EXHIBIT 1005

What makes a good User Interface pattern language?

E. Todd
e.todd@massey.ac.nz

E. Kemp
e.kemp@massey.ac.nz

C. Phillips
c.phillips@massey.ac.nz

Institute of Information Sciences & Technology
Massey University
Palmerston North
New Zealand

Abstract

A developer of user interfaces (UI) should be able to employ a user interface pattern language to design acceptable user interfaces. But, what makes a good pattern language? Three types of validation were identified as requiring consideration: the validity of the individual patterns, the internal validation of the pattern language and the external validation of the pattern language. This paper investigates internal validity. A set of six tests that a developer can use to test the internal validity of a pattern language has been identified.

Keywords: Pattern languages, user interface design, pattern language validation

1. Introduction

When an engineer designs a solution to a well known problem they normally refer to standards, guidelines, or templates representing accepted models for solving that type of problem. Civil engineers will refer to plans of bridges for similar foundation conditions, flood flows and anticipated traffic density when designing a crossing for a watercourse.

Software engineers are increasingly using patterns to define recognised good solutions for known types of problem. Patterns can present acknowledged good practice to guide many software engineering tasks. Users of CASE tools such as Rational Rose and Model Maker have access to pattern templates that can be loaded directly into their models for modification and use. However, the software engineer is given little guidance about combinations of patterns to use together.

Java developers identified this problem of how to use combinations of patterns as a major issue. John Cruppi (2001) reported that the participants in a technology developer focus group observed

> "… that they did not have a good handle on how and when to use patterns together to solve a business problem" (p 6)

Mullet (2002) also identified organisation as a major problem when using patterns to guide UI design. The participants in the CHI2002 UI patterns workshop (McInerney, 2002) when considering pattern collection evaluation reported that:

> "Collections lack guidance on how to use patterns together as components to solving a larger design problem." (p 3)

This problem is probably generic for any user of patterns who is trying to solve real world design problems if the relationships between the patterns have not been clearly defined, regardless of whether the domain is software development, user interface development or architecture.

Collections of related patterns, which are organised and linked into one or more interlocking hierarchies may be referred to as a pattern language. A pattern language should be able to provide guidance on how to successfully use combinations of patterns from a collection. The remainder of this paper discusses one approach for determining the validity of the internal connections between patterns that make up a potential pattern language.

2. Background

Patterns and pattern language concepts are derived from architecture and were first proposed by Alexander, Ishikawa and Silverstein (1977). Salingaros (2000) identifies pattern languages as useful because they are:

> "a way of understanding, and possibly controlling, a complex system … [they are] necessary design tools with which to build something that is functionally and structurally coherent" (p 154)

Controlling complex systems and building a functional and structurally coherent system are requirements of software engineering, which may explain why software engineers were early appliers of the concept of patterns for use in program development (Coplien and Schmidt, 1995). There are also numerous books referring to patterns published by software engineers (Alur et al., 2001, Gamma et al., 1995). More recently texts specifically related to User Interface (UI) development with associated pattern languages have appeared (Borchers, 2001, Duyne et al., 2003).

UI patterns became visible to the software engineering practitioner community when Jennifer Tidwell's (1998) paper "Common Ground" became available on the web. But, the earliest UI related references to Alexander's seminal works on patterns are in papers published in "User Centered System Design" (Norman and Draper, 1986). Since then a number of sets of UI related patterns have been published. Some are referred to as 'collections of patterns', like those found in the Amsterdam collection (Welie, 2001), while others are described as "pattern languages" (Borchers, 2001, Duyne et al., 2003).

3. Methodology

This research is associated with the CONDUIT project; researching tools for aiding user interface design ((Phillips and Kemp, 2002). It was motivated by the requirement to provide guidance for user interface engineers when they are designing new user interfaces. Griffiths and Pemberton (2001) regard UI patterns as more useful tools than UI Standards and Guidelines.

Some of the UI pattern collections (Duyne et al., 2003, Tidwell, 1998) included questions for choosing patterns when solving a specific problem. These collections were used to analyse existing interfaces to investigate the usability of the resulting models. One outcome was the realisation that the links guiding selections of further patterns were confused, leading to the research question guiding this project; what attributes define a quality UI pattern language? Few studies have considered the validity of a pattern language but analysis of those found led to the identification of potential questions for testing the validity of a pattern language.

These questions were used to guide case studies investigating a number of existing UI pattern collections. An analysis of these lead to a rewording of the questions and an initial attempt at determining whether the tests posed by the questions are realistic when applied to a specific collection of patterns.

4. What is a UI pattern language?

Appleton (1997) has provided a detailed discussion of pattern related terminology. He defines a pattern language as:

> "…a collection of such solutions [patterns] which at every level of scale, work together to resolve a complex problem into an orderly solution according to a predefined goal." (p 16)

Appleton highlights the underlying hierarchical nature of a pattern language based on some identifiable scale factors. As well, he identifies the concept of the shared objective or predefined goal, which could identify a root for the language map. This definition indicates that the links between the patterns should exhibit some recognisable harmony. The definition provided by Salingaros (2000) also highlights these "inherent structures and relationships". He considers it is the connectivity rules between patterns that make a collection of patterns into a language:

> "A pattern is an encapsulation of forces; a general solution to a problem. The 'language' combines the nodes [patterns] together into an organizational framework" (p 154)

Pemberton (2000) when discussing pattern languages again identifies connectivity between patterns as the attribute that adds power to a pattern language:

> "The fact that individual patterns are integrated into pattern languages…enables the collection for patterns to operate generatively, each pattern showing the sub-patterns required to resolve more detailed design issues, in the context of the larger design" (p 5)

Borchers (2001) presents a formal definition for a pattern language, which emphasizes connectivity. He agrees with Pemberton and Salingaros with the observation that:

> "The *context*, together with the *references*, represents the added value that turns a loose collection of patterns into a pattern language." (p 71)

The different definitions all indicate that the linking between patterns on one level can form a higher-level pattern that includes information not available from the individual patterns alone. This additional information is not available to the constituent patterns of the lower level. Grouping that links lower-level patterns to higher-level patterns creates a hierarchy of scale. This implies that it should be possible to describe a user interface at different levels within a UI pattern language hierarchy. That is, at different scales rather like a road map can be either low level showing just the main highways, or high level showing details of the transport network from highways down to walking tracks.

Salingaros (2000) makes the comment that: "A loose collection of patterns is not a system, because it lacks connections" (p 154) implying that the quality and nature of the connections between patterns is what determines whether a collection is a language or not.

5. Validating UI Pattern Languages

Salingaros (2000) identifies two forms of connectivity when discussing pattern languages external connectivity and internal connectivity. These two forms of connection are central to validating a pattern language.

External validation - Considers the relationship the language has to human function or behaviour, or the "feel right" factor.

Internal validation - Examines the connectivity between the levels in the language's hierarchy to determine the "ability to combine" to describe higher order patterns.

When discussing external validity Salingaros (2000) implies that the language has internal validity. External validation is related to the 'Value System" identified by Fincher (1999), which refers to that attribute of a pattern language that "is reflected by, and embodied in, their sense of audience" (p 2). For user interfaces there are two obvious audiences, UI designers and the user group that will work with the resulting system. If the language has external validity then the reader should recognise this if they: Approach the language from the bottom-up; can progressively build up in their mind the connectivity map from the small to the large in a natural progression; feel a sense of connection with the lower order patterns, because these patterns relate to their own experience.

This human dimension is a defining attribute of Alexander's pattern language (Alexander et al., 1977) that Salingaros (2000), when discussing external validation, highlights with the observation:

> "… what demonstrates the patterns' inevitability is their connection to fundamental patterns of human behaviour and movement" (p 153).

Interacting with a computer via a user interface is a human activity and the association identified within the

architectural domain may also be present within the UI domain. In more pragmatic terms external validation can be discussed in the terms identified by the CHI2002 workshop participants as suitable for evaluating a pattern language: breadth, depth, applicability, clarity and convenience.

Internal validation examines how related patterns are linked together. A graph created by connecting up every pattern in a pattern language is referred to as the language map. Salingaros (2000) makes the observation that a pattern language map is not a simple hierarchical tree structure as a pattern language may have more than one root, although he implies a sub-system within the language may culminate in a single root node.

When discussing what characterises patterns and pattern languages have, Fincher (1999) identified five elements that need to be considered. 'Capture of Practice', 'Presentation' and 'Abstraction' refer solely to patterns. The other two elements 'Value System' and 'Organising Principle' are identified as attributes of a pattern language. 'The 'Organising Principle' refers to the way that patterns can be related to each other so that they can be arranged based on some recognisable scaling factor such as system level to widget level. Whereas the 'Value System' relates to external validation 'The 'Organising Principle' relates to internal validation. Salingaros (2000) confirms this relationship when he states that:

> "One of the principal methods of validating a pattern language is that every pattern be connected vertically to patterns on both higher and lower levels." (p 156)

Borchers (2001) when defining a pattern language identifies the context of a pattern as those patterns that reference it. This indicates that languages defined under Borchers definition should have the same map for the context links as for the reference links. As he says 'the context is the "inverse function" of the references' (p 72).

This observation is consistent with the research by Salingaros (2000), which shows that internal validity of a pattern language can be established by examining the connectivity between patterns. He says:

> "Graph theory visually illustrates some key aspects of pattern languages: how patterns combine to form higher-level patterns containing new information; how linked patterns exist on different levels; how to find patterns in a new language; and how a pattern language is validated through its connective structure independently of each individual patterns validity" (p 149)

The Alexandrian pattern language (Alexander et al., 1977) has different context and reference maps, which Borchers (2001) explains is because:

> "Alexandrian patterns are, above all, a didactic medium for human readers, even (and especially) for non-architects. To Alexander this quality has priority over a mathematically correct representation" (p 22)

Salingaros (2000) does not draw attention to the inconsistencies in the Alexandrian pattern language but he does indicate that languages are developing and evolving and may contain inconsistencies at any point in time. He discusses how languages may evolve indicating that as a language matures the connections within the levels increase and a language:

> "… develops coherence overtime [and] may also develop a degree of self-similar scaling as a result of the connections across levels" (p 159).

He comments that:

> "The most elegant complex systems are nearly (but not perfectly) ordered." (p 159)

Using Salingaros's implication that the richness of connections between levels and within levels in a pattern language is a factor in determining a language's internal validity, by developing the map of an existing UI pattern collection the software engineer should get an indication as to the status of the collection. It should be possible to organise nodes within the map into a hierarchy where the higher-level patterns provide a conceptual description of an interface but also provide the context in which the lower level patterns could be used. Reading across levels could provide descriptions at different degrees of granularity. Links within levels may indicate that a language is developing maturity as "[readers] can better understand a language if it has organisation at different levels" (Pemberton, 2000), p 146).

Both Salingaros and Borchers identify spatial hierarchies based on size from small-scale objects up to larger-scale ones. Borchers says that there are two spatial dimensions possibly three that need to be considered in UI pattern hierarchies. Both writers identify a temporal hierarchy that relates patterns that follow each other in time (i.e. an object based on one pattern can not be accessed before an object based on a proceeding pattern has been accessed). Each of the different types of hierarchy may provide an alternative view of participating patterns thereby providing alternative views of the user interface like a topographical map is different from a cadastral map.

6. Determining internal validation

From the preceding discussion it is clear that to determine whether a collection of patterns is a pattern language first, its internal validity should be evaluated. Six questions have been proposed as forming the basis of *'tests'* that can be applied to a pattern language to determine whether a language has internal validity. The test questions are:

Test 1- Do the reference and context links between the patterns form a map? *(Borchers, Fincher, Pemberton, Salingaros)*

Test 2- Does the context map match the reference map? *(Borchers, Salingaros)*

Test 3- Can the map be ordered into a hierarchy of levels? *(Borchers, Fincher, Pemberton, Salingaros)*

Test 4- Can the levels be used to describe a user interface at different degrees of granularity (scale)? *(Fincher, Pemberton, Salingaros)*

Test 5- How 'rich' are the links within each level of the hierarchy? *(Salingaros)*

Test 6- Can the patterns be organised by different classification systems thereby providing alternative viewpoints? *(Borchers, Salingaros)*

The first four of these test questions can be used to

determine whether a collection of patterns has developed sufficiently to be classed as a pattern language. Once internal validation has been established then external validation can be investigated.

7. Internal validation of Pattern Collections

The six tests were used to evaluate three existing collections of patterns. Two of these pattern collections are maintained by van Welie, the graphical user interface pattern collection (GUI collection) and the web interaction design patterns collection (WEB collection). The third collection is Borchers' human computer interface pattern collection (HCI collection).

These collections were selected because they contain a manageable number of individual patterns and are representative of the types of user interface pattern collections studied so far. van Welie's (2001) web site contains three related pattern collections. These collections are works in progress, updated at irregular intervals. Not only are new patterns added to the collection, the format used to describe the patterns has been modified over time. Borchers' classifies his collection as a pattern language but van Welie refers to his as pattern collections.

7.1. Validating the GUI Pattern Collection

The GUI collection developed by van Welie contains twenty-seven patterns. These have been arranged, into six groups: Modes, Navigation, Guidance & Feedback, Presentation, Physical interaction and Selection. The context section of each pattern defines the functional situation in which the pattern can be used without referencing higher-level patterns. Context and reference patterns are identified in the "Related patterns" section.

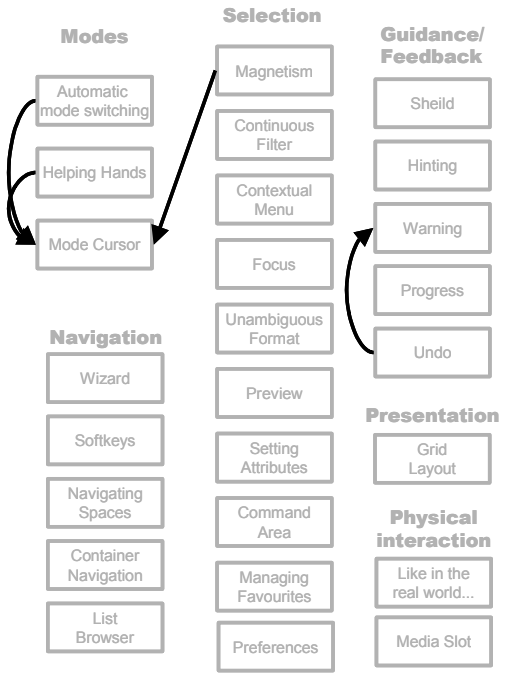

Figure 1 – Language Map for GUI collection.
(Website accessed – August 2003)

A language map for the GUI patterns was created from the links mentioned in the context and reference sections (Figure 1). There are few linkages between the individual patterns; therefore the collection does not pass the first test. This analysis indicates that the GUI collection is, as the author indicates, not mature enough to be referred to as a pattern language.

7.2. Validating the WEB Pattern Collection

In the WEB collection the context links are defined in a section labelled "Use when" and the reference links are found in the section labelled "related patterns". At the time the collection was accessed there were fifty-five defined patterns plus fifteen potential patterns that had not yet been defined. The patterns were organised into seven groups based on functionality. This collection was analysed using a similar approach to that used for the GUI collection. The language map is shown in Figure 2. Potential patterns are represented by dashed rectangles with names in italics. Nearly half of the fifty-five fully defined patterns are not linked into the language map; therefore the collection fails the first test.

Three different types of link were identified in Figure 2 from the context and reference links:

1. Links just mentioned in the context section are shown as dotted-lines with arrows.
2. Links just mentioned in the reference section are shown as dashed-line s with arrows.
3. Links identified in both sections are shown as solid-lines with arrows.

All links would be represented by solid lines if the context map and the reference map matched, as required to pass Test 2. Only two matches were found so the second test also failed.

A second language map was created for this collection including all links mentioned in each pattern's definition (Figure 3). The direction of a link was determined from the context within the definition. Only six patterns remained unlinked on this map therefore it may pass Test 1 if the pass condition is relaxed. Matching the context and reference maps is not possible, as the current descriptions do not clearly identify the context and reference links. Therefore Test 2 must fail.

Examining both potential language maps for the WEB collection indicates that there is no one pattern that could become a root node for a hierarchical structure. The seven groups that have been used to organise the patterns may become recognised levels within a hierarchy but as yet only sub-trees and linked lists of patterns can be identified. Therefore the collection does not pass Test 3 and it follows that it is inappropriate to apply Test 4.

The links between patterns within the classification groups may denote a degree of richness developing within levels of the collection, indicating that in the future Test 5 may be met. It is also possible that the patterns classified as 'Site Types' could become roots of an interlocking set of hierarchies, which indicates that Test 6, may eventually be passed.

DOCKET ALARM

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS
Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS
Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS
Sync your system to PACER to automate legal marketing.