of product documents and we found analysis of product documents valuable. The lack of access to experts and lack of transparency of their work when observed limited even structured observational methods (Beyer & Holtzblatt, 1997). Neither boundaries nor constraints are of the same type or play the same role in information domains as in analysis of physical work domains (Vicente, 1999) though the broad concepts remain important. Product-focused analysis helped us to identify the boundary of the work to be supported and to identify the entities, relations, operations, and resulting structure of the work domain.

Future Research

The method of document analysis should be applied to other work problems to better specify how it can best be done and to develop more systematic ways of representing the results of analysis. It would be possible to develop computerized methods to support analysis. Document analysis tools could find candidate patterns of interest for human review and modification. Both "syntactic" patterns, such as headers or reoccurring sets of delimiter marks, and "semantic" patterns, such as looking for references to the same time or identifying where numbers are a type of unit (angles, dollars, etc.), could be informative. Technology exists to identify many informative components that could help the analyst recover complex schema expressed through the document corpora.

More broadly, we believe needs analysis is a critical foundation for formal requirements, as well as the informal specifications used in this small software development project. A sound needs analysis should guide the entire process from requirements formulation, design, development, evaluation, and deployment.

Good alignment is necessary to assure a system is fit-for-purpose, but certainly not sufficient. The best aligned system can be made useless if combined with bad design choices on other aspects: Illegible font can defeat a good high-level design. However, we suspect that if the structure of the technology is badly aligned with the work, even the best interface "skin" cannot make the technology useful and usable. Product-document analysis is a method for improving alignment, and we hope it will be further explored. Our case study in the ADCO planning domain illustrates its promise.

ACKNOWLEDGMENTS

Parts of this research project were reported at CHI (Billman et al., 2011; Billman et al., 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; Jessica Lee for assistance with analysis; and Emilie Roth, Gudela Grote, and Paul Lee for comments on reports of this work. Author order after the first is alphabetical.

REFERENCES

Beyer, H., & Holtzblatt, K. (1997). *Contextual design: Defining customer-centered systems* (1st ed.). San Francisco: Morgan Kaufmann.

Billman, D., Arsintescu, L., Feary, M., Lee, J., 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., Arsintescu, L., Feary, M., Lee, J. C., & Tiwary, R. (this issue). Needs analysis and technology alignment evaluation: Evaluation case study of alternative software for controller planning work. *Journal of Cognitive Engineering and Decision Making*.

Billman, D., Feary, M., Schreckenghost, D., & Sherry, L. (2010). Needs analysis: The case of flexible constraints and mutable boundaries. In *Proceedings of the 28th of the international conference extended abstracts on human factors in computing systems* (pp. 4597–4612). New York: ACM Press.

Brill, E., & Marcus, M. (1992). *Automatically acquiring phrase structure using distributional analysis*. Berkeley, CA: Association for Computational Linguistics. doi:10.3115/1075527.1075561

Butler, K. A., & Zhang, J. (2009). Design models for interactive problem-solving. In *Proceedings of the 27th international conference extended abstracts on human factors in computing systems - CHI EA '09*. Boston: ACM. doi:10.1145/1520340.1520659

Butler, K. A., Zhang, J., Esposito, C., Bahrami, A., Hebron, R., & Kieras, D. (2007). Work-centered design: A case study of a mixed-initiative scheduler. In *Proceedings of the SIGCHI conference on human factors in computing systems* (pp. 747-756). Boston: ACM.

Charette, R. N. (2005). Why software fails [software failure]. *IEEE Spectrum*, *42*(9), 42-49. doi:10.1109/MSPEC.2005.1502528

Defense Science Board Washington DC. (2012). *Defense Science Board Task Force report: The role of autonomy in DoD systems* (no. ADA566864). Retrieved from http://handle.dtic.mil/100.2/ADA566864

Diaper, D., & Stanton, N. (2004). *The handbook of task analysis for human-computer interaction*. Mahwah, NJ: Lawrence Erlbaum. Retrieved from http://www.loc.gov/catdir/toc/ecip044/2003012434.html and http://www.loc.gov/catdir/enhancements/fy0745/2003012434-d.html

Eggleston, R. G. (2003). Work-centered design: A cognitive engineering approach to system design. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, *47*(3), 263-267. doi:10.1177/154193120304700303

Ellis, S. R. (2000). Collision in space. *Ergonomics in Design: The Quarterly of Human Factors Applications*, *8*(1), 4-9.

Emam, K. E., & Koru, A. G. (2008). A replicated survey of IT software project failures. *IEEE Software*, *25*(5), 84-90. doi:10.1109/MS.2008.107

FAA Human Factors Team. (1996). *The interfaces between flightcrews and modern flight deck systems*. Retrieved from https://www.faa.gov/aircraft/air_cert/design…/csta/…/fltcrews_fltdeck.pdf

Feigh, K. M., & Pritchett, A. R. (2013). Requirements for effective function allocation: A critical review. *Journal of Cognitive Engineering and Decision Making*. doi:10.1177/1555343413490945

French Inquiry Commission. (1994). *Rapport preliminaire—A330 Toulouse*. Commission D'enquete sur L'accident Survenu le 30 Juin 1994 a Toulouse-Blagnac (31) a L'airbus A330 N^{o}42 D'airbus Industrie Immatricule FWWKH. Retrieved March 19, 2013, from http://www.rvs.uni-bielefeld.de/publications/Incidents/DOCS/ComAndRep/A330-Toulouse/Rapport.html

Harris, Z. S. (1951). *Structural linguistics* (1963 4th impression.). Chicago: University of Chicago.

Hofmann, H. F., & Lehner, F. (2001). Requirements engineering as a success factor in software projects. *IEEE Software*, *18*(4), 58-66. doi:10.1109/MS.2001.936219

Institute of Medicine: Committee on Quality of Health Care in America. (2000). *To err is human: Building a safer health system* (L. T. Kohn, J. M. Corrigan, & M. S. Donaldson, Eds.). Washington, DC: National Academy Press. Retrieved from http://books.nap.edu/openbook.php?isbn=0309068371

Kieras, D. (1996). Task analysis and the design of functionality. In A. Tucker (Ed.), *Handbook of computer science and engineering* (pp. 1401-1423). Boca Raton, FL: CRC Press.

Kirwan, B., & Ainsworth, L. K. (1992). *A guide to task analysis*. London: Taylor & Francis.

Lee, J. & Billman, D. (2011). Modeling performance differences across systems, tasks, and strategies. In L. Carlson, C. Hölscher, & T. Shipley (Eds.), *Proceedings of the 33rd Annual Conference of the Cognitive Science Society* (pp. 3489-3495). Austin, TX: Cognitive Science Society.

Leffingwell, D. (1997). Calculating the return on investment from more effective requirements management. *American Programmer*, *10*(4), 13-16.

Leveson, N. (1995). *SafeWare: System safety and computers*. Reading, MA: Addison-Wesley.

Maris, K., Dulac, N., & Leveson, N. (2004). *Beyond normal accidents and high reliability organizations: The need for an alternative approach to safety in complex systems* (ESD Symposium, Massachusetts Institute of Technology). Retrieved from sunnyday.mit.edu/papers/HRO-final.doc

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.

McLeod, L., & MacDonell, S. G. (2011). Factors that affect software systems development project outcomes. *ACM Computing Surveys*, *43*(4), 1-56. doi:10.1145/1978802.1978803

Naikar, N., Hopcroft, R., & Moylan, A. (2005). *Work domain analysis: Theoretical concepts and methodology*. Melbourne, Australia: Australian Defence Science and Technology Organisation.

Nasir, M. H. N., & Sahibuddin, S. (2011). Critical success factors for software projects: A comparative study. *Scientific Research and Essays*, *6*(10), 2174-2186.

Pew, R. W., & Mavor, A. S. (Eds.). (2007). *Human-system integration in the system development process*. Washington, DC: National Research Council, The National Academies Press.

Pinheiro da Silva, P., & Paton, N. (2000). UMLi: The Unified Modeling Language for Interactive Applications. In A. Evans & S. Kent (Eds.), *Proceedings «UML» 2000,LNCS, Vol. 1939*. New York: Springer-Verlag.

Procaccino, J. D., Verner, J. M., & Lorenzet, S. J. (2006). Defining and contributing to software development success. *Communications of the ACM*, *49*(8), 79-83. doi: 10.1145/1145287.1145291

Roth, E. M. (2008). Uncovering the requirements of cognitive work. *Human Factors: The Journal of the Human Factors and Ergonomics Society*, *50*(3), 475-480. doi: 10.1518/001872008X288556

Sarsfield, L. P., Stanley, W. L., Lebow, C. C., Ettedgui, E., & Henning, G. (2000). *Safety in the skies: Personnel and parties in NTSB aviation accident investigations: Master volume* (no. MR-1122/1-ICJ). Retrieved from http://www.rand.org/pubs/monograph_reports/MR1122z1.html

Savolainen, P., Ahonen, J. J., & Richardson, I. (2012). Software development project success and failure from the supplier's perspective: A systematic literature review. *International Journal of Project Management*, *30*(4), 458-469. doi: 10.1016/j.ijproman.2011.07.002

Schraagen, J. M., Chipman, S. F., & Shalin, V. L. (2000). *Cognitive task analysis*. Mahwah, NJ: Lawrence Erlbaum Associates. Retrieved from http://www.loc.gov/catdir/enhancements/fy0634/99042583-d.html

Sherry, L., & Feary, M. (2004). Task design and verification testing for certification of avionics equipment. In *Proceedings of* *the 23rd Digital Avionics Systems Conference* (Vol. 2, pp. 101-110). Piscataway, NJ: IEEE.

Sherry, L., Medina, M., Feary, M., & Otiker, J. (2008). Automated tool for task analysis of NextGen automation. *IEEE*, 1-9. doi: 10.1109/ICNSURV.2008.4559185

Vallacher, R. R., & Wegner, D. M. (1987). What do people think they're doing? Action identification and human behavior. *Psychological Review*, *94*(1), 3-15. doi: 10.1037/0033-295X.94.1.3

Vicente, K. J. (1999). *Cognitive work analysis: Toward safe, productive, and healthy computer-based work*. Mahwah, NJ: Lawrence Erlbaum Associates. Retrieved from http://www.loc.gov/catdir/enhancements/fy0740/98031670-d.html

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.

Michael Feary is a research scientist in the Human-Systems Integration division at NASA Ames Research Center. He received his Ph.D. in Human Factors Engineering from Cranfield University, UK (2006). Dr. Feary 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.

Debra Schreckenghost (MEE, Rice University) is a Senior Scientist with TRACLabs in Houston, TX, investigating adjustable autonomy and human interaction with automation and robotics. She also is the Associate Team Lead of the National Space Biomedical Research Institute (NSBRI) Human Factors and Performance Team. Before working at TRACLabs, she was a Lead Engineer at MITRE and a NASA Shuttle flight controller.

Lance Sherry is Associate Professor of System Engineering and Operations Research at George Mason University. Dr. Sherry also serves as Director of the Center for Air Transportation Systems Research (CATSR) at George Mason University. Dr. Sherry's research interests include: analysis and simulation of complex adaptive systems, robust decision making in the presence of complexity and uncertainty, and operator-automation system design, testing, and certification.