Designing with varying design parameters: The Unified Design Process

Anthony Savidis[1], Constantine Stephanidis[1,2]

[1]Institute of Computer Science (ICS)
Foundation for Research and Technology-Hellas (FORTH)
Science and Technology Park of Crete
Heraklion, Crete, GR-71110, Greece,
Tel: +30 81 391741, Fax: +30 81 391740
E-mail: {as, cs}@ics.forth.gr

[2]Department of Computer Science, University of Crete

ABSTRACT

Designing on the basis of varying design parameters means that alternative design artifacts may need to be employed for different parameter values. In such situations, the potentially broad range of different parameter values renders impractical the production of distinct design versions. Consequently, there is a need for a design process which is capable of managing diversity *within the design act itself*. This paper presents the unified design process, which addresses systematically the management of design alternatives, by providing a framework for *unifying* diverging design decisions, based on the fundamental notion of *design polymorphism*. The Unified Design Method has been recently proposed and applied in the field of Human-Computer Interaction (HCI) for the purpose of achieving universal access, and in particular for the design of interfaces capable of self-adapting to individual users and usage-contexts. The discussion in this paper will be based on the findings of recent research and development work, while some important issues for future research will be also addressed.

Keywords

Design methods, universal access, self-adapting interface, unified interface, hierarchical task analysis.

INTRODUCTION

The design of interactive applications for potentially *anyone*, *anywhere* is based on a broad range of differentiating user parameters, such as age, computer knowledge and skills, purpose of use, interaction preferences, abilities, etc (Stephanidis, 2001). Hence, to achieve accessible and high-quality interaction, the designed dialogue patterns will have to maximally fit individual user-, and usage-context- attribute values.

Consequently, it is likely that alternative dialogue patterns, even for the same tasks / sub-tasks, will populate the resulting design space and as a result, instead of a "single" design, the outcome of such a design process will be a design space populated with well-documented sets of alternative dialogue artifacts for each user task / sub-task / activity. Alternative artifacts within each set correspond to the different instantiations of the design parameters (i.e., user- and usage-context- parameters).

FUNDAMENTALS OF UNIFIED DESIGN

The unified design method drives a "*divide & conquer*" design process, in which design problems are incrementally analyzed into sub-problems, producing a hierarchical decomposition, characterized by two key properties:

- Given a particular design problem, its hierarchical analysis may be driven in alternative manners, reflecting differing parameter values. This capability of design differentiation, according to varying design parameters introduces the notion of *polymorphism* in hierarchical design decomposition.
- In the context of polymorphic hierarchical design decomposition, design artifacts are associated to their respective design (sub-) problems, providing specific solutions documented with the particular snapshot of the design parameters they address. This feature constitutes a type of *embedded documentation* within the design artifacts.

UNIFIED DESIGN IN THE HCI CONTEXT

In the context of HCI, the unified design method is instantiated as a polymorphic hierarchical user-task analysis (Johnson et al., 1988). Polymorphic task hierarchies combine the fundamentals of: (a) hierarchical *task analysis*; (b) design *polymorphism*; and (c) user-task oriented *operators*. Polymorphism enables design differentiation at any level of the task hierarchy, on the basis of differing values of the design parameters (i.e., user- and usage-context parameters). Even though the hierarchical

decomposition may diverge at any level of the task hierarchy, alternative decompositions are always attached together, and documented on the basis of well-defined design relationships.

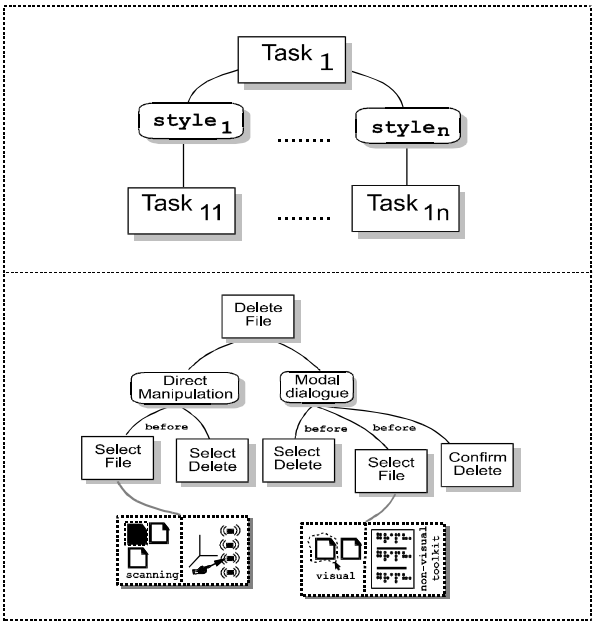

Figure 1: An Example of polymorphic task decomposition.

The concept of polymorphic task hierarchies is illustrated in Figure 4. Each alternative decomposition is called a decomposition style, or simply a style, and is given an appropriate name. The alternative sub-hierarchies are attached to their respective styles. This example shows how two alternative dialogue styles for a "delete file" task may be designed.

Categories of Polymorphic Artifacts

The unified user interface design method distinguishes three categories of design artifacts (see Table 1), all of which are subject to polymorphism on the basis of user- and usage-context- parameters. System and user tasks may be freely combined within task "formulas", defining how sequences of user-initiated actions and system-driven actions interrelate. The physical design provides the physical interaction context in which a particular user (sub-) task may be accomplished, thus annotating the task hierarchy with physical design information (see Figure 1).

Relationships among Alternative Styles

Four fundamental relationships among alternative styles (concerning the same polymorphic artifact) have been identified in the unified user interface design, reflecting the way in which artifacts may be employed during interaction for an individual user in a particular context (see Table 2).

Table 1: Categories of polymorphic artifacts.

User tasks, representing what the user has to do; they constitute the basis of polymorphic task analysis.
System tasks, representing what the system has to do, or how it responds to user actions (i.e. feedback); notationally, they are treated exactly as user tasks.
Physical design, concerning the interface components in which user actions are to be physically carried out.

Table 2: Design relationships among alternative styles.

Exclusion (s1, s2). Only one of **s1** or **s2** may be instantiated during interaction, not both of them.
Compatibility (s1, s2). Any of **s1** or **s2** may be instantiated during interaction.
Substitution (g1, g2). One group of styles **g1** substitutes another group **g2** during interaction to achieve better dialogue.
Augmentation (s1, s2). In the presence of **s1** during interaction, **s2** may be instantiated to augment **s1** dialogue.

Recording Design Documentation

During the polymorphic task decomposition process, design documentation is recorded by capturing briefly, for each sub-task, the underlying design logic, which directly associates user- / usage-context- parameter values and design goals with the constructed artifacts. In Table 3, an instance of such documentation records is shown for one of the two styles provided in Figure 1.

Table 3: Sample of recording design documentation.

Task: Delete file
Style: Direct manipulation.
Users: Expert, frequent, average.
Contexts: *Not applicable*.
Targets: Speed, naturalness, flexibility.
Properties: Object first, function next.
Relationships: Exclusion (*all*).

Design Re-use and Incremental Design

The unified user interface design method emphasizes the organization of design artifacts around a hierarchical structure, with the following well defined indexing parameters: (sub-) task, style, user / context attribute values, key design targets and interaction properties, and inter-style relationships. In the implementation phase, when design artifacts become implemented dialogue patterns, indexing parameters are also to be represented in software form and attached to their respective dialogue components. This

allows dynamic retrieval of dialogue components for adaptation-oriented dialogue assembly, based on design-oriented semantics. This feature, which is explicitly introduced in the unified implementation method (Savidis & Stephanidis, 2001), facilitates the location of artifact implementations through queries engaging design parameters. Hence, designers may search and retrieve directly from implemented unified interfaces particular dialogue components; i.e., unified interfaces are "reusable".

Additionally, updates or extensions on the original design can be systematically applied, since all artifacts are well documented and organized. Potential changes which can be made due to dialogue updates, or modifications on the underlying operation features, can be applied on either polymorphic or unimorphic artifacts. As a result, artifacts may be changed, created or dropped. Since the unified design method requires explicit annotation of the key design parameters, all updates are incorporated in a straightforward manner within their respective locations (according to the design parameters) in the unified design documentation structure.

KEY RESEARCH QUESTIONS

While designing for diverse users and usage-contexts, aiming to provide interfaces capable of self-adapting to individual users and contexts, the following key issues emerge.

Capturing User Diversity

Is it feasible to reveal those human-personality related parameters which are likely to affect the way interaction should be delivered? Is it possible practically, theoretically or legally to make such information available through a software system? For instance, consider the following user attributes: *user anxious, in a hurry, tired, does not understand the user interface feedback*. One way of reliably detecting such attributes is by employing methods such as:

- "Body language" analysis.
- Heart-beat rate or blood pressure monitoring.
- Facial expression analysis.

It is clear that there are a number of ethical issues that may need to be carefully considered.

Achieving Optimal Diversity-Based Design

In this context, optimal design concerns both quality and cost. There are two issues that need to be addressed. The first is how to decide, for particular user-attribute values that differ, if differentiation of design artifacts is inherent and necessary. The second issue concerns the way cost-effective design can be carried out, and what type of techniques can be employed to make the overall design process less resource-demanding or at least cost-effective. It is anticipated that design re-use, as well as available infrastructure, will provide considerable help in this respect, (i.e., theories, frameworks, tools, processes).

SUMMARY

The unified design process is intended to enable the "fusion" of potentially distinct design alternatives, suitable for varying instantiations of the design parameters (such as users and usage-contexts), into a single unified form, as well as to provide a design structure which can be easily utilized for a target realisation (such as software implementation). In the context of HCI, the method is considered to be especially relevant for the design of systems, which are required to exhibit self-adapting behavior, in order to support individualization to different target user groups and usage-contexts. In such interactive applications, the design of alternative dialogue patterns is necessary due to the varying requirements and characteristics of end-users.

In terms of process, the method postulates polymorphic task decomposition as an iterative engagement through which abstract design patterns become specialized to depict concrete alternatives suitable for the designated situations of use. Through polymorphic task decomposition, the Unified User Interface design method enables designers to investigate and encapsulate adaptation-oriented interactive behaviors into a single design construction. To this effect, polymorphic task decomposition is a prescriptive guide of what is to be attained, rather than how it is to be attained, and thus it is orthogonal to many existing design instruments.

REFERENCES

1. Johnson, P., Johnson, H., Waddington, P., & Shouls, A. (1988). Task-related knowledge structures: analysis, modeling, and applications. In D.M. Jones, & R. Winder (Eds.), *People and computers: from research to implementation - Proceedings of HCI '88* (pp. 35-62). Cambridge: Cambridge University Press.
2. Stephanidis C. (2001). User Interfaces for All: New perspectives into Human-Computer Interaction. In C. Stephanidis (Ed.), *User Interfaces for All – concepts, methods and tools* (pp. 3-17). Mahwah, NJ: Lawrence Erlbaum Associates. (ISBN 0-8058-2967-9, 760 pages).
3. Savidis, A., Stephanidis, C. (2001). The Unified User Interface Software Architecture. In C. Stephanidis (Ed.) *User Interfaces for All – concepts, methods and tools* (pp. 389-415). Mahwah, NJ: Lawrence Erlbaum Associates. (ISBN 0-8058-2967-9, 760 pages).