User Interface Design for Situation-aware Decision Support Systems

Nuka Nwiabu, Ian Allison, Patrik Holt, Peter Lowit, Babs Oyeneyin
School of Computing, IDEAS Research Institute, Robert Gordon University, Aberdeen, UK

***Abstract*—Information recall about general situations incurs memory and cognitive loads on operators. Recognition of information for specific situations identified with users' context and the state of the world is helpful to operators in performing tasks in complex environments. The emergence of ubiquitous, ambient, and pervasive technologies is increasingly providing methods to help operators to perform their tasks in smart and intelligent ways. Existing user interface design does not solve the problem of drawing together the information required for situation-aware decision support systems in a way that minimises cognitive load. This paper discusses a framework for user interface design of situation-aware systems that exploit inputs from users and the environment to provide information tailored to the user's tasks in specific situations. The user interface can reconfigure automatically in order to adapt to the current situation. The adaptation of the user interface to the current situation and the presentation of a reusable sequence of tasks in the situation reduces memory loads on operators. Hierarchical Task Analysis (HTA) is used to describe tasks for various types of situations. HTA is supplemented with scenarios to stimulate design ideas and requirements analysis is used to represent interrelationships between tasks.**

Keywords: **Situation awareness; Context awareness; User interface design; Cognition; Scenarios; Task Analysis; Requirements analysis.**

I. Introduction

User interfaces (UIs) represent the point of contact between systems and human users. The emergence of ubiquitous, ambient, and pervasive technologies has resulted in methods to assist users in smart and intelligent ways. One such way is context-aware computing; a trend whereby computing devices and systems serve their users beyond the traditional desktop in diverse environments [6]. Dey [10] defines context as any information that can be used to characterize the situation of an entity. A system is said to be context aware if it uses context to provide relevant information and services to the user [29]. Context awareness was introduced by Schilit [32] to develop an application that adapts to the location of use, nearby people and objects, and the change of those objects over time. With technology advancement and the growth of mobile computing in recent times, context awareness has attracted greater research attention [16]. Context-aware user interfaces allows systems to dynamically adapt to changes in a user's task domain by updating relevant information and service provision.

A related concept to context awareness is the notion of situation awareness. Situation awareness (SA) is a cognitive process in decision-making and is defined as "the perception of elements in the environment within a volume of time and space, the comprehension of their meaning, and the projection of their status in the near future" [14]. The Endsley SA model [14] has three layers comprising perception, comprehension, and projection. The perception layer recognises all the necessary information about the environment. The comprehension layer interprets the perceived information in order to understand the current state of the environment. The projection layer uses knowledge of the current state of the environment to predict its future state. Situation awareness and context awareness both focuses on information about the state of the environment in which these tasks are carried out [16]. Situation-aware systems exploit explicit and implicit inputs to provide information tailored to users' tasks in different situations. The system can adjust to a range of user abilities to solve the problem of variations in user's expertise, greater speed of performance, reduced operators workload, more consistency, greater flexibility in behaviours, and less training time [23]. But it is simplistic to assume that adaptive user modelling will solve all human-computer interaction problem. A growing body of research has examined the characteristics of human-operator interaction with adaptive display and described the human performance costs such as trust, complacency, skill and performance degradation, decrease user acceptance that can occur in such interaction [26],[31]. Designers of UIs for situation-aware systems must know what changes from users or environments are related to the tasks that the users perform to achieve goals by drawing up a task model, using a notation which allows it to describe tasks for various types of situations [9]. There appears to be no existing framework with a notation to support designers in building UIs for situation-aware systems from situation-based tasks.

This paper describes a framework for the design of situation-aware interfaces in a manner that input information (context and environmental cues) can be explicitly taken into account in the task specification. In order to achieve a concrete user interface (UI), it is assumed that the designer adds abstract UI components to the task model. This information is platform-independent so that the rendering back-end can ultimately use this information to construct a concrete UI for various platforms. The next step consists of creating the dialogue model. Designers can be supported by automatically generating the statuses and transitions between the various individual dialogues, so as to simplify the work of designers. The tool includes an algorithm to calculate the different dialogues and transitions between dialogues from the task specification. Designers can adjust, add or remove these transitions

according to the results of a previous testing stage or the designers' experience. This way situation-aware UI designers can manipulate transitions that would be triggered by situation changes. Designers thus have control over the influence of situation on the usability of the UIs.

The case study for the paper is the design of situation-aware UI for Hydrate formation prediction in subsea oil and gas pipelines. There will three transition statuses, Normal, Warning, and Danger [25]. Normal situations represent situations where there is no problem in the domain. A warning situation represents a situation that is not normal but not yet in danger. A danger situation is a crisis situation that means there are already problems in the domain. The UI executes reconfiguration after input variation so as to stay adapted to any of these situations that depict the current situation in the domain. Warning situations cause the presentation of preventive sequence of task while danger situations cause the presentation of remediation or repair sequences of task. Hierarchical Task Analysis (HTA) is used to describe tasks for these situations. HTA is supplemented with scenarios to stimulate design ideas. Each scenario has a setting that explicitly describes the starting state of the current and the future situations, and implicitly depicts the characters that take part in the situations in the scenario. Each scenario has actors who perform tasks to achieve goals in different situations. Requirements analysis is used to supplement our scenario-based HTA in representing interrelationships between tasks. Dialogues and transitions between dialogues are calculated from the task specifications.

The remainder of the paper is as follows. The following section provides a short overview of related work. Then the design process, and the task model to the approach are successively presented followed by a prototype architectural design for situation-aware UI. Finally, the design is evaluated and the work is summarised and concluded.

II. Related Work

The emergence of ubiquitous, ambient, and pervasive technologies has triggered research in context-aware UI design. Limbourg et al.[21] developed a language, UsiXML, to describe context-aware UIs. He provided tool support, however, concentrates on transformations between models in order to transform abstract descriptions to concrete ones, with no recognition of the fact that there could be unexpected changes of the UI when a context change occurs. Clerckx and Coninx [7] provided a mechanism to avoid these unexpected changes by incorporating context in UI development using transformations between models [8] but the integration with the context model is done by the designer. Mori et al.[24] describes the TERESA tool for designing UIs for mobile devices. Abstract models are used in order to deploy concrete UIs on several platforms. The approach is task centered implying that a lot of effort has been taken in visualizing the task model. A reconsideration of visual representation of task models is recently carried out by [27]. Techniques like semantic zoom (hiding information outside the point of focus) and fish eye views (increasing the size of elements in focus) are introduced in order to improve the effectiveness of viewing and constructing task models.

To express the solution for identified UI patterns in an abstract way, [3] provided a modelling tool for model-based UI design having two different levels of abstraction; wisdom presentation model, and canonical abstract prototypes. The tool applies the Wisdom model to UI patterns, easily expressing containment relationship, while the Canonical prototype is much closer to the concrete representation of the identified pattern. However, support for context-aware and multi-device UIs using the Canonical notation is not obvious and is therefore not considered by the approach.

Calvary et al. [2] describe a development process to create context-sensitive UIs. The development process consists of four steps: creation of a task-oriented specification, creation of the abstract interface, creation of the concrete interface, and finally the creation of the context-sensitive interactive system. The focus however, is on a mechanism for context detection and how context information can be used to adapt the UI, captured in three stages; recognizing the current situation, calculating the reaction, and executing the reaction.

Wu et al. [35] used HTA combined with scenario-based design to develop a UI to context-aware indoor navigation applications. The approach used HTA method to identify user, user-application, and application tasks. The work provided a framework of command interfaces for executing interaction between application tasks and user tasks. These command interfaces link users, user-application, and application tasks. The work did not look at how human variability influences usability. Also, no mention was made of the method of interaction between objects.

In a similar hybrid approach, Lewis [20] combined HTA with requirement analysis by replacing the abstract, and partial task elements of requirement analysis with real tasks from the task analysis. The approach does not however consider the possibilities of losing detail in the process of generalisation. Kim et al [18] and Liu [22] combined metadata definition with scenarios to build task knowledge structures in their works on sentence ends and interruption points in speech. Metadata was created within a specific context and for specific purpose, and different purposes and different contexts have different metadata requirements. Metadata are information and documentation associated with objects which makes data understandable and shareable for users over time relieving them of having to have full advance knowledge of the data existence or characteristics.

III. Designing Situation-Aware interfaces

This section provides an overview of the design process (Fig. 1.). The design process supports the design of declarative abstract models, describing the situation-aware user interface.

The aggregate of the models can be serialized in order to export these models to a runtime. To test the result of these models, the corresponding UI can be generated in the shape of a prototype to check the usability of the system.

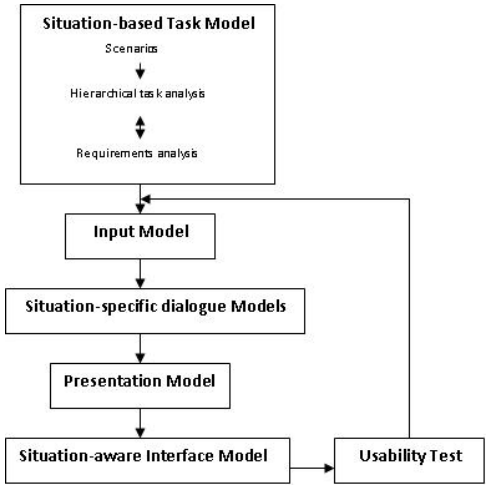

Fig. 1. Situation-aware User Interface Design Process

Considering the prototype, some changes to the models in the design process can be applied to alter for instance the presentation of the UI or how situation changes may affect the UI.

Situation-based Task Model: First, a task model is specified describing the tasks users and application may encounter when interaction with the system is taking place. Because we want to develop Situation-aware UIs, tasks also depend on the current situation. This is why tasks in the task model are drawn for specific situations. In this way the designer can describe different tasks for different situations.

Input Model: When the task model is specified, the designer has to denote what kind of input can influence the interaction, i.e. the tasks. This can be done by selecting objects for input gathering (Perception Objects or POs). These objects can be aggregated by the aggregation objects (AO) and interpreted by the interpretation objects (IO). The designer can do this by linking AOs to POs and selecting from a set of predefined interpretation rules how the input has to be interpreted. The IOs represent the interpreted information at the comprehension layer. When the input model is specified, the designer has to link the IOs to task model nodes (inter-model connection). In this way, the designer can denote which tasks can be performed in which situation.

Situation-Specific Dialogue Models: Next, the tool will automatically extract a dialogue model from the task model for each situation. Afterwards, inter-model connections are added automatically between states of the dialogue model and tasks of the task model that are enabled for each particular state. The dialogue model nodes (states) of the different dialogue models are linked to denote between which states situation changes may occur.

Presentation Model: To provide the interface model with information about how the interaction should be presented to the user, designers have to compose abstract UI components, and link these to the relevant tasks for each presentation model node. The presentation model nodes can be structured hierarchically in order to group presentation components for layout purposes. The designer can choose from several abstract UI components such as static, input, choice, navigation control, hierarchy, and custom widget. Finally the UI components can be grouped, and structured in a hierarchical structure.

Situation-aware Interface Model: The aggregate of all the models results in a situation-aware interface model.

Usability evaluations: Usability tests are then carried out to test and improve usability of the graphical interface with the models.

IV. Situation-based Task Design

The first step in the situation-aware user interface (SAUI) design process just like every other interface design is to draw up the task model, a hierarchic structure and a way of establishing temporal relationships between various (sub) tasks. Task analysis can help designers understand what needs to be accomplished by the user, the environment, and the system and break down the major task into the simplest component parts. Designers need to know what user tasks are necessary to operate the system and also need to know which part of user input can be transferred to the system task in order to increase the level of context awareness of the system. Hierarchical Task Analysis (HTA) focuses on the way a task is decomposed into subtasks and the order and conditions where these are executed. They are represented as a hierarchy of tasks, subtasks and plans. It provides a brief picture of user tasks and basic functional specification of the proposed application. The top down structure of HTA ensures completeness and is easy to comprehend [4] but cannot adequately address human factor and social issues, for example, emotion [6]. Such issues may be elicited from a scenario.

Scenarios according to Carroll [5], are examples of specific experience that exist to stimulate designers' creative imagination. Scenarios and claims are lightweight instruments that guide thought and support reasoning in the design process [5]. But scenarios also have their own downsides. According to Diaper [11] scenarios can lead to errors, as a scenario, or even a set of scenarios, do not explicitly guide a designer towards a correct model of the required system. Both scenarios and task analysis are criticised for omitting the explicit representation of communication between agents engaged in collaborative tasks and also not capturing the richness of interaction that occurs in the real world compared with other methods such as requirements analysis [19].

This paper designs a task model based on situations, using a hybrid technique of combining scenarios, HTA, and requirements analysis. Designers use the set of tasks that can be identified in the task specification as a basis for the different dialogues the user interface will need to complete its tasks.

A. Problem Scenario

We commence design with scenarios using the Robertson model (Fig. 2.) below:

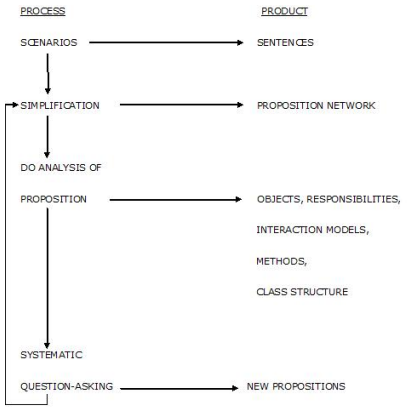

Fig. 2. Scenario-based design (Robertson, 1996)

Following the Robertson model, we used narrative texts to start a scenario in hydrate formation prediction in sub sea oil and gas pipelines, for example:

An engineer plans to use a system that senses the sea floor environment through sensors to predict the formation of hydrate in a subsea oil and gas pipeline. The system understands the situation in the pipeline by integrating the engineer's context with cues from the sea floor. The situation presented was a warning situation which consist of goal to be achieved and the corresponding tasks. Among the tasks are, reduce water dew point task, and chemical injection task. He decided to use chemical injection method to solve the problem in the absence of dehydrator. The available chemical for the engineer to use was methanol which is cheaper on a volume basis than glycol. Methanol distribute in three phases; aqueous, vapour, and liquid. At the aqueous phase the engineer used the Hammerschmidt equation to determine the methanol molecular weight and k-value before injection.

We simplify the task scenario by partitioning the scenario into propositions to identify candidate design objects as follows e.g. Engineer predicts the formation of hydrate, Engineer uses the system, system senses the sea floor environment, system integrate context and cues, "'Reduce water dew point" is a task, Methanol distributes in three phases, aqueous, vapour, and liquid. The propositions apart from helping to identify candidate objects also served as guides to object interactions e.g. showing interactions between Engineer and system. It shows the objects that are active, for example "engineer" and the ones that have been acted upon, for example "system". The propositions also show the interrelations among the objects and some basics about the properties of the objects [30], e.g. the object "molecular weight" and "k-value" defined by "Hammerschdt equation".

The proposition analysis provided useful information but it was however, not sufficiently detailed for design. Systematic question-asking was used to elaborate the propositional list [30]. Questions were asked on each item of the propositional list. The why-questions are used to receive both intentional and causal information. The how-questions provided the procedural, causal, and enablement information. Answers to the why-and-how questions exposed some of the content of the problem space and generated materials for the work.

Some why-questions were asked, for example (1) Why is knowledge of the environment required to understand the situation? (2) Why is chemical injection a method of preventing hydrate formation? (3) Why did the Engineer not use line heating method? Some how-questions were also asked, for example (1) How are context and cues integrated? (2) How do system retrieve past situation? (3) How are past situations preserved?

Answers to the why-questions revealed some important information that was not explicit in the scenarios. For example, an answer to a why-question "Why is knowledge of the environment required to understand the situation?" reveals the fact that the dynamic state of the environment affects the state of the gas and its flow in the pipelines. This information is not explicit in the scenario but helped in identifying other candidate objects. The answer to the how-question on "How are context and cues integrated?" gave the understanding that there must be perception before integration. This answer provided us a new candidate object, "perception". Similarly, the answer to the question "How do systems retrieve past situations?" gave birth to a new candidate object, "assess similarity".

B. Hierarchical Task Analysis

Hierarchical Task Analysis (HTA) techniques was used to decompose complex tasks identified in scenario design into subtasks and the order and conditions where these are executed. The output of HTA is represented diagramatically and textually [12]. HTA provides a brief picture of user tasks and basic functional specification of the proposed system. The break down of tasks enables us to stay focused on parts of the overall task without missing the picture of overall task activities. The top down structure ensures completeness and is easy to understand [33][4]. Also, in the task-design mapping, HTA provides a good description of all task functions for mapping on to the system [4].

User:
0. predict the formation of hydrate
- *1. use the system (User task)*
- *2. providing the context (User task)*
  - *2.1. provide well head pressure*
  - *2.2. provide well head temperature*
  - *2.3. provide flow rate*
  - *2.4. provide location of pipeline*
  - *2.5. provide time*
- *3. provide state of the environment (Sensors task)*
  - *3.1. provide solar radiation*

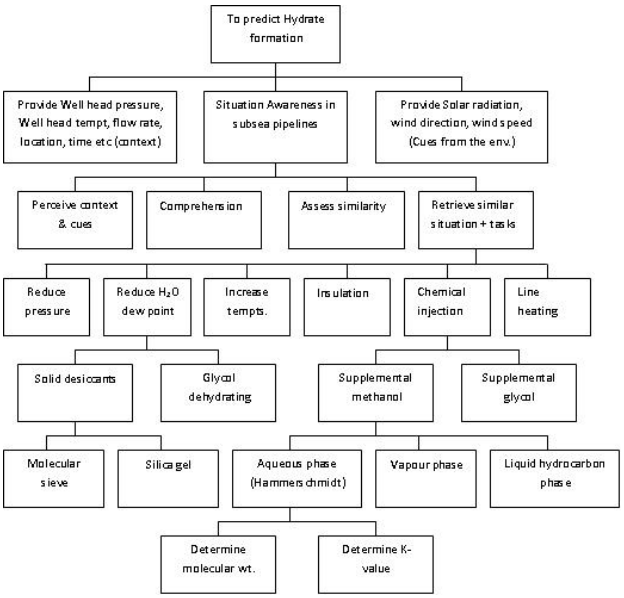

Fig. 3. Situation-based Task Model

3.2. provide wind direction
3.3. provide wind speed
4. situation awareness (Application task)
4.1. perceive context and cues
4.2. integrating context and cues
4.3. understanding the situation
4.4. assess similarity
4.5. retrieve similar past cases and tasks
5. perform "reduce water dew point" task (User task)
5.1. glycol dehydrating
5.2. solid desiccants
5.2.1. mollecular sieve
5.2.2. silica gel
6. perform "chemical injection" task (User task)
6.1. supplemental glycol
6.2. supplemental methanol
6.2.1. liquid hydrocarbon phase
6.2.2. vapour phase
6.2.3. HMS equation (User-Application task)
6.2.3.1. molecular weight (User-Application task)
6.2.3.2. determine k-value (User-Application)
7. preserve workable method (User-Application task)

C. Integrating HTA and Requirements analysis

Representative tasks from HTA are mapped into the abstract model of requirements analysis to supplement requirements analysis using Use Cases of the Unified Modelling Language (UML). The "human'user", "sensors" and the "application" are actors in the Use Case diagram. Actors "'represent the roles that people, other systems or devices take on when communicating with the particular Use Cases in the system"[1]. Use Cases are the different tasks performed by the human user, and the application [28].

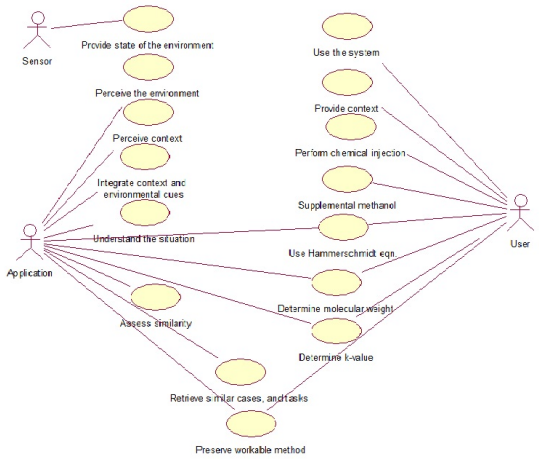

Fig. 4. Use case modelling

From the Use Case model, sensor provides the state of the environment. The application perceives the environment and user's context, integrates the context and cues from the environment, understands the situation in the domain, matches the present situation with past situations to assess similarity, and then retrieves a similar past situations together with the sequence of tasks that were performed to address it. The user interacts with the system, provides the context, performs chemical injection, applies supplemental methanol.The user and the application use hammerschmidt equation at the aqueous phase, determine the molecular weight, determine the k-value and finally preserve workable solutions.

Each of the actors has a number of Use Cases but some of the Use Cases depend on other Use Cases, for example, the application's Use Case, "understand the situation" depends on the Use Case, "integrate context and cues". Similarly, user's Use Case "apply methanol" is dependent on "perform chemical injection".

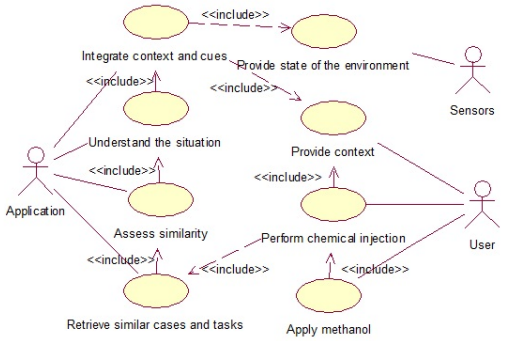

Fig. 5. Use case relationships and interactions

In the way, the Use Case of one Actor can depend on the Use cases of other actors. For example, the application's "integrates context and cues" which is the inclusion Use Case to "understand the situation", is dependent on the user's

"'provide context" Use Case and the sensor's "'provide state of the environment" Use Case. The dependencies of the Use Cases shows the interrelationships between the user's tasks and the application's tasks.

V. Situation-Aware User Interface Prototyping

The architecture for Situation-aware user interface design comprises the user interface component, situation awareness model, and the environmental sensing systems. The user provides the system with context, and receives decision support from the system through the user interface. The system senses the environment through some sensing systems or sensors. The SA component is the application core. The SA model is the first (perception) and second (comprehension) layers of the Endsley SA model [14]. The system perceives cues from the environment to understand the current situation. Additionally, the system accepts or ignores the user's context. Ignoring context gives a static interface which SA will be the same on each retrieval, although different actions could be explored by the user with different information. Taking the user's context into account gives an adaptive display whose behaviour will be customised to the user's specific need. In the adaptive mode at constant state of the environment, the system prompts the operator for context. Context input results to an automatic change in SA and action list. The system design is lower than the rigid system in level of automation (LOA) [13]. LOA is a level of task planning and performance interaction between an operator and the computer in a complex system with four systematic functions; monitoring, generating, selecting, and implementing [17]. Monitoring is the perception task of our SA computational model. Generating is the task of comprehension while selecting and implementing are human decision making and action performance tasks respectively. The system presents SA and a set of actions to be performed in the situations to the operator. The operator cannot generate any other option but to select the option provided by the computer to perform decision making and physically implementing the actions.

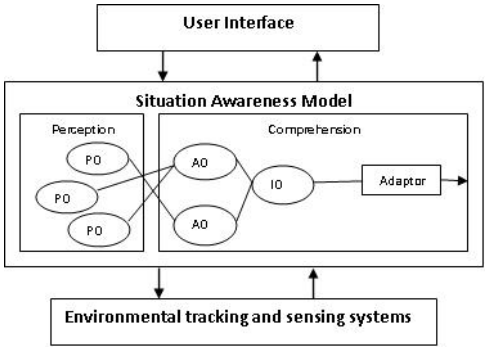

Fig. 6. Situation-Aware User Interface Design Model

A. Input Model

Input acquisition takes place at perception layer. Input comes to the application core from users' context, and sensors. These two types of inputs forms the perception object (PO). Three levels of processing take place at the comprehension layer; input aggregation, input interpretation, and adaptation, represented as aggregation object (AO), interpretation object (IO), and adaptor respectively. The interpretation object (IO) ensures that mapping takes place from POs to AOs each time new services become available or when services disappear. The AOs then indicate to the IO the categories of POs from which they can use input information. The IO carries out the tasks of (1) Recalculating the mapping of AOs on POs: a service can be a supplier of input information. If this is the case, the IO can make use of this and treat the service as a perception object and link to comprehension which can make use of this input information (2) Detecting input changes: if a context and environmental change takes place, the IO will look at the adaptor in order to decide whether the change has a direct influence, so that an interdialogue transition has to be implemented (3) Invoking an interdialogue transition: the IO sends an event to the adaptor and tells it that an input change has taken place and that the interdialogue transition has to be implemented if it is evident from this interpreted information that a situation change has taken place. If a transition exist in the dialogue model to follow up this situation change, the adaptor will invoke the appropriate transition.

B. Situation-specific dialogue

A separate dialogue model will be calculated automatically for these different types of situations and presented to the designer. The designer can then indicate between which statuses transitions are possible under the influence of situation changes. For example, in a three status situation, Normal, Warning, Danger, the designer can decide, only to make a transition from Warning to Danger when the user interface is in the main menu status. This avoids the user interface adjusting if this is not desirable. IOs are linked to these transitions to make it clear what has to be taken into account in order to make the transition. An example of an IO is the Warning object. This object can indicate if the situation is in the Warning state, using POs and AOs. The adaptor changes the state of the UI caused by a change in context and the environment. The tool provides a design technique that can carry out prediction of possible changes in the UI following termination of a task, the implementation of a user action or a situation change. The design tool generates a prototype UI which it derives from the tasks in the task specification. The specific presentation of the gathering of tasks is generated from a device-independent XML-based UI description which the designer can attach to the tasks.

C. Production of a running User Interface

During the application runtime, the adaptor controls communication between UI abstract input information and the application core. The adaptor possesses information about the user's tasks and how these can be influenced by the situation. The IO encapsulates input information at such an abstract level that it only tells the adaptor that the situation change that

has taken place is significant enough to adjust the status of the UI. The adaptor uses the dynamic dialogue model and the dynamic task model to decide when the UI has to be updated. These dynamic models are adjusted so that account can be taken of the current situation, if this influences the tasks the user wants to perform. The dynamic dialogue model consists of possible statuses of the UI. The difference is in the transitions that can occur. Here, we make a distinction between intra-dialogue and inter-dialogue transitions. An intra-dialogue transition is a transition between two states which is performed if the task described for the transition is performed by the user or the application. An inter-dialogue transition, by contrast, is a transition between two possible states of the UI, but can only be performed if a situation change has taken place which fulfills the conditions defined by the designer for the transition.

From the time the application is launched, the status of the UI and the application can be changed by the user, the application and the IO. The IO detects the current situation, supplied by the abstract interaction objects and then the adaptor is notified of the status in which the UI will have to be launched.

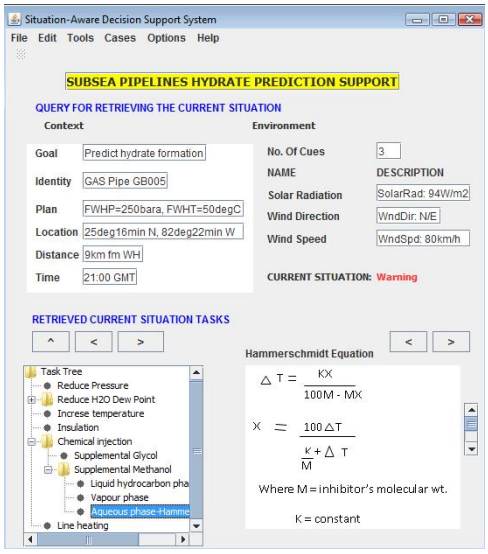

Fig. 7. A screen shot of Situation-Aware User Interface

VI. Design Evaluation

The design usability evaluation was carried out with end users. This procedure involves administering questionnaire forms consisting of 50 areas of satisfaction to 20 end users from an oil and gas organisation. The respondents had to decide whether they agreed, undecided, or disagree with each of the 50 items in relation to the UI they were evaluating. The respondents completed the inventory at their work place. These respondents who were end users used the system to accomplish task goals within the organisation for their daily work. The resulting matrix of inter-correlations between items was factor analysed using the Software Usability Measurement Inventory (SUMI). The items were observed to relate to a number of different meaningful areas of user perception of usability. The program (SUMISCO) analysed and transformed the data into Global and five other subscales namely: Efficiency, Affect, Helpfulness, Control, and Learnability.

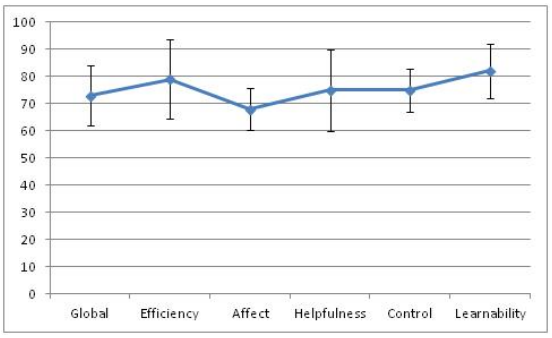

Fig. 8. Profile Analysis

Efficiency measures the degree to which users feel that the UI assists them in their work and is related to the concept of transparency. The Affect subscale measures the user's general emotional reaction to the UI or Likability. Helpfulness measures the degree to which the UI is self-explanatory, as well as more specific things like the adequacy of help facilities and documentation. The Control subscale measures the extent to which the user feels in control of the system, as opposed to being controlled by the system, when carrying out the task. Finally, Learnability, measures the speed and facility with which the user feels that they have been able to master the UI, or to learn how to use new features when necessary.

Figure 8 shows the median rating for each usability scale, with upper and lower confidence intervals. Feedback from users on efficiency rating indicated that users were satisfied with the UI adaptation to current situations and the presentation of reusable sequence of tasks for hydrate prevention especially chemical injection tasks and procedures. Users were also satisfied with less commands required for a given task performance, reduced number of clicks or keystrokes required to carry out tasks, and less options on the screen at one time. However, users were of the opinion that the HELP facility should have more information than already provided. The comments on the HELP facility resulted to the "Affect" low rating of 68%. The generally high usability rating of the UI implies low cognitive load on users. Instead of having to make extra effort to understand the UI, a user only need to be focus on task performance. All the comments have been noted and the response will be reflected in further work.

VII. Conclusion and Further Work

The core of the study is that the addition of context to input in situation-aware systems results to automatic change in SA and action list, making the UI to adapt to the specific need of the individual operator. Also, that adaptation will take place only when the change in context and the environment is significant enough to result to transition between two possible statuses of the user interface.

The adaptation of the interface to the current situation as specified in this prototype and the presentation of reusable tasks in the situation with reduced number of commands, clicks, and options reduces cognitive loads on operators and thereby facilitates interactions.

The work has also demonstrated a method of combining scenarios, HTA, and requirements analysis in task modelling. The approaches complement each other by using scenarios to stimulate and support reasoning in task analysis. Task analysis provides an integrated picture of tasks. Mapping real, complete and representative tasks of HTA to abstract and partial tasks of requirements analysis helps to ensure that all important users' tasks with their relationships and interactions are identified.

Further work will be on the unification of SA and ecological theories. Both concepts emphasise on human cognition and behaviour in real world settings. Although work in ecology is trying to shift from cognitive constructs to the environment while SA is broadening the study of the environment and cognitive theories [15]. Ecological Interface Design (EID) probably due to its origin from the engineering plant domain [34] is mostly applied to specific subsystem displays. Situation-aware interface design is an SA-Oriented Design (SAOD) that adapt to the current situation with emphasis on the whole system. The study in Situation-Aware Ecological Interface Design (SAEID) will provide a framework for interface design that will attempt to support operators in dealing with unfamiliar and unanticipated abnormal situations in complex domains. There will be a comparative performance and usability evaluation of the static and adaptive displays. The effectiveness of the user interface in terms of the reduction of the cognitive load of the operators shall be estimated using cognitive load reduction (CLR) index.

REFERENCES

[1] S. Bennett, S. McRobb, and R. Farmer. *Object-oriented system analysis and design using UML*. McGRAW-HILL education, 2006.

[2] G. Calvary, J. Coutaz, and D. Thevenin. Supporting context changes for plastic user interfaces: A process and a mechanism. In *In Joint Proceedings of HCI and IHM 2001. Lille, France*, page 349364, 2001.

[3] P.F. Campos and N.J. Nunes. Canonsketch: a user-centered tool for canonical abstract prototyping. In *Proceedings of EHCI and DSV-IS, Germany 2004*, pages 108–126, 2004.

[4] M. S. Carey, R. B. Stammers, and J. A. Astley. *In Task Analysis for Human-Computer Interaction*. Ellis Horwood, 1989.

[5] J.M. Carroll. Making use is more than a matter of task analysis. *Interacting with Computers*, 14:619–627, 2002.

[6] Y.M. Cheng and C. Johnson. Applying task analysis to facilitate the design of context-aware technologies. In *2nd Workshop on complexity in design and engineering*, 2007.

[7] T. Clerckx and K. Coninx. Towards an integrated development environment for context-aware user interfaces. In *Dagstuhl seminar. Mobil computing and ambient intelligent: The challenge for multimedia*, 2005.

[8] T. Clerckx, K. Luyten, and K. Coninx. The mapping problem back and forth: Customizing dynamic models while preserving consistency. In *3rd International Workshop on Task Models and Diagrams for user interface design (TAMODIA 2004), Prague, Czech Republic*, pages 33–42, 2004.

[9] T. Clerckx, K. Luyten, and K. Coninx. Designing interactive systems in context: From prototype to deployment. In *19th British HCI Group Annual Conference (HCI 2005), Napier University, Edinburgh, United Kingdom*, 2005.

[10] A.K. Dey. Understanding and using context. *Personal and Ubiquitous Computing*, 5:5–7, 2001.

[11] D. Diaper. Task scenarios and thought. *Interacting with Computers*, 14:629–638, 2002.

[12] A. Dix, J. Finlay, G. Abowd, and R. Beale. *Human-Computer Interaction*. Prentice Hall Europe, 1998.

[13] M. Endsley and D. Kaber. Level of automation effects on performance, situation awareness and workload in a dynamic control task. *Ergonomics*, 42:462–492, 1999.

[14] M.R. Endsley. Toward a theory of situation awareness in dynamic systems. *Human Factors and Ergonomics Society*, 37:32–64, 1995.

[15] M.R. Endsley. *Situation Awareness: Progress and Directions*. Ashgate publishing, 2004.

[16] Y. Feng, T. Teng, and A. Tan. Modelling situation awareness for context-aware decision support. *Expert Systems with Applications*, 36:455–463, 2009.

[17] D.B. Kaber and M.R. Endsley. The effects of level of automation and adaptive automation on human performance, situation awareness and workload in a dynamic control task. *Theoretical Issues in Ergonomics Science*, pages 1–40, 2004.

[18] J. kim, S.E. Schwarm, and M. Ostendorf. Detecting structural metadata with decision trees and transformation-based learning. In *HLT/NAACL*, pages 137–144, 2004.

[19] K. Kuutti. *Workprocess: Scenarios as a preliminary vocabulary. In J. M. Carroll (Ed.), Scenario based design*. Wiley, 1995.

[20] C. Lewis and J. Rieman. *Task-Centered User Interface Design*. http://www.hcibib.org/tcuid, 2008.

[21] Q. Limbourg, J. Vanderdonckt, B. Michotte, L. Bouillon, and V. Lopez-Jaquero. *USIXML: a Language Supporting Multi-Path Development of User Interfaces*. Kazman and Palanque, 2004.

[22] Y. Liu, E. Shriberg, A. Stolcke, B. Peskin, J. Ang, D. Hillard, M. Ostendorf, M. Tomalin, P. Woodland, and M. Harper. Structural metadata research in the ears program. In *ICASSP*, volume 5, 2005.

[23] C.A. Miller, H. Funk, R. Goldman, J. Meisner, and P. Wu. Implications of adaptive vs. adaptable uis on decision making: Why automated adaptiveness is not always the right answer. In *In Proceedings of the 1st International Conference on Augmented Cognition, Las Vegas, NV*, 2005.

[24] G. Mori, F. Patern'o, and C. Santoro. Design and development of multidevice interfaces through multiple logical descriptions. *IEEE Transactions on Software Engineering*, 30(8), 2004.

[25] N.D. Nwiabu, I. Allison, P. Holt, P. Lowit, and B. Oyeneyin. Situation awareness in context-aware case-based decision support. In *IEEE International conference on Cognitive Methods in Situation awareness and Decision Support, Miami Beach, FL*, pages 9–16, 2011.

[26] R. Parasuraman, T. Sheridan, and C. Wickens. A model for types and levels of human interaction with automation. *IEEE Transactions on Systems, Man, and CyberneticsPart A: Systems and Humans*, 30:286–297, 2000.

[27] F. Patern'o and E. Zini. Applying information visualization techniques to visual representations of task models. In *3rd International Workshop, Tamodia04, Prague, Czech Republic, November 15-16, 2004*, pages 105–112, 2004.

[28] M. Priestley. *Practical object-oriented design with UML*. McGraw-Hill education, 2003.

[29] J. Raz, A. Juhola, J. Serrat-Fernandez, and A. Galis. *Fast and efficient context-aware services*. Number ISBN: 0-470-01668-X. John Wiley & Sons Inc., 2006.

[30] S. Robertson. *Generating object-oriented design representations via scenario queries.* John Wiley & Sons Inc., 1995.

[31] P. Satchell. *Innovation and automation*. Aldershot, UK: Ashgate, 1998.

[32] B. Schilit and M. Theimer. Disseminating active map information to mobile hosts. *IEEE Network*, 8:22–32, 1994.

[33] A. Shepherd. *In Task Analysis for Human-Computer Interaction(Ed, Daiper, D.)*. Ellis Horwood, 1989.

[34] K.J. Vicente and J. Rasmussen. Ecological interface design: Theoretical foundations. *IEEE Transactions on systems, man, and Cybernetics*, 22:589–606, 1992.

[35] H. Wu, A. Marshall, W. Yu, and Y.M. Cheng. Applying hta method to the design of context-aware indoor navigation for the visually-impaired. In *4th Intl. conference on Mobile Technologies, Applications and Systems*, 2007.