LCdr. Michael Shumberger, USN (ret), Dr. Andrew Duchowski, and Mr. Kevin Charlow

Human Factors Engineering in Command, Control & Intelligence (C2I) User Interface Development Processes: Use of Eye Tracking Methodology

ABSTRACT

The Space and Naval Systems Center Charleston (SSC-C) engineers and supports warfighter focused Command, Control and Intelligence (C2I) applications through continuous process and product improvement. This paper discusses implementing the Usability Engineering Life-Cycle (UELC) [Mayhew] for C2I software development at SSC-C. Human Factors concepts, as they relate to software development, are briefly reviewed with references to international standards. Following the discussion of general HFE concepts, specific User-Centered Design (UCD) tasks and metrics are introduced into the SSC-C Software Development Process. Lastly, we discuss the use of eye tracking methodology in the Software Development Process, as applied in the context of UCD as an instantiation of a Human Factors Engineering (HFE) process.

1. INTRODUCTION

The Space and Naval Systems Center Charleston (SSC-C) is responsible for providing engineering support to Navy, Department of Defense (DoD), and other government agencies. SSC-C's mission is to "… engineer, deliver, and support fully integrated, interoperable information technology systems, through dedicated warfighter focus, professional employee development, industry partnership, and continuous process and product improvement." As such, SSC-C performs acquisition, systems engineering, software engineering, and security engineering functions in accordance with the Defense Acquisition System Directive, 5000.1 and the Operation of the Defense Acquisition System Instruction, 5000.2.

Best industry practices indicate that the optimal approach to developing quality, cost effective system support is through use of rigorous, disciplined development processes. Accordingly, SSC-C has defined a Software Development Process that combines the processes and activities of international standard ISO/IEC 12207, *Software Life-Cycle Processes* with the best practices of Systems Engineering Institute's (SEI) *Capability Maturity Model® Integration (CMMI®) for Systems Engineering and Software Engineering, Version 1.1, Staged Representation.*

This paper discusses implementing the Usability Engineering Life-Cycle (UELC) [Mayhew] for software development at SSC-C. Human Factors concepts, as they relate to software development are briefly reviewed with references to the ISO/IEC 9126 and ISO/IEC 9241-11 standards. Following the discussion of general HFE concepts, specific User-Centered Design (UCD) tasks and metrics are introduced into the SSC-C Software Development Process. Lastly, we discuss the use of eye tracking methodology in the Software Development Process, as applied in the context of UCD as an instantiation of a Human Factors Engineering (HFE) process.

2. PURPOSE

Where practicable and cost effective, system designs shall minimize or eliminate system characteristics that require excessive cognitive, physical, or sensory skills; entail extensive training or workload-intensive tasks; result in mission-critical errors; or produce safety or health hazards. The purpose of the Human Factors Engineering (HFE) Process Model for Software Development is to assist the Project Engineer by ensuring that human factors engineering/cognitive engineering is employed

during systems engineering over the life of the program to provide for effective human-machine interfaces and to meet Human-Systems Integration (HSI) requirements. The HFE processes allow flexibility while clearly defining activities that are required for process discipline, consistency, and management insight. These activities, as supported by international standards and accepted practices, shape a new engineering field at the crossroads of software engineering and human-computer interaction, the field of Usability Engineering (UE).

At the core of UE practices lies iterative evaluation of the software being developed and conducted throughout the life of the software system. Evaluation of the system depends on qualitative and quantitative metrics obtained during testing, performed throughout the software design process. This is a key point: if user feedback is obtained only at the end of the development process, e.g., upon delivery, it is likely that a great deal of effort may be wasted if intended users are not satisfied with the delivered software product. UE thus attempts to ensure user satisfaction, along with software Compliance (to standards), Operability, Understandability, Learnability, and Attractiveness (COULA attributes) by requiring user involvement in software quality evaluation at early stages of the process. Since the software product under development will not be complete and available for full testing at the outset, UE relies on early conceptual renditions of the product prior to its full implementation. Such early renditions may consist of pencil-and-paper sketches of the User Interface (UI), early UI mock-ups, or early functioning prototypes of the UI.

UE practices are not without cost, however. In general, evaluation of the UI at any stage requires effort. Effort may be needed to endow the software with additional functionality (e.g., recording of user actions, eye movements, and/or UI events), development of prototypical UI components, and performance of formal usability tests (i.e., human subject experiments). Besides additional potential coding effort, additional knowledge of experimental design and statistical analysis may be required. Hence, UE practices generally require addition of considerable effort in the development process, particularly during early stages of the process, specifically during design phases. However, the expected benefit of adoption of UE practices is greater user satisfaction and widespread adoption of the eventual software product.

A significant advantage obtained through UE can lead to evaluation of the Software Development Process at CMMI Process Maturity Level 4. A capability level 4 process is a quantitatively managed defined process that is controlled using statistical and other quantitative techniques. In general, UE involves three global strategies [Mayhew]:

- Early focus on users and tasks
- Empirical measurement (software evaluation)
- Iterative design

Usability is a measurable characteristic of a product user interface that is present to a greater or lesser degree. To achieve usability, software design needs to take into account and be tailored around a number of factors, including:

- Cognitive, perceptual, and motor capabilities and constraints of intended users
- Special and unique characteristics of the intended users
- Unique characteristics of the users' physical and social work environment
- Unique characteristics and requirements of users' tasks supported by the software
- Unique capabilities and constraints of the chosen software and/or hardware and platform

Usability Engineering is a discipline that provides structured methods for achieving usability with roots in several disciplines including cognitive psychology, experimental psychology, ethnography, and software engineering. Cognitive psychology is the study of human perception (vision, hearing, etc.) and cognition (memory, learning, reasoning, etc.). UE draws knowledge about these aspects of human information processing and applies it to

software design. Basing software design on this knowledge leads to software whose functionality is intuitive (*understandability*), software that is easier to learn (*learnability*), easier to use (*operability*), visually and functionally appealing (*attractiveness*). Experimental psychology uses empirical methods to measure human behavior. UE draws on these methods to measure user performance and satisfaction with the software interface. Software engineering defines application requirements, goals, and iterative testing cycles until goals are met. Basing software design on this methodology leads to software designed under an engineering-like process and software that is compliant (*compliancy*) with the guidelines and standards identified by the process.

3. USABILITY ENGINEERING PROCESSES AND STANDARDS

It has been observed that the user interface is often the single most important factor in the success of a software project. It has been estimated that between approximately 50% and 80% of all source code developed is concerned with the human-computer interface. [Avouris] There are a wide variety of development and evaluation techniques that have been shown to lead to more usable software applications.

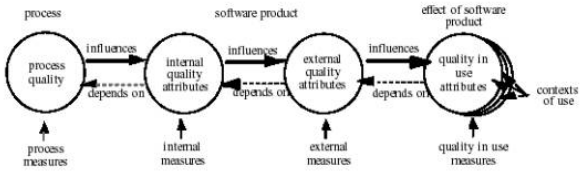

Figure 1: ISO 9126 Quality Life Cycle Model

ISO/IEC Standard 9126 relates to software quality and development of measures as they pertain to the context of use of the software. At the same time many practical techniques for measuring usability have been proposed in the interactive software development lifecycle. Usability was originally related to making systems easy to use and easy to learn, as well as supporting the users during their interaction with computer equipment. There have been however many attempts to relate the term to more attributes and metrics.

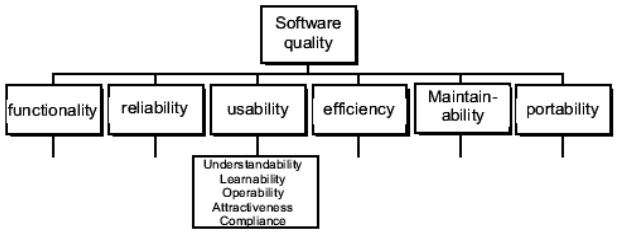

Figure 2: Usability as an Attribute of Software Quality According to ISO 9126. [Avouris]

In ISO/IEC 9241-11 draft standard Usability is defined as the "extent to which a product can be used with *effectiveness*, *efficiency* and *satisfaction* in a specified context of use" (ISO 9241). [Avouris] The attributes which a product requires for usability depend on the nature of the user, task, and environment. A product has therefore no intrinsic usability, only a capability to be used in a particular context. Usability cannot be assessed by studying a product in isolation. There are three potential ways in which the usability of a software product could be measured, according to (ISO 9241):

- By analysis of the features of the product, required for a particular context of use. Usability could be measured by assessing the product features required in a particular context, meaning, for example, how particular interface features are used during specific tasks.

- By analysis of the process of interaction. Usability could be measured by modeling the interaction between a user carrying out a task with a product. This approach leads to cognitive modeling, providing insights into a user's cognitive processes during software use.

- By analyzing the effectiveness and efficiency, which results from use of the product in a particular context, and measuring the satisfaction of the users of the product. These are direct measures of the attributes of usability. If a product is more usable in a particular context, usability measures will be better (see Figure 1).

According to standard ISO/IEC 9126, usability is an attribute of software quality. According to this standard, the term is used to refer to the capability of a product to be used easily. This

corresponds with the definition of usability as a software quality: "*a set of attributes of software which bear on the effort needed for use and on the individual assessment of such use by a stated or implied set of users*". This is related to the capability of the software product to be understood, learned, used and be attractive to the user, when used under specified conditions. It is observed that there is an inter-relation between some aspects of product functionality, reliability and efficiency that will also affect usability, but for the purposes of ISO/IEC 9126 are not classified as usability. It is also observed that users may include operators, end users and indirect users who are under the influence of or dependent on the use of the software. Usability should address all of the different user environments that the software may affect, which may include preparation for usage and evaluation of results.

Usability is further analyzed in standard ISO/IEC 9126 in *Understandability, Learnability, Operability, Attractiveness and Compliance*. These are briefly described in the following:

- *Understandability* is defined as the capability of the software product to enable the user to understand whether the software is suitable, and how it can be used for particular tasks and conditions of use. This attribute will depend on the documentation and initial impressions given by the software.

- *Learnability* is the capability of the software product to enable the user to learn its application.

- *Operability* is the capability of the software product to enable the user to operate and control it. Aspects of suitability, changeability, adaptability and installability may affect operability. Also this attribute corresponds to controllability, error tolerance and conformity with user expectations as defined in ISO 9241-10. For a system, which is operated by a user, the combination of functionality, reliability, usability and efficiency can be measured externally by quality in use.

- *Attractiveness* is the capability of the software product to be attractive to the user. This refers to attributes of the software intended to make the software more attractive to the user, such as the use of color and the nature of the graphical design.

- *Compliance* to standards and guidelines refers to the capability of the software product to adhere to standards, conventions, style guides or regulations relating to usability.

In Figure 3 the key quality factors according to ISO 9126 are shown.

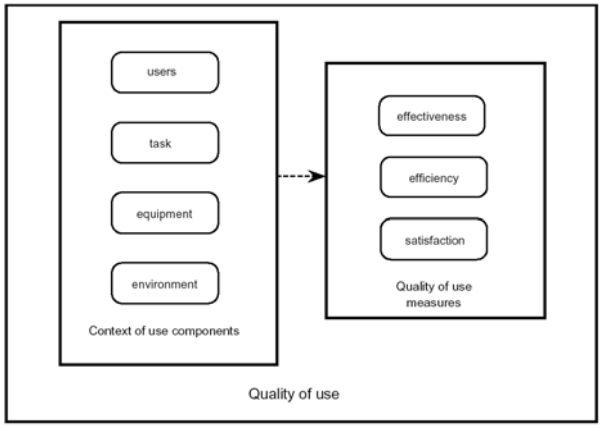

Figure 3: Usability Factors

4. USABILITY ENGINEERING METRICS

Many attempts have been reported to further analyze software usability in more practical measurable terms. Usability has been analyzed in terms of: *easiness and speed of learning of system use, efficiency to use, easiness to remember system use after certain period of time, reduced number of user errors and easy recovery from them, subjective satisfaction of users*. [Nielsen] While the emphasis in SSC-C UE techniques is on ease of measure of usability factors, some provision is made to accommodate various classes of users, like novice (easiness and speed of learning), occasional users (remember use) and expert users (efficiency of use). Many frameworks have been proposed to measure usability according to these dimensions and evaluate interactive software systems. Also many attempts have been made to relate these

aspects with system performance. For instance, measure of performance can be considered the measure of improved learning, e.g., better understanding by the user of the task and better relation of the task to the available tools and operators. An overview of techniques to measure usability-related factors is included in the following sections.

4.1 Inspection Methods

Usability inspection methods are evaluation methods involving usability experts examining the software UI. Many inspection methods can be based on specifications that have not necessarily been implemented yet, so they can be performed early in the software lifecycle, though some methods also address issues like the overall system usability concerning the final prototype. The main methods of this category are:

- *Heuristic evaluation* involves usability specialists who judge whether each dialogue element follows established usability principles (the "heuristics").

- *Cognitive walkthrough* uses a detailed procedure to simulate task execution at each step through the dialogue, determining if the simulated user's goals and memory content can be assumed to lead to the next correct action.

- *Pluralistic walkthrough* uses group meetings where students, developers, and usability experts step through a learning scenario, discussing each dialogue element.

- *Feature inspection* lists sequences of features used to accomplish typical tasks, checks for long sequences, cumbersome steps, steps that would not be natural for students to try, and steps that require extensive knowledge/ experience in order to assess a proposed feature set.

- *Standards inspection*, during which experts inspect the interface for compliance with certain standards. This can involve user interface standards as well as domain-specific software standards, departmental standards if they exist, etc.

- *Guidelines checklists* help ensure that usability principles will be considered in a design. Usually, checklists are used in conjunction with a usability inspection method. The checklist gives the inspectors a basis by which to compare the product.

4.2 Testing Methods

Tests measure system performance against pre-defined criteria. These criteria are defined according to the usability attributes, suggested by the usability standards and empirical metrics discussed in the previous section. Typically individual users are observed performing specific tasks with the system. Data are collected on measured performance. For example, time required to complete the task or number of errors made. Selection of appropriate users and representative tasks is essential. Also a properly designed and organized usability laboratory is important. The most widely accepted usability testing techniques are:

- *Thinking Aloud Protocol* is a technique widely used during usability testing. During the course of a test, the participant is asked to vocalize his/her thoughts, feelings, and opinions while interacting with the software, performing a task - part of a user scenario. This technique may be particularly difficult to use with some user groups, like young students, who are distracted by the process, however it provides a valuable insight to user cognitive processes, while interacting with the software.

- *Co-discovery* is a type of usability testing where a group of users attempt to perform tasks together while being observed, simulating typical work process, where most people have someone else available for help. This can be particularly suitable in many work scenarios.

- *Performance measurement*. Some usability tests are targeted at determining hard, quantitative data. Most of the time this data is in the form of performance metrics, e.g., required time to execute specific tasks. The ISO 9241 promotes in particular a usability evaluation approach based on measured performance of pre-determined usability metrics.

- *In-field, or ethnographic studies* concern observation of the users performing their tasks in their usual environment of study/work. These techniques have the advantage of the natural user performance and group interaction however they present limitations in terms of measuring performance, since the necessary testing equipment cannot be used in a typical workplace.

4.3 Inquiry Methods

Inquiry methods (based on *questionnaire and interview protocols*) prompt the users by asking direct questions about the system. The users' ability (or lack of) to answer questions can help evaluators decide about parts of the system interface that present difficulties for the users.

While inquiry methods can be used to measure various usability attributes, their most common use relates to measurement of user satisfaction. A known technique for measuring user satisfaction is through SUMI, the Software Usability Measurement Inventory, developed by a research group of the University College Cork, to measure user satisfaction, and hence assess user perceived software quality. SUMI is an internationally standardized 50-item questionnaire, available in several languages. It takes a maximum of 10 minutes to complete and needs only small user sample sizes.

The results that SUMI provides are based on an extensive standardization database built from data of various software products such as word processors, spreadsheets, CAD packages, communications programs etc. SUMI results have been shown to be reliable, and to discriminate between different kinds of software products. In particular, the SUMI database allows evaluation of a product against what is considered to be the prevailing market norm, and the statistical background to SUMI enables the analyst to pinpoint quite precisely the relative standing of the product being assessed to the market as a whole. SUMI results are analyzed into 5 sub-scales: Affect, Efficiency, Helpfulness, Control, and Learnability. These scales have been derived by an iterative process of factor analysis of large databases, and present a view of subjective usability for which there is a high level of empirical support.

5. USABILITY ENGINEERING WITHIN THE SOFTWARE DEVELOPMENT PROCESS

For inclusion of selected relevant UE techniques (testing methods), an appropriate Software Development Life-Cycle must be followed during the Software Development Process. The Usability Engineering Life-Cycle (UELC) [Mayhew], composed of tasks given in Figure 4, provides a suitable alternative to the traditional life-cycle models (waterfall, incremental, evolutionary, and spiral). Thus the UELC satisfies the original goal of flexibility specified in the SSC-C SDP Manual while simultaneously providing HFE principles as an option to the process.

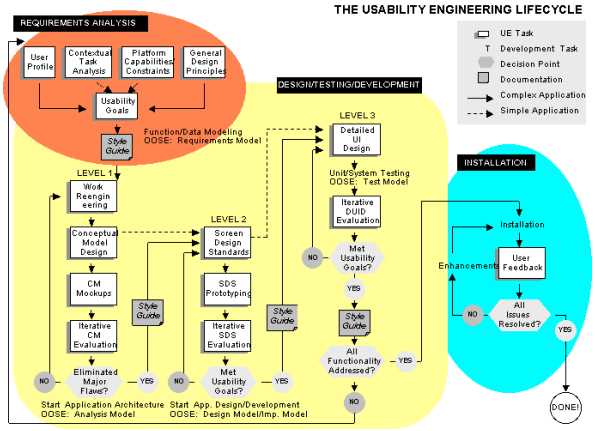

Figure 4: Usability Engineering Life-Cycle [Mayhew]

The decision to apply HFE principles, by virtue of selection of the UELC, must be made early, at Process Implementation. Following selection of the UELC, this particular life-cycle model immediately suggests application of related HFE methods and metrics at several task levels (including but not limiting to design, validation, and verification).

5.1 Process Implementation

During the Process Implementation activity, if HFE is to be applied to the Software Development Process, selection of the UELC must be made here along with appropriate

standards (ISO/IEC 9126 and ISO 9241). UELC tasks can then be mapped to the Software Development Process if the development process activities are considered as follows:

- Requirement Analysis:
  - o System Requirements Analysis
  - o System Architectural Design
  - o Software Requirements Analysis
  - o Software Architectural Design
- Design/Test/Development:
  - o Software Detailed Design
  - o Software Coding and Testing
  - o Software Integration
  - o Software Qualification Testing
  - o System Integration
  - o System Qualification Testing
- Installation:
  - o Software Installation
  - o Software Acceptance Support

System Requirements Analysis

Following the UELC, the following tasks are specified during System Requirements Analysis:

User Profiles: The purpose of this task is to establish user characteristics around which the User Interface (UI) design must be tailored. This is accomplished via questionnaires distributed to users and via interviews with stakeholders. Results of this task are documented in questionnaire/review forms, a data summary, and reported analysis and conclusions.

Contextual Task Analysis: The purpose of this task is to devise a user-centered model of work as it is currently performed and to extract from this the usability requirements for the product. This can be accomplished by conducting contextual observations or interviews, by brainstorming task scenarios, or other techniques. A task analysis document is issued at the end of this task.

Software Requirements Analysis

Usability Goals Setting: The purpose of this task is to establish qualitative and quantitative usability goals that will drive UI design. A subset of high-priority goals may be quantified to be used in usability testing as acceptance criteria. Both qualitative and quantitative usability goals are documented as a result of this task.

Software Architectural Design

Platform Capabilities/Constraints: Establish the capabilities and constraints of the technology platform, which will limit UI design alternatives. User Interface capabilities and constraints are studied with respect to the chosen technology platform and these are documented as a result.

Software Detailed Design

If HFE is to be incorporated into the Software Development process, this activity requires expansion at the three design levels of the UELC. Level I is concerned with the design of a Conceptual Model (CM) of the intended UI. Level II is concerned with the design of Screen Design Standards (SDS) for the intended UI. Level III is concerned with a Detailed UI Design (DUID). It must be noted that each level requires evaluation of each of the CM, SDS, and DUID. This evaluation, both qualitative and quantitative, is critical for adoption of HFE principals.

<u>Software Detailed Design Level I</u>

Work Reengineering: This subtask reengineers the current user work model for the purposes of realizing the potential of automation and effective support of mission-critical goals, while minimizing retraining and maximizing operability. The reengineered work model, resulting from this task, can be validated with the card sorting technique or with Task Scenario walkthroughs.

Conceptual Model Design: A coherent and rule-based, high-level UI design framework is established that sets the stage for design at lower levels. The resultant Conceptual Model Design (CMD) may be adapted from platform style guides (e.g., MS Windows, Apple Macintosh).

Conceptual Mock-ups: These preliminary mock-ups, possibly as rough as paper-and-pencil mock-ups or running prototypes support evaluation, refinement, and validation of the CMD.

Iterative Evaluation of Conceptual Model: This tasks aims to formally evaluate, refine, and validate the CMD through stages of evaluation plan, recorded evaluation data, analysis of data, and results in reporting of conclusions and recommendations for design changes.

Software Detailed Design Level II

Screen Design Standards: The objective of this task is to establish and define a set of design standards that, along with the CMD, will set the stage for Detailed User Interface Design. SDS may be adapted from platform style guides (e.g., MS Windows, Apple Macintosh).

Screen Design Standards Prototyping: Running prototypes representing Screen Design Standards, resulting from this task, support the evaluation, refinement, and validation of the SDS.

Iterative SDS Evaluation: This tasks aims to formally evaluate, refine, and validate the SDS through stages of evaluation plan, recorded evaluation data, analysis of data, and results in reporting of conclusions and recommendations for design changes.

Style Guide Development: Documentation of the CMD, the SDS, and the output from all the Requirements Analysis tasks results in one, evolving document, the Style Guide which contains the final validated CMD and SDS as well as the main results of all Requirements Analysis tasks.

Software Detailed Design Level III

Detailed User Interface Design: Design the complete, detailed UI, and provide the resultant DUID specification. The Style Guide standards are applied to design the UI at all levels of functionality.

Iterative DUID Evaluation: Evaluate, refine, and validate key subsets of the DUID through formal usability testing or usability inspection methods through stages of evaluation plan, recorded evaluation data, analysis of data. The DUID evaluation tasks results conclusions and recommendations for design changes.

Software Qualification Testing

User Feedback: Obtain usability data, applying one (or more) of a variety of objective evaluation techniques to obtain feedback from actual experienced users of the developed software. Available techniques include formal usability testing, questionnaires, interviews, focus groups, and usage studies. Results are fed back into the UI design for later releases of this or related software.

System Qualification Testing

User Feedback: Obtain usability data, applying one (or more) of a variety of objective evaluation techniques to obtain feedback from actual experienced users of the developed software. Available techniques include formal usability testing, questionnaires, interviews, focus groups, and usage studies. Results are fed back into the UI design for later releases of this or related software.

6. EYE TRACKING

There are a variety of techniques for carrying out each UELC task. The Usability Engineering (and hence HFE) tasks identified by the UELC should be carried out in a particular order and integrated within the existing software development process. For any particular UELC task, the usability practitioner (designer, developer, tester) has a set of techniques to choose from to accomplish the basic goals of that task.

It should be stressed that the focus of the UELC is evaluation. Evaluation can occur at each level of the design/test/develop iterative cycle of the UELC. As an example, at any evaluation stage where an objective evaluation technique is needed, eye tracking may be used to obtain quantifiable metrics concerning users' *scanpaths* (see below) which offer insights into users' cognitive processes. An example of a scanpath is provided in Figure 5.

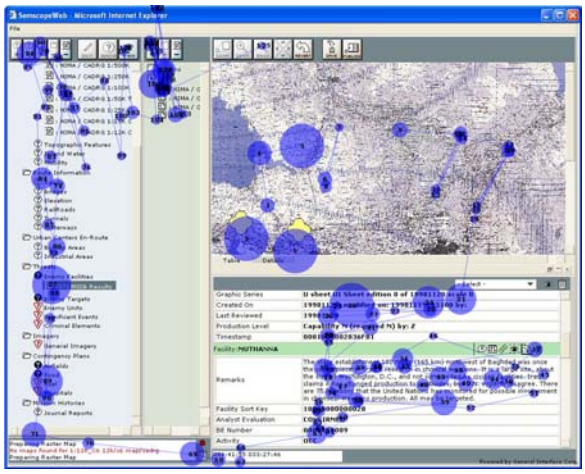

Figure 5: Example Eye Tracked Scanpath of a User Viewing a UI.

Eye tracking, with the quantitative metrics of number and duration of fixations as shown in Figure 5, can be applied at any of the evaluation stages to evaluate the software in terms of general usability criteria, e.g., attractiveness, operability, learnability, and understandability. Of course eye tracking may not by itself fully validate these criteria, however, depending on the task, it may provide good quantitative data supporting the criteria in certain cases. Other techniques, e.g., Talk Aloud Protocol, may be used as alternative techniques, or in concert with eye tracking, or in combination with other methods.

It should also be pointed out that application of the UELC to the software development process is flexible and adaptable. As shown in Figure 4, depending on the constraints imposed on testing (e.g., limited resources, equipment, time, manpower), certain tasks can be omitted in the UELC. For example, smaller projects might not require the complete iterative cycles presented in levels I and II of the UELC. Following the dotted paths in the diagram, under certain circumstances, it may be possible to jump from Conceptual Model Design to Screen Design Standards without creating CM mockups or performing iterative CM or SDS evaluation.

Furthermore, for already completed projects, it may be possible to apply a subset of the UELC tasks by, for example, obtaining user feedback through usability testing as suggested in the Installation stage of the UELC.

Eye Movement indicators provide insight into system failures or anomalies that are attributed to operational causes rather than to hardware or software discrepancies. Eye movements provide a measure of the operational process (vs. operational performance). Analysis of these indicators identifies where system usability could be improved. Examination of related errors often identifies corrective actions to reduce or eliminate the errors. For example, a recurring operator error may be caused by an error in an operations manual, difficulty reading a display, or inadequate training, all of which can be fixed after the problem is identified.

Eye movements should be analyzed to determine if a specific type of interface component is too complex or a specific function is not being adequately taught. For example, scanpath, hot spot, or visual area coverage plots may help characterize the user interface components in terms of its "visual complexity", potentially posing usability problems.

Eye movement recording is dependent on the particular software product under investigation. This is due to the variability of the product's display presentation (stimulus) and manner of interaction. Examples may be graphical displays of maps or static simulations of user interface mock-ups (static images in both cases), web pages, or arbitrary desktop applications. Several COTS applications are available to record eye movements over most forms of stimulus displays, however, the granularity and control of eye movement data will depend on the synchronization available between the eye tracking software (server to be precise) and the application under investigation.

Visual Attention is an indicator of a user's distribution of visual attention over a user interface. This indicator, in combination with other usability indicators can help corroborate and explain a user's satisfaction (or dissatisfaction) with operation of a user interface. For example, combined with verbal comments made by users during a Talk Aloud protocol usability evaluation session, scanpaths can resolve deictic references made by the user (e.g., "I'm looking at this but I can't figure out what it's supposed to do.").

UI Visual Layout is an eye movement-related indicator that can be used to supplement other indicators of user effectiveness. The UI Visual Layout indicator, as generated by aggregate eye movement representations (visual area coverage) may show which UI components were seen or missed by the user(s). Using graphical representations such as scanpaths, hot spots, or area coverage provides insight into what the user was attending to at any particular moment in time. It may thus be possible to infer portions of the UI that are frequently attended to, or conversely, unattended (and hence underutilized). Furthermore, comparison of this indicator between users (between subjects experimental design) may offer insights into commonalities or discrepancies between different user groups (e.g., experts vs. novices).

Cognitive Load is an eye movement indicator that may facilitate estimation of the user's efficiency during operation of a software product. Quantitative data provided along with scanpaths, hot spots, or area coverage (e.g., number of fixations, fixation durations, direction of fixations, etc.), if available, can lead to statistical estimates of efficiency of the user. For example, significant dwell time over specific UI components may suggest difficulty in understandability of the component. Similarly, exceedingly long fixations over such components may indicate problems for operability and learnability. Comparison of eye movements between users (between subjects experimental design) may provide more powerful statistics in identification of such visually and hence mentally demanding components.

7. EYE MOVEMENTS: ANALYSIS GUIDANCE AND EXAMPLE

As an example of eye movement analysis, a user task under investigation was performed while operating the Marine Air-Ground Task Force (MAGTF) Intelligence Collections Applications (MICA) Command and Control Personal Computer (C2PC) software component, following the Software Acceptance Testing specifications. Specifically, in this case SAT1-033 "Add Named Areas of Interest (NAIs) to a Requirement" was performed.

Usability metrics for this task were selected from the Testing and Inquiry Methods specified in the SSC-C Software Development Process Manual [SSC-C DSWDPROC-MAN-1.2]. These metrics comprised a subset of measures used to describe the SEE (Satisfaction, Effectiveness, Efficiency) usability attributes of the software system.

7.1 Testing methods

Performance measurement required obtaining the following categorized measures, selected from the PSM [PSM, Version 4.0b]:

- User Effectiveness (e.g., accuracy)
    - User Interface (UI) Visual Layout (Aggregate Eye Movements: Spatial Distribution of Fixations)
- User Efficiency (e.g., speed)
    - Cognitive Load (Eye Movements)

7.2 Inquiry methods

The chosen inquiry method, based on a usability questionnaire, prompted the user by asking direct questions about user's satisfaction with the system:

- User Satisfaction

Because a formal questionnaire was not administered in the pilot evaluation, only anecdotal observations can be made based on a loose *talk-aloud* testing protocol.

7.3 Analysis

UE metrics observed during software operation are related and in some cases dependent on each other. In this case, usability evaluation of the software product was prompted by verbal remarks made by the subject during test.

7.3.1 User Satisfaction

The user indicated that considerable time was spent on changing the application's default colors for identification of selected NAI regions (e.g., boxes, rectangles placed on the map displayed by C2PC).

Although informal, the *frequency of discretionary use* indicator of user satisfaction appeared to be too high leading to dissatisfaction with this subtask. In effect, user impressions suggested that performance of this particular color-changing task interfered with the task's main objective (NAI identification).

7.3.2 User Effectiveness

The UI Visual Layout indicator suggests that a disproportionately large number of fixations fell on the NAI property dialog box and related color palette dialog boxes during the course of the task (see Figures 6 and 7).

7.3.3 User Efficiency

Cognitive load, as indicated by the proportion of fixations on specific user interface features (in this case the property box and color palette) suggest that a significant amount of time was devoted to these features (see Figure 8). Because in this case this was not a subtask directly related to the task objective, these particular interface features can be seen as distracting to the task at hand.

It must be emphasized that the data in this instance (number of fixations atop the property box and color palette) is confounded by the presence of fixations atop interface regions below the dialog box at times when the dialog box was not visible.

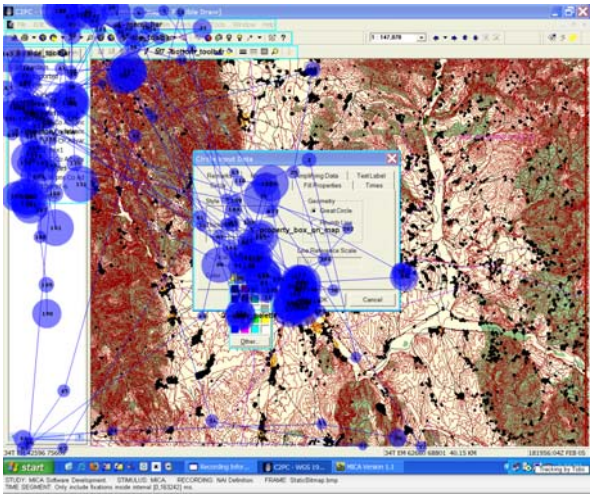

Figure 6: Scanpaths indicating sequentially fixated elements of the interface. Circles of a larger diameter indicate relatively longer dwell times. In this instance, telltale eye movement patterns can be seen over the directory view and relatively long fixations are evident over the color property dialog box and color palette.

Figure 7: Hot spots with analyst-selected Areas of Interest (AoIs) indicate aggregate fixations over the interface. In this case, the property box and color palette appear to be viewed heavily by the user. NOTE: because these are aggregate views of fixations over a dynamically changing interface, it must be noted that the diagrams show fixations atop "legitimate" regions of the interface, i.e., the map surface below the property box. Due to the program's dynamic nature, it must be understood that the property box was only in view for brief periods of time.

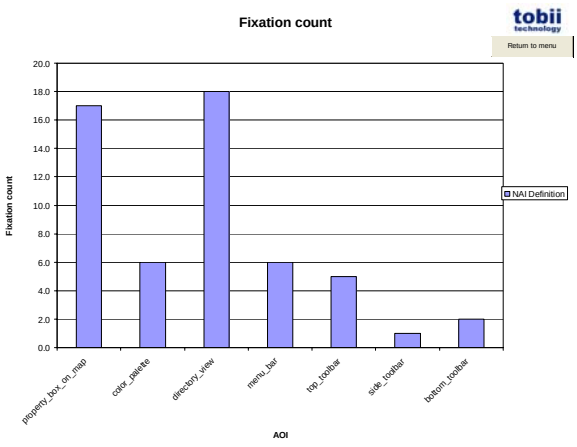

Figure 8: Fixation count comparison over selected AOIs.

Recommendations from this preliminary evaluation pilot test suggest that the color of NAI regions (e.g., boxes, rectangles, etc. used to represent them) should either be the appropriate color (red in this case) by default, or should be set via a suitable "Preferences" menu item by the user a priori. That is, there should either be a preference setting for NAI region colors, and/or the Software Acceptance Test criteria should include this setting.

Note that the above example is extremely limited in scope. Anecdotal observations are provided from a short test case performed by a single user. No comparisons between users were made and thus the above recommendations are rather crude, at best. However, the above example is illustrative of the quantitative measurement potential of eye tracking methodology.

8. CONCLUSION

Collecting data on usability attributes such as satisfaction, effectiveness, and efficiency, augments the software development process by considering Human Factors during an iterative design cycle. Usability Engineering (UE) metrics generally consist of performance and process measures related to the user. For example, error timing may correspond to the installation of new equipment or to changes in the operational procedures. Appropriate uses of an eye tracker may aid in the collection of general UE performance metrics as well as leading to insights about the user's cognitive processes during software use and evaluation. In general, however, eye movement indicators alone cannot provide a comprehensive analysis of usability.

9. REFERENCES

Government Documents

- *SSC-C Software Development Process Manual*
- *SSC-C Measurement and Analysis Process Manual*
- *SSC-C Process and Product Quality Assurance Process Manual*
- *SSC-C Requirements Development Process Manual*
- *SSC-C Validation Process Manual*
- *SSC-C Verification Process Manual*
- CJCSI 3170.01B, *Requirements Generation System*, Chairman of the Joint Chiefs of Staff
- *SSC-C (CODE 60) Business Framework*

Non-Government Documents

- Avouris, N. M., "An Introduction to Software Usability", *Proc. 8th Panhellenic Conference on Informatics*, Nicosia, November 2002. URL:
- CMMI® for Systems Engineering and Software Engineering (CMMI®-SE/SW,

Version 1.1), Staged Representation, January 11, 2002, Software Engineering Institute, CMU/SEI-2002-TR-002

- Mayhew, Deborah J., *The Usability Engineering Lifecycle: A Practitioner's Handbook for User Interface Design*, Academic Press, 1999.

- Nielsen, J., *Usability Engineering*, Academic Press, London, 1993.

- ISO 9001:2000, Quality systems - Models for quality assurance in design, development, production, installation and servicing

- Shebalin, Paul V., "Software Development Standards and the DoD Program Manager," Defense Acquisition University, *Acquisition Review Quarterly*, Summer 1994.

10.BIOGRAPHY

Michael Shumberger, LCDR USN (ret)

Mr. Shumberger is a Senior Systems Engineer with Eagan McAllister & Associates Inc. and provides direct support for SPAWARSYSCEN, Charleston (SSCC). He has over 28 years comprehensive DoD experience, with over nine (9) years experience in systems engineering in support of SPAWAR programs. He received his M.Sc. Electrical Engineering from the Naval Postgraduate School, a B.Sc. Chemistry and B.A. Business Administration from the University of Puget Sound. Additional training & certifications include: Department of Defense Acquisition Workforce Professional Program Management Level II & Systems Planning, Research, Development and Engineering Level II, and Naval Engineering Duty Officer. Current areas of research and engineering include Human Factors Engineering (HFE), Capability/Cost analysis for various Branches of the Chief of Naval Operations, and modeling and simulation architecture.

Andrew Duchowski, PhD

Dr. Duchowski is an associate professor of Computer Science at Clemson University. He received his B.Sc. ('90) and Ph.D. ('97) degrees in Computer Science from Simon Fraser University, Burnaby, Canada, and Texas A&M University, College Station, TX, respectively. His research and teaching interests include visual attention and perception, eye movements, computer vision, graphics, and virtual environments. He joined the Computer Science faculty at Clemson in January, 1998 and is currently investigating gaze-contingent perceptual graphics and collaborative virtual reality systems.

Mr. Kevin Charlow

Mr. Charlow is the Enterprise Technologies Manager in the Enterprise Systems Solutions Division at Space and Naval Warfare Systems Center (SPAWARSYSCEN), Charleston, SC. He has a B.Sc. in Computer Engineering degree from Clemson University and an MBA from Webster University. He provides program management and engineering support to the Navy Modeling and Simulation Office (NMSO) and the Office of Naval Research (ONR) in areas of Verification, Validation and Accreditation (VV&A); Live, Virtual and Constructive M&S to support training and experimentation; M&S standards and tools; and human factors engineering.