JUS
JOURNAL OF USABILITY STUDIES

Vol. 2, Issue 4, August 2007, pp. 190-210

Metaphor-Based Design of High-Throughput Screening Process Interfaces

David B. Kaber
North Carolina State University
400 Daniels Hall
Raleigh, NC 27695 USA
dbkaber@ncsu.edu

Noa Segall
Duke University Medical Center
Box 3094
Durham, NC 27710 USA
noa.segall@duke.edu

Rebecca S. Green
North Carolina State University
400 Daniels Hall
Raleigh, NC 27695 USA
seawalker04@yahoo.com

Abstract

This paper describes work on developing usable interfaces for creating and editing methods for high-throughput screening of chemical and biological compounds in the domain of life sciences automation. A modified approach to metaphor-based interface design was used as a framework for developing a screening method editor prototype analogous to the presentation of a recipe in a cookbook. The prototype was compared to an existing screening method editor application in terms of effectiveness, efficiency, and satisfaction of novice users and was found to be superior.

Keywords

Metaphor-based design, cognitive task analysis, usability evaluation, life sciences automation

Introduction

System designers have identified various ways to deal with the increasing complexity of user interfaces. One of these is the metaphor-based approach where properties of an interface are designed to take on the appearance and behavior of real-world devices or objects in an environment that is familiar to system users. Interface metaphors enable users to map knowledge from a familiar source domain to an unfamiliar target domain, allowing them to use prior experience to comprehend and navigate in novel situations (Neale & Carroll, 1997). Thus, metaphors are viewed as an important tool for facilitating learning and novice use of interfaces, since it is easier to construct a new concept from more established concepts than to develop understanding of an unknown abstract idea (Carroll & Mack, 1985). Other research has indicated that metaphor-based design may enhance interface usability (Kuhn & Blumenthal, 1996) and user satisfaction (e.g., Dutton, Foster, & Jack, 1999).

Target Domain

Contemporary high-throughput screening (HTS) processes involve chemical-based assays of organic and inorganic compounds for effects on human cellular functions, or enzyme reactions that are common in cells (Entzian, Allwardt, Holzmüller-Laue et al., 2005). Among the uses of HTS is the testing of compounds that may have the potential to serve as bases for drug derivatives for use in future medications to treat cancer, viruses, and more.

At the University of Rostock (Germany) Center for Life Sciences Automation (CELISCA), marine compounds undergo enzyme-based (for example, Trypsin) testing to identify compounds that might be useful for such drug derivative development. Testing the compounds involves several steps, including pipetting liquids (enzyme substrates, test compound extracts and other reagents) at different quantities and concentrations into micro (culture)- plates with many sample wells; incubating the micro-plates to elicit enzymatic reactions similar to those that would occur in the human body; and analyzing the reactions using optical measurements.

Complete automation of enzyme-based screening (such as a Trypsin test) includes the integration of an optimized robot for chemical analysis (ORCA) with analytical measurement devices on a process line (see Figure 1). The ORCA, which is programmed, controlled, and monitored from a central process control system (PCS), transports micro-plates to and from the various workstations on the screening line. The automated devices on the line include a bar coder for labeling and tracking micro-plates in which compounds are tested; an automated pipetting (liquid transfer) robot for filling micro-plates with liquids; an incubator for bringing the temperature of reactions in the micro-plates to human body temperature; and an automated micro-plate reader for analyzing the enzyme activity in each well of a plate using luminance tests of light reflected off samples. The reaction of various compounds with the enzyme causes biological products to be generated in wells that turn different colors. Target wavelengths (or reaction products) are identified by biochemists in advance of the screening process.

With the advent of highly automated analytical measurement devices that can transport and chemically evaluate thousands of biological samples on a daily basis, the role of human operators in HTS has dramatically shifted. Time-consuming manual material handling tasks such as pipetting biological compounds into micro-plates and mixing compounds with reagents, which were traditionally performed manually by operators, are now fully automated with robotics. The human’s task has become that of supervisory control – planning, managing, and analyzing the results of highly automated screening lines.

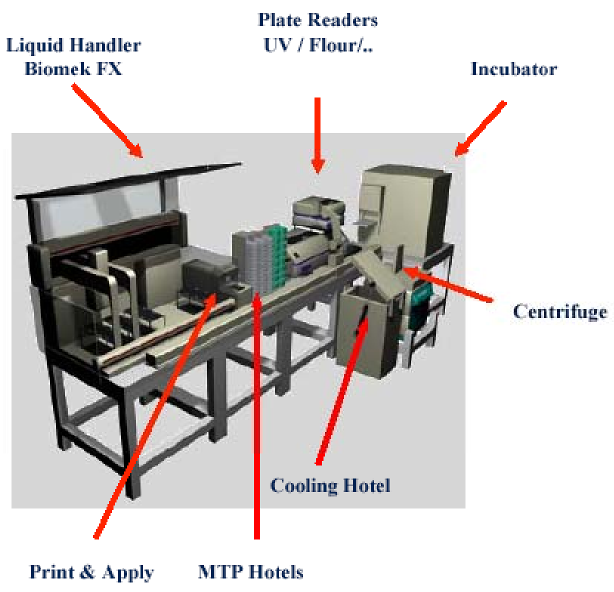


Figure 1. HTS screening line at CELISCA with labels of automated devices. (Note: The Biomek FX is a commercially available pipetting robot. Plate readers are used for ultraviolet (UV) and fluorescence and luminance testing of samples in plates. “Hotels” are temporary storage areas for micro-plates. The “print and apply” device is a barcoding machine.)

Specialized biological screening tests, such as the Trypsin screening test, are developed by scientists and published in the biotechnology literature (Thurow, Entzian, & Eberlein, 2004). “Bench-top” versions describe how to manually perform a particular screening assay.

Biopharmacologists at CELISCA adapt the manual bench-top protocol of the screening tests to automated HTS lines (Thurow et al., 2004). This process involves steps, such as identifying the appropriate micro-plate to be used in the automated assay, determining the mixture of liquids within each micro-plate, selecting and programming the automated devices that will be used to perform the assay, planning the pipetting processes (for example, which pipetting tips and reservoirs will be used), and establishing an optimal sequence of assay steps.

After developing the automated screening test protocol, operators use a computer-based screening “method editor” to program the specific methods that will be executed by devices and the sequence. For example, Beckman-Coulter currently publishes a software application called SAMI® (Sagian^{TM} Automated Method Development Interface) that allows a biochemist to develop graphical models (resembling flow charts) of screening tests to be conducted on a HTS line. Screening methods are constructed at the user interface by dragging icons representing available HTS devices (which will be used in the assay) to a central “scratch pad” and connecting them with arrows to define transportation of micro-plates among the device workstations by the ORCA (see Figure 2). Icons representing different types of plates are also available for further specifying a method. Operators then program pipetting robots and plate readers, which have their own proprietary control software. These devices are then integrated in the screening process with the method editor and the PCS, including an executive software controller.

The screening method editor includes standard Windows action/configuration dialogs for all devices on the HTS line, which operators use to set device parameters and to send data to the pipetting robot and plate reader software modules. For biopharmacologists, who may prefer to have samples (or the micro-plates that contain them), rather than HTS devices, at the “center” of the method-development process, this approach to programming the system may be less intuitive.

A typical HTS experiment poses a high cognitive workload for supervisors, who must keep track of the timing of process steps, whether robot motions are accurate, and whether chemical reactions are occurring safely. Errors can occur at many different phases of the screening process. For example, robot material handling errors can be caused by initial incorrect placement of labware by operators. Chemical reactions may not progress as planned because of operator errors in programming pipetting operations or wear of device components (for example, cables and pumps in the pipetting robot), leading to delays in reactions or the need to scrap an entire experiment. This can be very costly to the test facility because many of the organisms being investigated are extremely rare and extracts are expensive to develop.

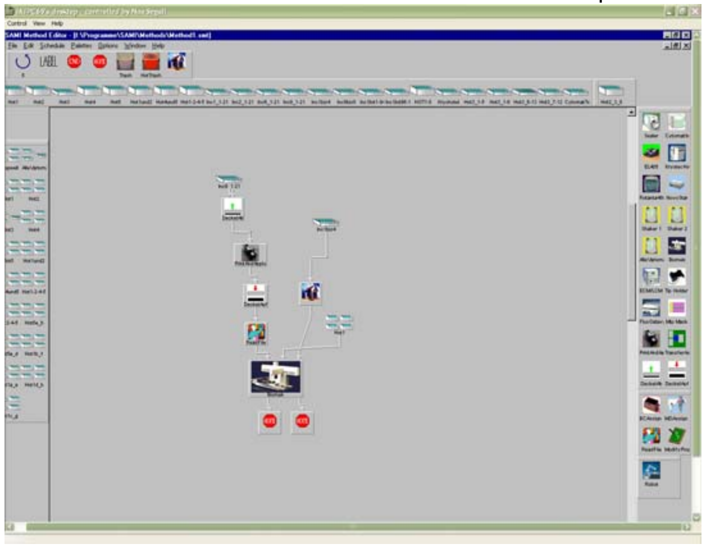


Figure 2. SAMI® method editor interface by Beckman-Coulter.

Consequently, operators may need to periodically intervene in the master robot control to prevent such errors and to keep a line from "crashing". The process control interface must provide the capability for both off-line and near real-time, closed-loop programming of process events based on the progress of an experiment.

Cognitive Task Analysis

To address the potential for costly errors, we previously evaluated the HTS process with the goal of identifying functional limitations of automated devices and usability problems with existing software interfaces for programming, executing, and analyzing screening experiments (Kaber, Segall, Green, et al., 2006). To this end, two cognitive task analysis (CTA) methods were employed, including abstraction hierarchy (AH) modeling and goal-directed task analysis (GDTA). An AH model is the representation of a work system along multiple levels of abstraction (Rasmussen, 1985), ranging from physical component properties to system purposes, which are linked in the model based on means-end relationships. AH models have been used for systems analysis and design and as a framework for describing control tasks necessary to maintain adequate system performance (Rasmussen, 1985). We developed AH models for all automated devices integrated in the HTS line at CELISCA, including the automated pipetting robot (Biomek2000®), incubator, ORCA (material transport robot), and automated micro-plate reader.

For example, in the AH model of the bar coder, at the highest level of abstraction, the purpose is specified as assigning IDs to plates and recognizing plate labels during processing. One of the generic functions to this goal is the "process of printing a barcode". This includes generalized functions of label feeding and applying thermal ink. These functions are supported by components, including the printer and label paper feeder.

A separate AH model was developed to represent the software or automation used to control the mechanisms of the bar coder. One of the general functions in this model is "printing control," including subfunctions of "selecting a barcode type," "defining the content of the code," and "storing the code." These functions are supported by different interface features, including dialogs presenting label-type options for selection and input windows for manual string entry or selection of files containing lists of defined plate IDs.

GDTA is a methodology that focuses on identifying operator critical decisions and situation awareness (SA) requirements relevant to performing complex systems control (Endsley, 1993). GDTA uses a diagram format similar to hierarchical task analysis, but concentrates on the organization of an operator's goal set and not on low-level operations with system interfaces. The outcomes of GDTA include perceptual requirements for task performance and operator needs in terms of system-state comprehension and projection. The results can be used as a basis for display design, training program development, and operator selection (Usher & Kaber, 2000).

We developed a GDTA describing biopharmacologist goals, tasks, decisions, and information requirements relevant to supervisory control of the HTS line. The complete GDTA included 25 high-level goals, 20 subgoals, and a total of 88 tasks to these subgoals. On average, there were 4.4 tasks to each subgoal and 2.2 critical decisions per task. An analysis of the SA requirements associated with the various operator decisions revealed a total of 228 unique pieces of task information.

For example, one of the subgoals in the GDTA deals with the application and reading of barcode labels during the HTS process. This goal involves two tasks: integrating the bar coder into the process (in the PCS software) and determining how it will be used during the assay. The operator must make two decisions when integrating the bar coder: what information will be included in the label and where the label will be applied to a micro-plate. The information required to address these decisions is the code that will be used on the label. To achieve the second task, that of determining the functions of the bar coder, the operator must decide whether a new barcode needs to be applied to micro-plates or whether an existing barcode is to be read. In addressing this decision, the operator needs to know whether a barcode is already present on the micro-plates (from the manufacturer or client) and what step is to follow bar coding in the assay process. To facilitate data collection (using the micro-plate reader), for example, it is first necessary to read the content of the barcode label. These operator tasks, decisions, and information requirements are captured in hierarchical outline form in the GDTA.

The combination of AH models, describing screening line devices and automation, with the results of the GDTA on biopharmacologist performance of HTS operations, allowed us to determine whether the task and functional requirements of operators were being met by existing system elements. The generalized functions of robotic devices, identified in the AH models, were related to the functions that operators required to achieve screening task goals, as captured in the GDTA. Furthermore, operator information requirements for task decisions, as identified through the GDTA, were compared with specification of current interface action sequences and display content based on the AH models to identify potential usability issues with the existing software interfaces (for example, SAMI®). That is, we were able to determine whether particular screening method editor displays and action sequences led to the information operators needed for certain process decisions. In our previous work, we made direct comparison of the GDTA results and AH models to formulate interface design and automation functionality recommendations for enhancing the existing software applications used in the HTS process at CELISCA (Kaber et al., 2006).

Design recommendations were formulated for all goals in the GDTA that pertained to the use of HTS software. The recommendations addressed usability goals based on Nielsen’s (1993) and Norman’s (1995) work, including providing a good match between the system and the real world, thereby promoting operator recognition rather than recall, enhancing the visibility of system status, preventing errors, and developing a parsimonious interface design. These recommendations ultimately were reflected in a new prototype of a control dialog as part of the existing screening method editor software. (We describe this prototype later in the paper, along with a metaphor-based revision of the primary method editor interface and usability evaluations.)

The objective of this research was to develop an interface design metaphor that could be used as a framework for implementing the specific design recommendations resulting from the previous CTAs to create usable and effective HTS supervisory control and method editing interfaces. Although the recommendations for automated device control dialogs and the existing HTS method editing software were useful, we needed an overarching framework in which to organize the recommendations for coherent design revisions. We conducted two usability evaluations to assess whether application of the metaphor and the specific design modifications led to actual improvements in usability and effectiveness for system operators. In general, the evaluations served to establish whether the new interface designs were suitable for the purpose of HTS device control and assay method editing, as well as the utility of the metaphor for this domain.

The Cookbook Metaphor

A cookbook typically contains multiple recipes, each with its own set of task components, such as the preparation procedure, the required ingredients along with their quantities, the equipment needed to prepare the dish, and estimates of how much time it will take to prepare the dish based on the number of servings. Similarly, the bench-top protocols used in biological screening processes, which describe how to manually perform a particular screening test (assay), identify required reagents along with their quantities, the equipment needed to perform the assay, and a detailed preparation procedure. When comparing a cookbook and a bench-top protocol, the chemical and biological reagents in the protocol become the recipe’s ingredients, the concentrations and amounts of the reagents are the ingredient amounts, the automated devices become the cooking equipment, and the protocol method becomes the “dish” preparation procedures. This strong similarity in the structure of the two concepts, cooking and bench-top assaying, led to our development and use of a cookbook metaphor for redesigning the interfaces of the method editing software previously created for the HTS domain.

Cookbook metaphors have been used in the domain of software development for organizing help utilities according to the overall layout of a cookbook. For example, Schappert, Sommerland, and Pree (1995) proposed the use of an “active cookbook” tool that provides semi-automated assistance to guide programmers in developing new software. “Recipes” in this cookbook describe (in an informal way) how to code certain generic application tasks. They contain a purpose, procedure, and source code examples. The “recipes” serve as a basis for generating source code based on programmer decisions and they help to structure the code by requiring that certain steps in the procedure be carried out before others. In Schappert et al.’s (1995) work, the cookbook metaphor is applied at the book level; that is, the active cookbook has components or features that represent those of a food cookbook. However, no prior work appears to have used the style of presentation of recipes in a cookbook as a metaphor specifically for interface design.

As a chef normally writes the cooking procedure for a recipe in natural language, we also wanted to prototype the capability for biochemists to develop automated assay methods using natural language processing (NLP). Previous use of the cookbook metaphor in software development has not integrated use of NLP for user input to automated assistance applications. The metaphor-based method editor interface design presented below incorporates a mock-NLP capability for assay programming.

In an actual application, natural language understanding systems must convert samples of human language into more formal representations that a computer program can manipulate. Some of the problems that make NLP difficult to implement, including text segmentation, word sense disambiguation, syntactic ambiguity, and speech acts, restrict NLP integration in intelligent interfaces (Dale, Moisl & Somers, 2000). However, since only a finite vocabulary is required to develop automated assay procedures from bench-top protocols, we expected this limitation to have little effect on an actual NLP-based HTS interface, and therefore, we prototyped this capability in our redesigned screening method editor interface. Other interface features were also incorporated in the method editor interface to capture the cookbook metaphor, such as lists of tools and reagents to be used in the HTS process being programmed. These are discussed in more detail below.

Usability Goals and Metaphor-Based Interface Design

Based on a review of human-computer interaction design literature, we adapted Neale and Carroll's (1997) metaphor-based design methodology to prototype a new method editing interface for HTS assay development. The usability of this interface was compared to that of Beckman-Coulter's SAMI® method editor currently in use at CELISCA. Three overall usability goals were identified for the new interface: (1) increased efficiency, that is, a shorter task completion time; (2) enhanced effectiveness, that is, a smaller number of errors; and (3) improved user satisfaction. It was hypothesized that the metaphor-based method editor prototype would be more efficient and effective and that users' reaction to it would be positive.

To develop a research foundation for implementing metaphor-based interface design, we reviewed work on approaches to using metaphors (e.g., Alty, Knott, Anderson et al., 2000; Carroll & Mack, 1985; Neale & Carroll, 1997). Neale and Carroll (1997) developed a five-step methodology, with the first stage involving identification of system functionality as a basis for mapping the system to potential source domains (metaphors). In the second stage, a designer is to generate possible metaphors, like the cookbook metaphor for the HTS protocol development. (We have already presented some information in line with these steps.) Neale and Carroll (1997) refer to several methods similar to iterative design processes, such as interviewing users, analyzing users' work context, and empirically analyzing their semantic and procedural knowledge, as related to the system.

The next stage is to determine how user tasks and methods at the conceptual interface relate to achievement of goals and plans. The tasks, methods, and appearance of the metaphor are analyzed for their association with the goals and plans of the user and the elements that they may work with in reality.

Stage four identifies discrepancies in the software domain related to the metaphor by determining when interface functions will lead users to perform erroneous actions rather than leading to new insights into the system domain.

The final stage provides strategies for assessing and controlling these metaphor-interface mismatches. The authors indicate that designers can use composite metaphors, teach modification rules for elements not represented by the metaphor, or encourage exploration of the system features represented by the interface. (Below we talk about complementing our cookbook metaphor with a cooking metaphor to address some discrepancies between assay method editing and developing a cooking recipe.)

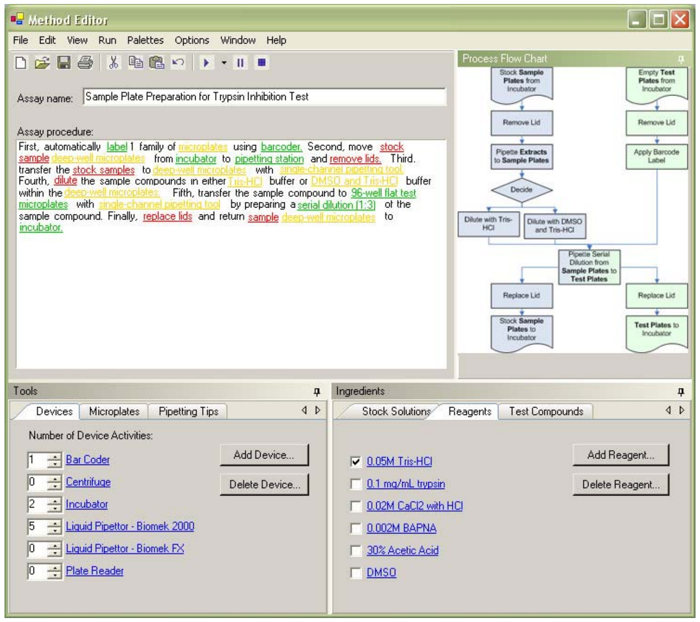
We adapted Neale and Carroll's (1997) approach by integrating the CTA techniques described above, as a basis for designing enhanced displays and controls as part of the HTS method editor interface. The first stage of Neale and Carroll's methodology involves identifying the system functionality and the features a system must have to meet user needs. To this end, we used our AH models for the HTS devices and automation to specify current system functionality. The device models identified all components supporting specific functions. The software models identified all existing interface features for controlling HTS automation. The GDTA content was used to identify users' needs with respect to the system components (for example, what type of information was required from device displays to support task performance).

In stage two of the methodology, the process of generating possible metaphors was also based on the empirical results of the GDTA. Like many CTA methods, the GDTA involves observing operators at work and conducting structured interviews on task goals, decisions and information requirements, as Neale and Carroll (1997) suggested.

The general procedure to HTS method programming and control revealed through the GDTA supported our identification of the cookbook as a possible metaphor for structuring the information content of a "bench-top" protocol for a biological assay. The major subgoals of the GDTA related to the protocol development included: "identify steps…," "identify appropriate plates to use…," "establish plate configuration…," "identify devices to perform steps," "identify time critical steps…," and so on. These goals are all akin to the goals of a chef in cooking and led to the inspiration to use the cookbook metaphor. As we stated, when comparing a cookbook recipe and a bench-top protocol, for example, an enzyme-based test, both contain a required ingredient list, a list of the equipment needed to prepare the dish/assay, and ordered preparation procedures for the dish/assay.

With these three components in mind, our new metaphor-based design of the screening method editor is comprised of three main windows: (1) a window for entering the assay procedure (deciphered by the system using pseudo-NLP); (2) a window containing the required ingredients for the assay; and (3) a window containing the equipment that will be used throughout the assay (see Figure 3). A fourth window was included to provide the user with a visual representation (in the form of a flow chart) of the assay procedures applied to micro-plates according to the "recipe" input by the biochemist in natural language. Flow charts of recipe preparation procedures can be found in many basic cookbooks. Thus, the flow chart window did not represent a violation of our metaphor (in format).

In the third stage of Neale and Carroll's (1997) methodology, we assessed whether the goals and plans of a biochemist in HTS protocol development were supported by the metaphor by comparing examples of activities in developing a cookbook recipe with HTS process development (see Table 1). We identified few "breakdowns" of the metaphor. For example, multiple instances of a HTS device in a tools palette interface could be considered unbounded or unpredetermined in the method-editing environment; however, multiple uses of a device in the physical environment could be considered bounded and predetermined. Similarly, the specification of physical re-use of a particular utensil could be considered unbounded and unpredetermined in the development of new recipes for a cookbook but might be expected in the actual cooking process.



Assay name: Sample Plate Preparation for Trypsin Inhibition Test

Assay procedure:

First, automatically label 1 family of microplates using barcoder. Second, move stock sample deep-well microplates from incubator to pipetting station and remove lids. Third. transfer the stock samples to deep-well microplates with single-channel pipetting tool. Fourth, dilute the sample compounds in either Tris-HCl buffer or DMSO and Tris-HCl buffer within the deep-well microplates. Fifth, transfer the sample compound to 96-well flat test microplates with single-channel pipetting tool by preparing a serial dilution (1:3) of the sample compound. Finally, replace lids and return sample deep-well microplates to incubator.

Number of Device Activities:

1	Bar Coder
0	Centrifuge
2	Incubator
5	Liquid Pipettor - Biomek 2000
0	Liquid Pipettor - Biomek FX
0	Plate Reader

- ✓ 0.05M Tris-HCl
- ☐ 0.1 mg/mL trypsin
- ☐ 0.02M CaCl2 with HCl
- ☐ 0.002M BAPNA
- ☐ 30% Acetic Acid
- ☐ DMSO

Figure 3. Metaphor-based method editor prototype.

Therefore, the key differences between the physical cookbook and the application of the cookbook metaphor to the method-editing interface were addressed by extending the metaphor to the basic process of cooking to encompass our design notions.

Table 1. Comparison of cookbook recipe components and HTS process elements.

Cookbook Metaphor	Prototype Feature
Writing recipe	Using NLP to author assay protocol
Specifying ingredients	Identifying chemical and biological reagents in "Ingredients Window"
Identifying necessary cooking utensils	Identifying automated devices and tools for use in "Tools Window"
Ingredient amounts	Concentrations and amounts of reagents
Reuse of utensils	Multiple instances of automated devices in a "Tools Window" palette
Selection of mixer attachments for use in dish preparation	Selection of barcode label to apply to micro-plate; Selection of pipetting tool for HTS process
Selection of baking dish	Selection of micro-plate

Related to the previous step, the fourth stage of Neale and Carroll's approach involves identifying discrepancies in the software domain related to the metaphor. They recommend covering as many aspects of the interface design (suggested by the metaphor) as possible, but remind designers of the importance of focusing on communicating system functionality rather than on mimicking every aspect of the source domain. We found that the cookbook metaphor more accurately reflected the functionality in some parts of the HTS system than others. One example of a metaphor match is the use of the micro-plate incubator in a HTS protocol, as the device carries-out a process similar to using an oven for cooking food. A metaphor mismatch example is the barcode print and apply device, which is used in the HTS environment for applying and reading identification labels on micro-plates, as it has no direct analogy with a recipe in a cookbook or typical processes in the cooking domain. That is, in cooking, one does not typically use a device to label materials with barcodes prior to preparing a dish: materials in cooking processes are typically uniquely recognizable by a chef, as compared to an ORCA handling hundreds of identical micro-plates.

This mismatch between the cookbook metaphor and the HTS environment was addressed by extending the concept of cooking utensils to appliances. In this case, the barcode device was implemented as a configurable cooking appliance. Therefore, the cookbook metaphor, represented by the interface, more closely resembles both the cookbook recipe and the cooking process.

Another mismatch example is the use of different colored hyperlinks in the assay protocol to indicate terminology or system references for which the editor requires further detail from a user. A "traffic light" metaphor was used for this feature (see Figure 3). These interface properties are also not analogous to the cookbook metaphor.

Both the third and fourth stages of the Neale and Carroll (1997) method can be facilitated through scenario-based analysis of interface prototypes. Consequently, we developed a HTS scenario that required interaction with prototype features including those for the identification of plates/sources, liquid transfer, heating, and returning micro-plates to home positions. To further ensure that users could perform the tasks and methods with the interface by thinking about the metaphor and relating features to their goals, evaluations of the prototypes were conducted with subject matter experts based on the defined scenario (see below). In this manner, we also verified that the prototype supported user tasks as revealed through the GDTA.

The final stage of Neale and Carroll's (1997) approach is to provide strategies for assessing and controlling metaphor-interface mismatches. We proposed to develop and teach biochemists rules for elements not directly represented by the metaphor, as in the case of the bar-coder device. Another method for managing mismatches is the use of composite metaphors, which can create a match with one metaphor where a mismatch occurs in another (Neale & Carroll, 1997).

Our use of the cooking process and traffic light metaphors were examples of this. Finally, mismatches may be handled by designing the interface such that it will encourage exploration of system features without penalty (Carroll & Mack, 1985). The colored hyperlinks and flow chart support such investigation of the new method editor prototype by allowing users to interact with interface elements and to learn about their function without affecting previous work.

This adaptation of Neale and Carroll's (1997) methodology resulted in an interface design supporting a process- (or micro-plate) oriented approach to HTS method programming. The metaphor-based prototype was expected to be more intuitive and easier for biochemists to use than the existing method editing application currently installed at CELISCA, which supports a device-oriented programming approach.

The new process-oriented method editor interface requires a user to initially describe an assay procedure, rather than identify the devices used to conduct screening. The metaphor-based interface includes a menu bar and toolbar, an assay name field and four windows, specifically the Assay Procedure Window, Process Flow Chart Window, Tools Window, and Ingredients Window (Figure 3).

To program a new assay, the user types a sequence of instructions in natural language (for example, "...Remove 1 family of 96-well flat test micro-plates from incubator and label using the bar coder...") in the Assay Procedure window. As text is entered, a flow chart is automatically created in the Process Flow Chart window (based on the NLP technology), documenting the different steps that a micro-plate is to pass through. The numbers of devices and amounts of materials to be used in the process that are recognizable by the system are also updated in the Tools and Ingredients windows of the interface, based on the assay description entered by the user. Finally, certain keywords in the description (recognizable by the system) that represent objects, such as devices, equipment, or materials, become hyperlinks.

Red hyperlinks represent objects for which the system needs further configuration or specification information to operate (for example, what type of label should the bar coder apply?). Yellow hyperlinks represent partially defined objects for which the system needs some additional information, but even if this information is not provided, the method will run using its default values. For example, if the user requests deep-well micro-plates, the default micro-plate will be used, but the number of wells in the micro-plate can also be specified. Green hyperlinks represent objects for which the system has all the necessary information to function – they are fully defined. Clicking on any hyperlink (in the Assay Procedure, Process Flow Chart, or Tools Windows) will open a configuration dialog for the selected device or material. After the equipment or material is configured, the associated hyperlink becomes green.

Usability Evaluation

Two usability evaluations were conducted, with the first focusing on the new bar coder action/configuration dialog (Kaber et al., 2006) and the second dealing with the prototype method editor interface.

We reviewed various usability evaluation methods for validating our approach to the enhancement of the existing HTS control interface. We considered survey methods, as Neale and Carroll (1997) advised using surveys to evaluate metaphor-based interface design. Furthermore, Anderson, Smyth, Knott et al. (1994) used a questionnaire to evaluate a metaphor-based interface by requiring users to identify the degree of agreement of, and to make confidence ratings on, source and target domain pairings. This allowed the designers to reason about the effectiveness of source-target interactions and their associated mappings. However, this type of assessment is largely focused on evaluation of the metaphor and less on the resulting usability of the interface design based on the metaphor. For the purposes of our research, we decided to use the System Usability Scale (SUS; Brooke, 1996) to capture user responses pertaining to satisfaction with the new prototype HTS interfaces. This survey has been used in several empirical assessments of usability (see Brooke, 1996) and provides a basic indicator of user preferences among interface alternatives. We expected that such a method would reveal whether users preferred our new interface to the existing software design, as influenced by the match between the system and the selected real-world analog (the cookbook recipe and cooking).

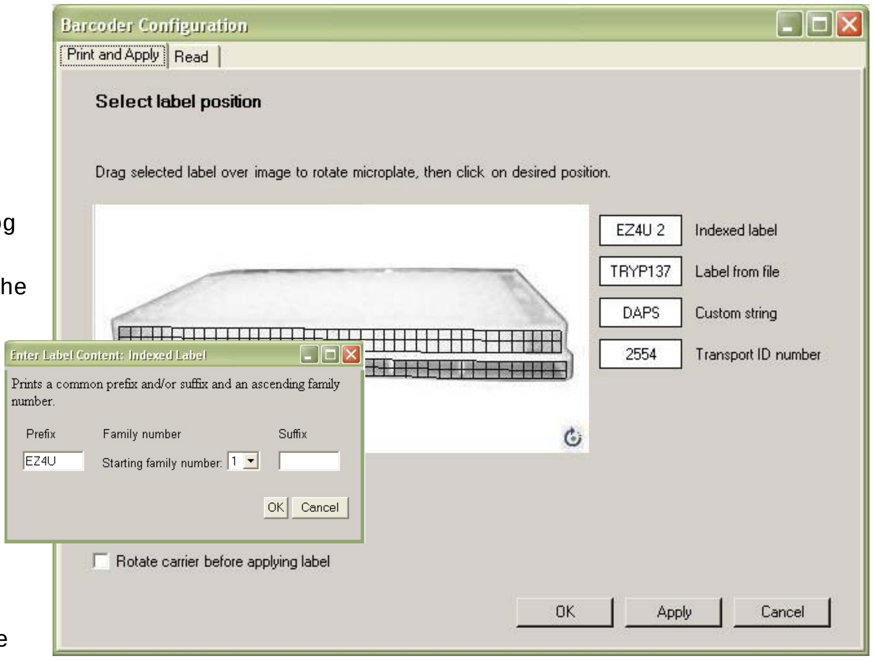

Barcoder Configuration
Print and Apply
Read
Select label position
Drag selected label over image to rotate microplate, then click on desired position.
EZ4U 2
Indexed label
TRYP137
Label from file
DAPS
Custom string
2554
Transport ID number
Enter Label Content: Indexed Label
Prints a common prefix and/or suffix and an ascending family number.
Prefix
Family number
Suffix
EZ4U
Starting family number: 1
OK
Cancel
Rotate carrier before applying label
OK
Apply
Cancel

Figure 4. New bar coder interface.

Erickson (1990) suggests conducting usability assessments by identifying user problems with existing systems in the context of actual work tasks. We thought this approach would also be useful for determining whether users encountered more or fewer problems in simulated method programming using the metaphor-based prototype versus the existing screening process control software.

Erickson (1990) also encourages the use of scenarios for performing similar tasks with different interface alternatives involving different interaction methods. We also applied this approach by comparing the new metaphor-based prototype to the existing (SAMI®) method editor in developing a HTS protocol. The scenario-based evaluation allowed us to assess if users of the new prototype made fewer errors than users of the existing interface and if they were able to complete the scenario in less time.

Methodology

The first usability study involved five domain experts comparing the new bar-coder dialog with the existing software. The new dialog (Figure 4) was designed based on the results of our AH modeling and GDTA and reflected Nielsen's (1993) usability heuristics, such as providing prompt feedback. Three male and two female biopharmacologists and process engineers at CELISCA were asked to select and configure a barcode label for application to micro-plates in a HTS process using both interfaces. The software underlying the interfaces recorded performance data, including time-to-subtask completion and number of errors.

Once experts completed the task, they were asked to fill out the SUS for both the existing and the new bar coder dialogs. The survey included statements on 10 characteristics of interfaces (complexity, consistency, learnability, ease of use, and so on). The subjects rated their degree of agreement with statements like, "I think this interface is unnecessarily complex," on a scale from 1 (strongly disagree) to 5 (strongly agree). They were also asked to provide written comments on the advantages and disadvantages of each interface alternative.

In the second study, we evaluated our metaphor-based approach to the redesign of the entire HTS method editing software. We wanted to determine whether the cookbook metaphor might enhance interface usability. Again, we recruited five domain experts (biopharmacologists and process engineers) at CELISCA to evaluate both the new prototype and the SAMI® method editor interfaces. Their ages ranged from 31 to 41 and three were male. They were novices when it came to using the new prototype.

In addition to our work with experts at CELISCA, five graduate students from North Carolina State University, between 23 and 40 years old, were recruited to evaluate only the SAMI® method editor interface in terms of usability. Four of the students were male. They were all novices at using the software, but some had previously seen the application interface. We made a comparison of the new prototype interface design, as evaluated by the CELISCA personnel, with the original SAMI® method editor interface evaluated by the students in terms of errors committed with the applications and the subjective satisfaction ratings.

At the beginning of the study, each participant was provided with an introduction to the interface being evaluated and its purpose. The participants were also given brief user manuals, which included an overview and descriptions of the types of tasks, devices, and software and how they are used for developing new methods or controlling devices. Subsequent to familiarization with the interface, the participants were required to complete scenarios in which they programmed a new assay procedure for the HTS line.

The users used the features of the existing SAMI® and the new metaphor-based prototype for programming the processes of micro-plate labeling, ingredient transfers, and incubation. They were provided with step-by-step instructions for following the scenarios, including explanations of the purpose of each step.

During completion of the scenarios by the expert biochemists from CELISCA, time-to-subtask completion and the number and types of interface errors were recorded by a Javascript embedded in an online version of the new metaphor-based prototype. A semi-functional prototype of the SAMI® method editor was also developed in Java because of access limitations to the actual installation of the software at CELISCA and to allow us to remotely collect time-to-subtask completion and the number of interface errors.

Subtasks in method programming included, for example, configuring the bar coder for plate labeling and configuring the incubator for sample heating. After completing the task scenarios, CELISCA users filled-out the SUS questionnaires for the existing application and new prototypes. Since the NCSU users were novices to the SAMI® interface, they only evaluated and completed SUS questionnaires for that interface. They were asked to rate the interface and dialogs in terms of the 10 characteristics, and to comment on their shortcomings and advantages.

Although the scenarios used with each interface were designed to be similar in steps, task performance with the prototype and the SAMI® editors was not identical (from a functional perspective) because of certain interface features. For example, to configure the incubator, in both scenarios users were able to set the incubation time. However, the incubator interface as part of the metaphor-based prototype also allowed users to set variables, such as temperature and incubator gas mixture, which the SAMI® interface does not include (in the existing system, these variables are set at the physical incubator control panel). Thus, more scenario steps were required for interacting with the incubator dialog in using the new interface than in using the existing SAMI® interface. Therefore, we could only compare performance results for tasks that were functionally similar across both interfaces. These included: (1) assay development, which involved dragging and dropping devices with the existing interface and text entry in the new interface; and (2) bar coder configuration, which involved selecting a label location and content in both interfaces.

Results

Table 2 presents the subtask performance times for the new bar coder prototype and SAMI® bar coder dialog for the CELISCA participants in the first study. A one-sided Wilcoxon Mann-Whitney test revealed no significant differences in performance times for the bar-coder configuration procedure among the two interfaces ($p > 0.05$). No errors were committed while using either bar-coder interface. (Table 2 only presents data on those subtasks for which statistical comparisons were considered valid, based on functional similarities across the SAMI and new prototype interfaces.)

The subjective rating survey yielded scores for the redesigned bar-coder dialog and the existing bar-coder action/configuration dialog of 4.04 and 3.24, respectively. A one-sided Wilcoxon Mann-Whitney test showed satisfaction ratings to be significantly higher for the redesigned bar coder dialog (*p*=0.0397). The expert biochemists commented that the new interface provided for graphic-oriented control, displayed the micro-plate barcode position in a clearer manner, presented less unnecessary information, offered "wizard"-like popups, and was generally easy to understand and more intuitive. The participants said that the existing software control mechanisms were not state-of-the-art (for example, no popups) and that the interface was not as interactive as the new prototype. The main disadvantage of the prototype that the domain experts noted was that the graphical depiction of the micro-plate was not realistic enough (Kaber et al., 2006). Based on these data, we inferred that the design of the new prototype dialog was an improvement over the existing dialog from a user satisfaction perspective and that the CTA-based approach to interface design guideline formulation and prototype development was effective.

Table 2. Performance times for prototype and SAMI® bar coders (in seconds).

Task	**SAMI® Bar Coder Interface**		**New Bar Coder Interface**	
	Mean	**SD**	**Mean**	**SD**
Select Side	3.6	1.8	9.9	13.9
Select Label	3.8	2.4	4.5	2.7
Enter Content	12.9	6.1	9.8	11.3
Whole scenario	24.8	10.4	24.3	27.5

Note: No significant differences, *p*>0.05.

With respect to the second study, Table 3 presents the subtask performance times for the new prototype and existing method editors for the expert biochemists at CELISCA. In general, the new interface appeared to allow for quicker task performance. A one-sided Wilcoxon Mann-Whitney test revealed assay development to be significantly shorter using the new interface (*p*=0.004), but there was no evidence of a difference in bar-coder configuration and total development times between the two interfaces. The incubator configuration tasks were not functionally comparable, and therefore, statistical analyses were not considered valid.

Table 3. Performance times for prototype and SAMI® method editor (in seconds).

Task	**SAMI® Interface**		**New Interface**	
	Mean	**SD**	**Mean**	**SD**
Bar coder configuration	34.2	15.3	40.1	41.7
Incubator configuration	13.2	8.4	29.4	29.1
Assay development	254.2	72.9	46.7*	59.7
Whole scenario	373.5	116.7	236.2	274.8

* Significant difference, *p*<0.05.

Novices to the two interfaces (metaphor-based interface users at CELISCA and SAMI® users at NCSU) were compared with respect to the number of errors committed during the HTS method programming scenarios. The users evaluating the new prototype made one (1) error of commission (taking an incorrect step) and no errors of omission (failing to take the correct step). Users evaluating the existing method editor made 16 commission errors and 11 omission errors. Although users made more errors using the existing interface than when using the new interface, since the two operating scenarios were not functionally identical across all steps, it is not possible to make statistical comparison of these results.

CELISCA biochemist evaluations of the new prototype, in terms of user satisfaction, were higher than the student evaluations of the SAMI® interface. The average user responses to the SUS questionnaire for the SAMI® interface and the new metaphor-based interface were 3.06 and 3.92, respectively. A one-sided Wilcoxon Mann-Whitney test confirms that satisfaction ratings were significantly higher for the new prototype ($p=0.0476$). (It is important to note here that participants at CELISCA also evaluated the existing SAMI® interface, and gave it an average score of 4.14. Their SAMI® and new prototype scores were not significantly different. However, it is difficult to make any inferences based on these results, since CELISCA evaluators represented novices with respect to the new prototype, but were highly experienced with the SAMI® interface.)

The five evaluators at CELISCA indicated the main advantage of the new metaphor-based prototype was a single, comprehensive interface for assay development. They were not required to make use of several software packages, as with the SAMI® application, to develop an assay from start to finish. The experts also felt that the metaphor-based interface was easy to use, but was too text-oriented. Based on these results, we inferred that the new prototype design was superior to the existing method editor from a usability perspective and that our use of the Neale and Carroll (1997) metaphor-based approach to the redesign of the HTS control software was generally effective.

Discussion and Conclusions

Three usability goals were identified for our metaphor-based redesign of the HTS method editor interface: efficiency in task completion, error prevention, and user satisfaction. The goal of improved efficiency can be influenced by the development of recognizable system components (for example, interactive components) and the match of the system components to analogous processes in the real world (for example, NLP).

Furthermore, the structuring of the different windows and automatic content updating (around a core structure within the metaphor-based redesign) assist the user in linking the interactive components within the interface to each other and to the cookbook metaphor itself. The insignificant results of the usability evaluation of the bar-coder dialogs revealed that the experts can adapt quickly to the new interface to achieve performance comparable to the use of the existing software with training. The large difference in assay development times for the two method editor interfaces was due to the differences in the interaction provided by the metaphor.

The goal of increased effectiveness can be linked to the development of system components that support both novice and expert user understanding of tasks (for example, the process flow chart). The usability evaluation indicated that novice users made more errors using the existing HTS method editor interface than when using the metaphor-based interface. This represents a reduction in all errors, as any uncorrected errors in the assay development will lead to later performance problems.

Finally, the goal of user satisfaction can be linked to the use of feedback and success measures (for example, task completion). The usability evaluation indicated that novices found the metaphor-based interface to better support their completion of the HTS task.

In general, these results demonstrate the effectiveness of Neale and Carroll's (1997) metaphor-based approach to developing usable interfaces in a very unique and complex domain, that is, biological compound screening for drug development. Though it is possible that other factors, such as the novelty of the design, could account for the greater user satisfaction with the metaphor-based interface, it is unlikely that such factors influenced the effectiveness of this interface, as evidenced by the smaller number of errors committed.

We did find some steps in Neale and Carroll's (1997) approach to be more useful than others. For example, those steps that deal with mismatch handling, in some cases, may be less relevant. That is, a metaphor may be robust enough such that no significant mismatches are apparent. Furthermore, if there are metaphor mismatches, it may be difficult to translate Neale and Carroll's strategies (for example, teaching metaphor modification rules, encouraging exploration of system features, and so on) into structured recommendations for managing them. However, Hamilton (2000) cautions against ignoring mismatches in interface design, claiming that users can experience a degree of cognitive dissonance if they are not handled properly.

This study expanded on the methodology proposed by Neale and Carroll (1997) by integrating the use of formalized CTA methods to guide the novel metaphor-based design with the purpose of enhancing supervisory control interfaces in life sciences automation.

The combination of AH modeling and GDTA with the Neale and Carroll approach proved to be effective in identifying the cookbook metaphor and the development of usable interfaces for the target domain, as demonstrated through the achievement of the usability goals. Related to this, several guidelines have been published in the attempt to provide designers with different approaches to applying metaphors to interface design (e.g. Alty et al., 2000; Carroll & Mack, 1985). Although none of these guidelines explicitly calls for the use of CTA methods, some do require a functional description of the system, as can be achieved through AH models.

Several caveats are important to discuss with respect to this study. First, we used a remote usability evaluation method (Dumas, 2003) to assess performance with the metaphor-based prototype. We provided participants at CELISCA with detailed instructions for accessing the new prototype (online) and evaluating its usability, but we were not physically present in Germany during the evaluations to directly observe biochemist interface behaviors. However, this method may have produced conservative results because we did not provide real-time assistance or explain design decisions to the evaluators, which might have inflated satisfaction ratings, for example. Beyond this, remote usability evaluation can be considered similar to real-life situations, in which novice users try out new software applications alone and learn how to use them by trial and error.

Second, it would have been helpful to have a large group of domain experts, who are familiar with HTS (but not with SAMI®), evaluate both interfaces. This would have allowed for a more direct comparison of error results with the method editors. The graduate students at NCSU who evaluated the SAMI® only had some familiarity with life sciences automation, while the experts at CELISCA may have been influenced in usability ratings and performance by their prior knowledge of the SAMI® software. Relevant to this, it is difficult to recruit potential participants – HTS biologists, chemists and process engineers – for such experiments, since this expert population is very small.

Finally, the two participant groups who evaluated the method editors represented novices. As a result, individual differences may have played a large role in our outcomes, such as the number of errors committed. An alternative approach to the usability evaluation could involve training subjects to a predetermined level of proficiency with the software and prototype before assessing their performance.

In the future, it would be interesting to develop a fully functional prototype of the metaphor-based method editor interface and to carry out a more comprehensive evaluation, as described above, with a larger participant population to fully understand any errors that could result from the metaphor-based interface design.

Other usability evaluation methods could be used as well, such as verbal protocols (Wiedenbeck, Lampert & Scholtz, 1989), to allow for detailed user behavior and error analysis. Another evaluation method that could be considered for future work is cognitive modeling techniques, such as GOMS (goals, operators, methods, selection rules; Card, Moran, & Newell, 1983). GOMS models can predict time-to-task completion (Card et al., 1983), time to learn how to perform a task (Kieras, 1999), and task complexity, and can thus replace user testing at early interface design stages.

One of the issues related to interface design for cognitive tasks, such as HTS method programming, is that software manufacturers develop applications that may not “speak the users’ language” (Nielsen, 1993). Human factors experts, who have studied operator information needs through CTA methods and have detailed knowledge of usability evaluation methods, can act as mediators between biopharmacologists and software developers to better specify interface requirements. By using human-computer interaction methods to explicitly represent biopharmacologist needs via prototypes, manufacturers may be able to more effectively establish software design requirements and specifications.

Last, one ancillary result of this work is the development of recommendations for redesigning actual cookbooks. The combination of a method flow chart with the mock-NLP assay scratch pad in the metaphor-based interface suggested that the development of more flow charts of cooking processes in cookbooks might be helpful to chefs. Furthermore, not all cookbooks include lists of tools and ingredients that support work organization prior to, and while, cooking. Adding these components to cookbook recipes may increase their usability.

Practitioner's Take Away

- Metaphors can be a powerful tool for guiding interface design in specific domains. They enable users to map knowledge from a familiar source domain to an unfamiliar target domain, thus they are particularly useful for novices.
- The combination of formalized CTA methods with existing frameworks for developing metaphor-based interfaces can be helpful in developing, evaluating, and refining an appropriate metaphor.
- Mismatches between software environments and the real-world analog defining a design metaphor can be effectively addressed by using composite metaphors, including object and process references.
- Metaphor-based interfaces can significantly promote system usability, in particular effectiveness, efficiency, and user satisfaction in complex systems.

Acknowledgments

A National Science Foundation (NSF) Information Technology Research Grant (no. 046852) supported this research. Ephraim Glinert was the technical monitor for the NSF. The views and opinions expressed in this paper are those of the authors and do not necessarily reflect the views of the NSF. We thank the University of Rostock and CELISCA for providing us with access to high-throughput biological screening systems and supporting the research through allocation of biopharmacologist and process engineer time to the effort.

References

Alty, J.L., Knott, R.P., Anderson, B. & Smyth, M. (2000) A framework for engineering metaphor at the user interface. *Interacting with Computers,* 13, pp. 301-322.

Anderson, B., Smyth, M., Knott, R.P., Bergan, M., Bergan, J., & Alty, J.L. (1994) Minimizing conceptual baggage: Making choices about metaphor. In *People and Computers IX, Proceedings of HCI '94* (pp. 179-194). Huddersfield, UK: Cambridge University Press.

Brooke, J. (1996). SUS: A 'quick and dirty' usability scale. In Jordan, P. W., Thomas, B., Weerdmeester, B. A., & McClelland, I. L. (Eds.), *Usability Evaluation In Industry*, pp. 189-194. London, UK: Taylor & Francis.

Card, S., Moran, T., & Newell, A. (1983) *The Psychology of Human-Computer Interaction*. Hillsdale, New Jersey: Erlbaum.

Carroll, J.M., & Mack, R.L. (1985) Metaphor, computing systems and active learning. *International Journal of Man-Machine Studies, 22*(1), pp. 39-57.

Dale, R., Moisl, H., & Somers, H. (Eds.) (2000) *Handbook of Natural Language Processing*. New York: Marcel Dekker.

Dumas, J. S. (2003) User-based evaluations. In J. Jacko and A. Sears (Eds.), *The Human-Computer Interaction Handbook* (pp. 1094-1115). Mahwah, NJ: Erlbaum.

Dutton, R.T., Foster, J.C., & Jack, M.A. (1999) Please mind the doors – Do interface metaphors improve the usability of voice response services? *BT Technology Journal, 17*(1), pp. 172-177.

Endsley, M. R. (1993). A survey of SA requirements in air-to-air combat fighters. *International Journal of Aviation Psychology, 3*(2), pp. 157-168.

Endsley, M. R., Bolstad, C. A., Jones, D. G., & Riley, J. M. (2003) Situation awareness oriented design: From user's cognitive requirements to creating effective supporting technologies. In *Proceedings of the 47th Annual Meeting of the Human Factors & Ergonomics Society* (pp. 268-272). Santa Monica, CA: Human Factors & Ergonomics Society.

Entzian, K., Allwardt, A., Holzmüller-Laue, S., Junginger, S., Roddelkopf, T., Stoll, N., & Thurow, K. (2005) Automationslösungen für biologische und chemische Screeningverfahren. Proceedings *GMA-Kongress "Automation als Interdisziplinäre Herausforderung"*, Baden-Baden, June 7-8, pp. 235-242.

Erickson, T.D. (1990) Working with interface metaphors. In B. Laurel (Ed.), *The Art of Human-Computer Interface Design* (pp. 65-73). Reading, MA: Addison-Wesley Publishing Company, Inc.

Hamilton, A. (2000) Interface metaphors and logical analogues: A question of terminology. *Journal of the American Society for Information Science, 51*(2), pp. 111-122.

Harper, B., Slaughter, L., & Norman, K. (1997) Questionnaire administration via the WWW: A validation & reliability study for a user satisfaction questionnaire. In *Proceedings of WebNet 97: International Conference on the WWW, Internet, and Intranet*, Toronto, ON, Canada.

Kaber, D.B., Segall, N., Green, R.S., Entzian, K., & Junginger, S. (2006) Using multiple cognitive task analysis methods for supervisory control interface design in high-throughput biological screening processes. *International Journal of Cognition, Technology & Work* (Available online: DOI 10.1007/s10111-006-0029-9).

Kieras, D. (1999) *A guide to GOMS model usability evaluation using GOMSL and GLEAN3*. University of Michigan, Ann Arbor, Michigan.

Kuhn, W., & Blumenthal, B. (1996) Spatialization: Spatial metaphors for user interfaces. In A.U. Frank (Ed.), *Geinfo Series, 8*, Vienna, Austria: Technical University Vienna (Tutorial notes from the ACM Conference on Human Factors in Computer Systems (CHI ’96).

Neale, D.C., & Carroll, J.M. (1997). The role of metaphors in user interface design. In M. Helander, T.K. Landauer, and P. Prabhu (Eds.), *Handbook of Human-Computer Interaction* (pp. 441-462). Amsterdam: Elsevier Science.

Nielsen J (1993) Usability Engineering. Academic Press, Boston.

Norman, D.A. (1995) The psychopathology of everyday things. In R. M. Baeker, Grudin, J., Buxton, W. A. S., and Greenberg, S. (Eds.), *Readings in Human-Computer Interaction: Toward the Year 2000* (pp. 5-22). San Francisco, CA: Morgan Kaufmann Publishers.

Rasmussen, J. (1985) The role of hierarchical knowledge representation in decision-making and system management. *IEEE Transactions on Systems Man and Cybernetics, 15*, pp. 234-243.

Schappert, A., Sommerlad, P., & Pree, W. (1995) Automated Support for Software Development with Frameworks. *Symposium on Software Reusability, SSR ’95, ACM Software Engineering Notes*, Seattle, WA: Association of Computing Machinery.

Thurow, K., Entzian, K., & Eberlein, G. (2004) Toxicological and pharmacological evaluation of new drug candidates by in vitro robotic high throughput cell assays. *Journal of the Association for Laboratory Automation, 9*(3), pp. 159-162.

Usher, J.M., Kaber, D.B. (2000) Establishing information requirements for supervisory controllers in a flexible manufacturing system using goal-directed task analysis. *Human Factors & Ergonomics in Manufacturing*, *10*(4), pp. 431-452.

Vicente, K.J. (1999) Wanted: Psychologically relevant, device- and event-independent work analysis techniques. *Interacting with Computers, 11*, pp. 237-254.

Wiedenbeck, S., Lampert, R., & Scholtz, J. (1989) Using protocol analysis to study the user interface. *Bulletin of the American Society for Information Science, 15*(5), pp. 25-26.

David B. Kaber is a professor in the Edward P. Fitts Department of Industrial & Systems Engineering at North Carolina State University and a visiting professor at the Center for Life Sciences Automation of the University of Rostock. His research interests include human-automation interaction, interface design for complex systems and analysis of situation awareness. He received his Ph.D. in industrial engineering from Texas Tech University in 1996.

Noa Segall is a research associate in the Department of Anesthesiology and the Human Simulation and Patient Safety Center at Duke University Medical Center. She received her Ph.D. in Industrial & Systems Engineering at North Carolina State University in 2006. Her research interests include human factors in medical systems, human-computer interaction, and human factors in automation design.

Rebecca S. Green is a doctoral candidate in the Department of Psychology at North Carolina State University. She received a B.S. in biomedical engineering at the University of Pittsburgh and an M.A. in experimental psychology at East Carolina University. Her research interests include interface design in computer-based instruction and human-automation interaction in medicine.