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.

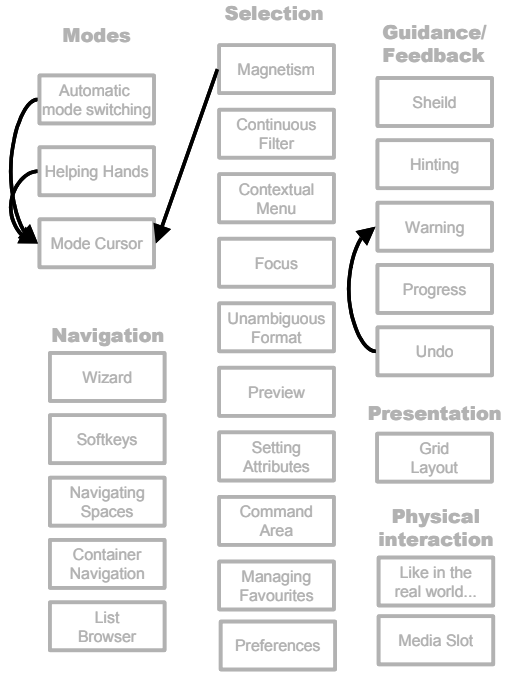

Figure 1 – Language Map for GUI collection.
(Website accessed – August 2003)

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.

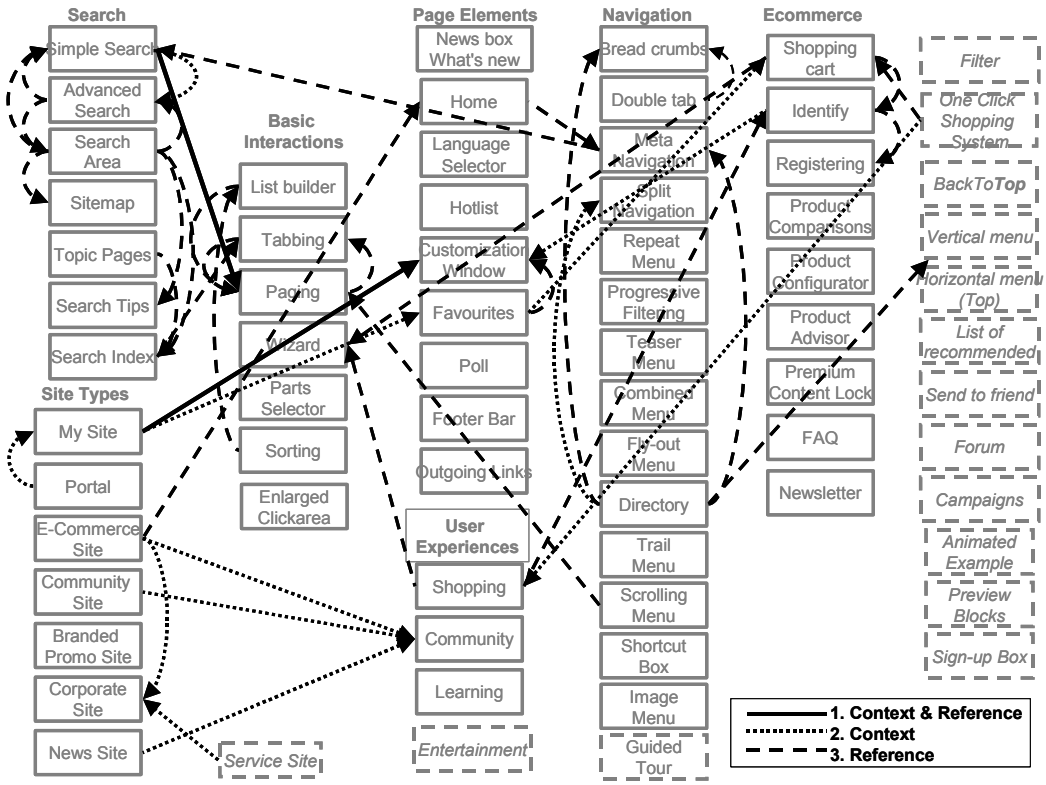

Figure 2 - Language Map for WEB showing just prelude and postscript links. (Website accessed - April 2003)

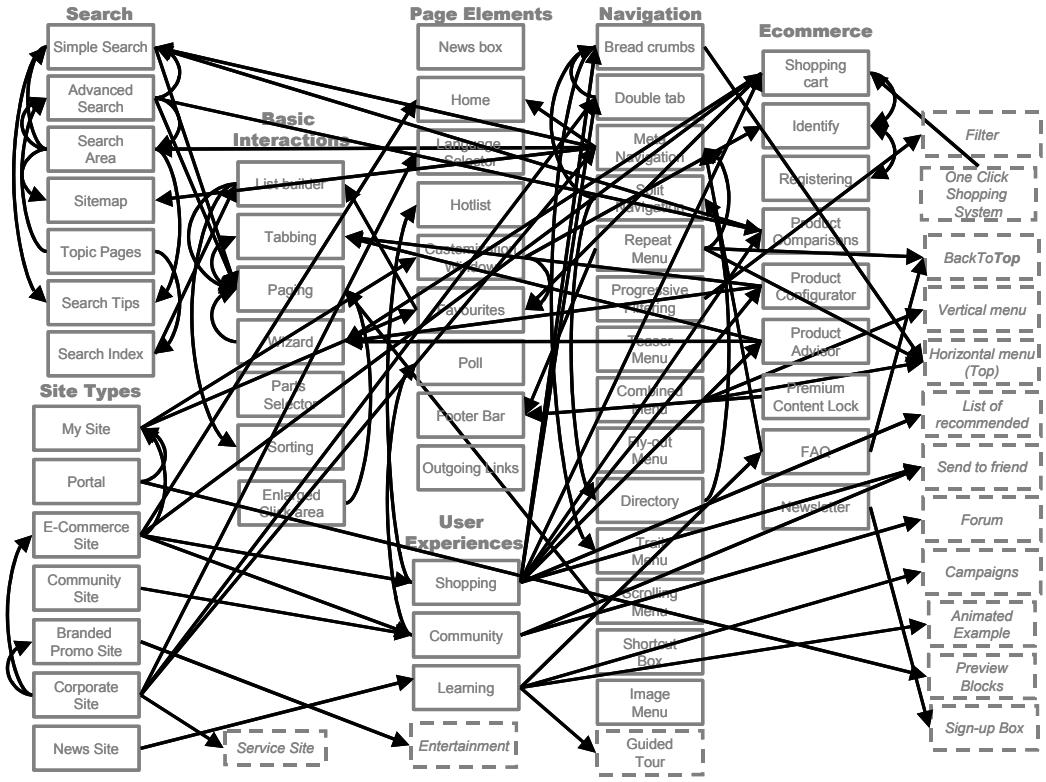

Figure 3 - Language Map for WEB collection showing all referenced pattern links.

The "E-commerce Site" sub-system is reasonably well developed containing twenty-nine patterns and therefore warranted further investigation. Figure 4 shows this collection of patterns, reorganised into a hierarchical structure with a single root and four recognisable levels. The E-commerce sub-system passes Test 1 as all members are linked. It does not pass Test 2 for the same reason that the WEB collection fails. It does pass Test 3 as it can clearly be organised into a hierarchy. Although there are obvious anomalies in its current state it may also pass Test 4. The root node defines the application area.

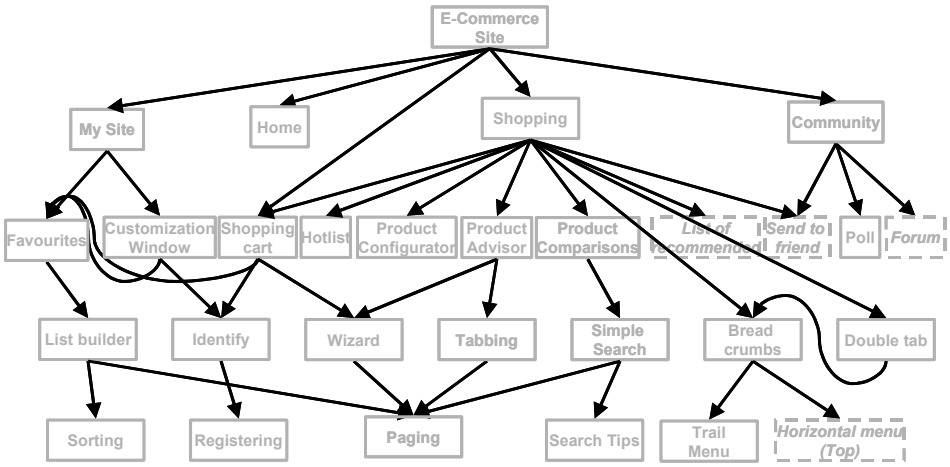

Figure 4 – The E-commerce sub-system hierarchy of the WEB collection.

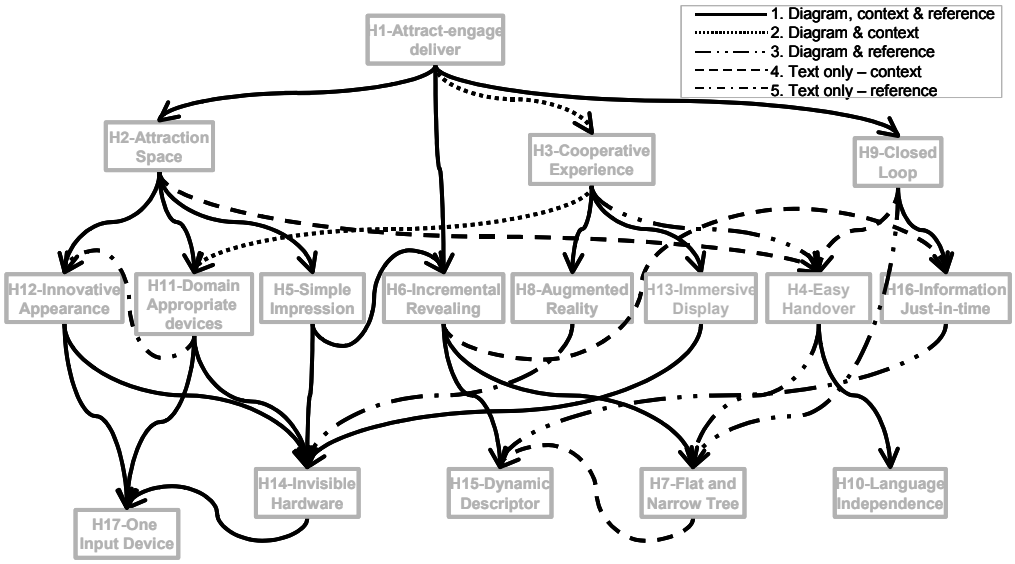

Figure 5 - Idealised HCI collection language map combining all sources of link information.

The first level indicates the basic task-structure provided for the user. The second level defines the main components of the interface and the lower levels are moving towards the abstract description of widgets. This collection has not developed sufficiently to pass either Test 5 or Test 6.

7.3. Validating the HCI Pattern Collection

Borchers (2001) HCI collection is published with a map of the language similar to that shown in Figure 5. Each of the patterns has a reference number but although the root is identified by 'H1' the higher numbers do not identify only the patterns in the lowest level as in Alexandrian pattern language (Alexander et al., 1977).

Borchers' map (2001) has been compared with the definitions of the individual patterns by matching the context and reference links found in the pattern definitions with the lines on the map. The pattern descriptions are similar to the Alexandrian format (Alexander et al., 1977). The types of line representing the links on the diagram in Figure 5 are the result of this analysis Five states were identified:

4. The link appears on the map and both the pattern's context prelude and reference postscript agree;
5. The link appears on the map but the reference postscript of the pattern does not agree with the associated pattern's context prelude;
6. The link appears on the map but the context prelude of the pattern does not agree with the associated pattern's reference postscript;
7. There is no matching link on the map but there is a reference in the context prelude of the pattern;
8. There is no matching link on the map but there is a reference in the reference postscript of the pattern.

There are no links represented on Borchers' language map that are not also defined in one of more of the pattern definitions. The difference between the map given by Borchers (2001) and the one in Figure 5 is those lines represented by the dashed lines and the dash-dotted lines. These lines represent states four and five above. All patterns participate in this language map therefore Test 1 is passed.

The second analysis of the HCI language used the results of the first to create a context map and a reference map Figure 6. A comparison of these two maps as indicated from the initial analysis, shows that they do not match. This result indicates that this collection will fail Test 2 if it is rigorously applied.

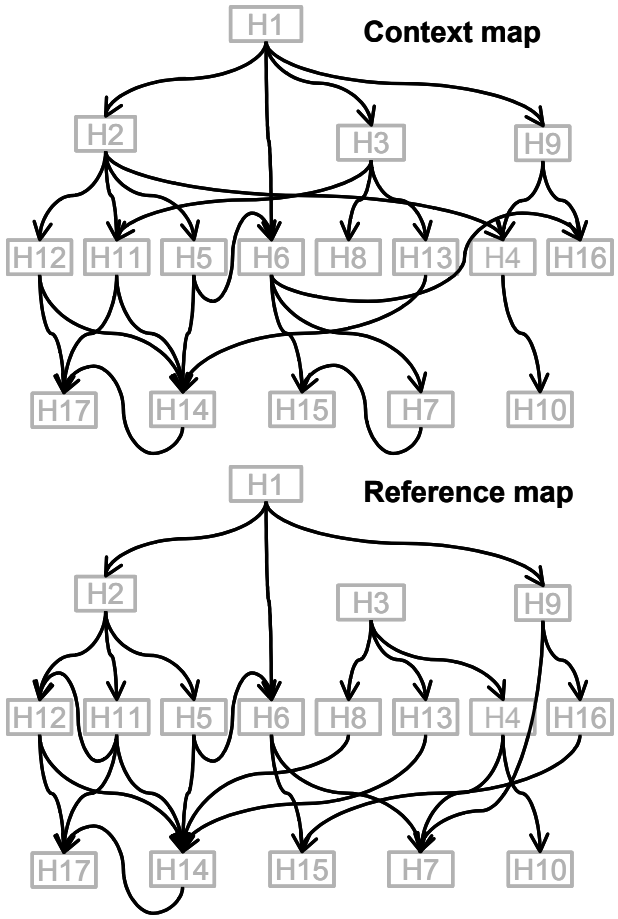

Figure 6 - Comparison of the topology of the network of links for the Context map and Reference map

This language is composed of seventeen patterns, which can be arranged into a three-level hierarchy (Borchers shows four levels as he places pattern H17 on a level by itself). Clearly, the HCI collection passes Test 3.

When examining the language map, reading the patterns across the levels should describe the user interface at that scale. Looking at the Level 1 context links of the HCI collection, there appears to be a problem with the positioning of the pattern H6 Incremental Revealing. Should it be part of Level 1 or in Level 2 as shown? A reading of the three patterns (H2, H3 and H7) that make up Level 1 indicates that a high-level description of the interface is possible without including pattern H6. Reading across this level indicates that the designer first considers designing how the system will look when first encountered so it will attract the attention of potential users, then how to involve more than one user in a co-operative activity and finally how a user can pass over control to another user and leave the exhibit. Adding the description for H6 certainly makes the definition of the design clearer at this scale but the pattern appears to fit equally well in Level 2. H6 seems to fit the property of self-similar scaling described by Salingaros (2000).

Although there could be some debate as to the status of the definition of the interface by reading across Level 3 this language is sufficiently developed to pass Test 4.

Most links are between nodes on different levels in the hierarchy but twenty-three percent of the links are within two levels of the hierarchy indicating that the language is developing a degree of richness and has the potential to pass Test 5. As there is only one root there does not appear to be any alternative way to view the patterns in this language therefore Test 6 fails. Another limiting factor of this collection is that even if it was internally valid it is probably too small to be classified as mature.

8. Discussion

From the initial discussion and the three case studies reported above it is clear that applying strict criteria for passing each of the tests would result in all pattern languages failing to attain internal validation. Therefore to be useful as a measure of pattern language quality a less stringent set of criteria should be defined. In the following discussion each test is considered and possible pass criteria suggested.

Test 1: Do the reference and context links between the patterns form a map?

The case studies resulted in only the HCI collection, passing Test 1. Of the fifty-five defined patterns (or seventy identified patterns) in the WEB collection twenty-four are not linked into the language map when it is created using just the context and reference links. Creating a map using all links mentioning in the pattern definitions reduces the number of unlinked patterns down to six. That is a participation rate of 92%. The first question this finding raises is, how many stand-alone patterns can a collection include and still pass Test 1? A participation rate of just thirty-seven percent is clearly not sufficient for a pass but, *is a rate of over ninety percent participation in the language map sufficient?*

The inclusion of links to other patterns within the body of the pattern descriptions raises a second question, should all references to other patterns in the collection be included in the language map when a non-Alexandrian format is used? This question in turn identifies a third issue; which types of link should be considered when developing the language map? Mullet (2002)considered the three basic object-oriented class diagram relationships: derivation, aggregation and association when discussing link types. Salingaros (2000) identified five different types of link. Six link types were identified from the case studies. These are:

Contains - One pattern contains another smaller-scale pattern. *(Salingaros, Mullet)*

Derivation - Distinct patterns share a similar structure, thus implying a higher-level connection. *(Salingaros, Mullet)*

Uses - One pattern can use another normally smaller-scale pattern to solve the problem. *(Mullet)*

Complementary - Two patterns are complementary and one needs the other for completeness. *(Salingaros)*

Overlapping coexistence - Two patterns solve distinct problems that overlap and coexist on the same level. *(Salingaros)*

Alternatives - Two patterns solve the same pattern in alternative, but equally valid ways. *(Salingaros)*

The first three relationships: contains, generalise and uses clearly form the inter-level hierarchical structure of a pattern language. The other three relationships: overlapping-coexistence, complementary and alternatives are more likely to be represented as lines within a level while sharing a common pattern in their context.

Test 2: Does the context map match the reference map?

When developing a UI pattern language, checking that the context and reference maps match is mechanical but, even in a situation where the author of the pattern language has identified this criteria as important (e.g. Borchers 2001) it is still difficult to actually attain accuracy. As both Salingaros and Borchers acknowledge this ideal is not necessary but desirable.

But, why are there discrepancies between the context and reference maps even where the author of the patterns has clearly defined that consistency between these two maps is important? The idealised language map includes all links from the context map and the reference map. There are thirty-two links on the idealised language map for the HCI collection but the Context map and the Reference map identify different sets of 26 links. That is an 81% match to the idealised map. *If the difference between the context map and the idealised map is less than twenty percent and similarly between the reference map and the idealised map, is this sufficient to pass Test 2?*

Creating a language hierarchy that conforms to the definition of a pattern language is not easy. Borchers' paper (2000) included a language map (figure 3, page 375) for his HCI pattern language that has a different topology to that published in his book. In turn the version of the language map as published has been modified via the errata sheet. This map is different to that developed by analysing the text of the patterns, (Figure 5). Possibly Borchers's formal definition is never realised as the language develops towards maturity. As the language creator works with the patterns *how they can be used* and *how they can use other patterns* becomes clearer as patterns are moved between levels and additional links are discovered.

The discrepancies identified indicate that development of a pattern language is consistent with the process of 'repair and replacement' even though Salingaros (2000) is discussing language change over time rather than initial development. As languages mature inconsistencies should be identified and removed. Keeping such changes consistent between linked patterns and the associated map probably needs a computer based pattern language editor and a pattern management system.

Another issues that needs to be considered is, which relationships should be included when determining the context and reference match? Undoubtedly, the three relationships: contains, generalise and uses should be included but what about the other types of relationship that have been identified? The intra level relationships are considered when quantifying the richness of the language with Test 5.

Test 3: Can the map be ordered into a hierarchy of levels?

Of the collections analysed, there was no discernible primary hierarchy in van Welie's GUI or WEB collection. How many levels does the hierarchy require to pass Test 3? If the HCI collection is a pattern language then the following criteria may be sufficient to qualify: At least 2 levels below a root are required and in the *lower levels more than one node per level must be linked to a higher level.*

The HCI collection, the WEB collection and the E-commerce sub-system could pass this test. There are so many links on the WEB collection map it has not been possible to identify more than some chains of patterns and some sub-trees. The HCI pattern language map is very clearly a hierarchy with only one root as is the E-commerce one.

Further questions raised during the analysis associated with Test 3 are: Do all the patterns have to be linked into the primary hierarchy? Is it important to nominate a primary hierarchy? In a mature pattern language it would not be necessary and may not be desirable to have all patterns linked within a single hierarchy, as the richness of the language will be partly represented by the inter-relations between alternative hierarchies. Identifying hierarchical structures linking patterns should also identify the criteria for defining different levels of granularity or scaling factors.

Test 4: Can the levels be used to describe a user interface at different degrees of granularity (scale)?

Test 4 considers describing interfaces at different degrees of granularity. This has proven to be a difficult test for which to define a set of evaluation criteria. Mullet (2002) highlights the problem of defining degrees of granularity in his discussion of pattern language organisation saying:

> "The murky ground between application-level and widget-level patterns is the terrain that must be addressed by an organisation scheme." (p 4)

He also considers the "level of design activity that the individual patterns described" as potential levels of granularity, identifying: conceptual level patterns that describe content and organisation, patterns that describe the behaviour or feel of the interface and patterns describing the appearance or look of the interface.

The HCI language has a very narrow scope and only 17 patterns, which may explain why different levels of granularity can be identified. Each of the three levels below the root of the HCI collection, provide a description of a multi-user interactive kiosk interface and therefore it was deemed to have passed this test. No attempt has been made to determine how complete these descriptions are. A second related issue is to determine how many levels describing the interface are required to pass this test The E-commerce sub-system ranges across more levels of abstraction than the HCI collection, from the application area down to abstract widgets such as "Trail Menu". Possibly another question to consider is, how many levels of abstraction should be described?

	Map	Match	Hierarchy	Levels	Richness	Views
Van Welie						
GUI	No	No	No	No	None	No
WEB	Yes	No	Developing	No	Unknown	Possibly
E-commerce	Yes	No	Yes	Probably	Minimal	No
Borchers						
HCI	Yes	Yes	Yes	Probably	Developing	No

Table 1 - Summary of pattern language attributes for collections analysed.

The hierarchy that orders the Alexandrian pattern language (Alexander et al., 1977) is based on a well-understood scale from a city down to such artifacts as individual doors into a dwelling. A comparable 'natural' hierarchy for user interfaces has yet to be identified. This topic was discussed at the Usability Pattern Language Workshop at Interact'99.

Test 5: How 'rich' are the links within each level of the hierarchy?

According to Salingaros (2000) a pattern language's maturity, is determined by how rich the language is. He mentions both inter and intra level links when he discusses richness. As inter-level links have already been considered by the previous test questions, a more restrictive definition of richness, which concentrates on intra-level links has been used. Even this constraint raises questions on how to quantify the richness of a language: *How many intra-level links represents richness?* And: *Can there be too many intra-level links?*

No clear conclusion could be determined from either the GUI or WEB collections. Over twenty percent of the links in the HCI collection are intra level links and occur in two of the three levels below the root. Should the HCI collection be defined as rich? As a classification criterion such as 'rich' is by its nature subjective, a relative scale for comparing language maps was preferred. To compare the collections analysed, a six-point scale of richness is proposed:

None – no observed intra level links in any level of the hierarchies identified.

Unknown – so many links the user becomes confused.

Minimal – less than ten percent of the links are intra level links.

Developed in one level – at least a third of the reference links within only one level are intra level links.

Developing - more than ten percent of the links are intra level links and occur in more than half of the levels below the root.

Rich - more than thirty percent of the links are intra level links and occur in more than half of the levels below the root.

Test 6: Can the patterns be organised using different classifications of categories thereby providing alternative viewpoints?

Salingaros states that a pattern language "has multiple tops and horizontal connections" (Salingaros, 2000). Because only one hierarchy was traced through the patterns of the HCI collection and the E-commerce sub-system no obvious alternative views were identified. The implication being that an alternative view will start with an alternative root. Is this a valid proposition? The Alexandrian pattern language has only one root, Pattern 1 " INDEPENDENT REGIONS" (Alexander et al., 1977) but Salingaros reports that different views can be identified. Therefore some other method of identifying alternative viewpoints needs to be defined.

A number of authors have proposed alternative categorisations for their pattern collections (Welie and Traetteberg, 2000, Tidwell, 1998), some with a different purposes to that of using combinations of patterns. These take the form of lists of patterns from the collection that fall into each of the categories. This leads to the question, what is the nature of the links when representing an alternative view?

The E-commerce sub-system is part of the WEB collection, so is it just one view of part of that collection? What is the difference between a linked subset or sub-language and an alternative view? Including all the patterns in the language in an alternative view as extreme as insisting that views be mutually exclusive non-overlapping sub-sets. Which raises the issue of how many patterns have to participate in a view?

9. Summary

Using the relaxed criteria discussed above, the six tests for evaluating internal validity were applied to the three pattern collections plus the E-commerce collection (Table 1). Based on this analysis, as would be expected in such a young discipline as UI Design, none of the collections can be described as mature. Only the HCI collection could be classed as a pattern language. It definitely passes tests 1, 2 and 3. A scale of relative values has yet to be developed for determining a pass for Test 4 but it is expected that the HCI collection should score relatively highly. The HCI collection is not internally valid, as it does not yet pass all six tests. It could be considered to be a sub-language because the collection represents a very specific type of interface rather than generic user interfaces.

The E-commerce sub-system almost classifies as a sub-language. Re-wording the descriptions of the patterns with the goal of creating a language should result in this collection passing the first four tests.

More work is required to determine how UI patterns can be categorised. This should lead to the definition of a suitable set of criteria for applying Test 4. Identifying suitable categorisation should also help determining how to evaluate Test 6.

It is possible that if a pattern language passed a strict application of all six of the internal validity tests then it maybe obsolete.

10. Conclusions

The tests developed, even though still incomplete have been shown to be useable. Applying the tests is

straightforward, but a software engineer will have to exercise a degree of judgement when deciding whether a language meets a specified criterion. Creating a hierarchically structured language map with matching context and reference maps should be solvable with a pattern management system, once the nature of the links that should be included has been determined.

The development of such a system is a goal of this research. A pattern management system should include a tool for editing patterns so that a pattern writer can create dynamic links to other patterns and define the link types. More research needs to be carried out on the nature of the types of link to be used to create the language map. Probably a subset of link types will be used while other links add to the richness of the language.

Over time many of the patterns described in the different collections may become amalgamated to create better patterns. Some will be discarded and those that survive will eventually be linked into an accepted UI pattern language structure. Salingaros (2000) suggests that a mature language is not only rich within itself it also has links to languages from other related disciplines. Collections of patterns that constitute a UI pattern language may in the future have links to design patterns, enterprise patterns and management patterns

11. References

Alexander, C., Ishikawa, S., Silverstein, M., Jacobson, M., Fiksdahl-King, I. and Angel, S. (1977): A Pattern Language, New York, Oxford University Press.

Alur, D., Crupi, J. and Malks, D. (2001): Core J2EE Patterns: Best practices and design strategies. Upper Saddle River, Sun Microsystems Press, Prentice-Hall.

Appleton, B. (1997): Patterns and Software: Essential Concepts and Terminology. *Object Magazine Online*, 3(5): 16.

Borchers, J. O. (2000): A pattern approach to interaction design.. *Proc. Designing interactive systems: Processes, practices, methods, and techniques.*

Borchers, J. O. (2001): *A Pattern Approach to Interaction Design, Chichester*, England, Wiley.

Coplien, J. O. and Schmidt, D. C. (Eds.) (1995): *Pattern Languages of Program Design*. Reading, Addison-Wesley.

Cruppi, J. (2001): Core J2EE Patterns. *Java Developer's Journal*, **6**(8): 9.

Duyne, D. K. v., Landay, J. A. and Hong, J. I. (2003): *The Design of Sites - Patterns principles and processes for crafting a customer-centred web experience*. Boston, MA, Addison-Wesley.

Fincher, S. (1999): What is a Pattern Language. From Patterns workshop at *INTERACT'99* Edinburgh. http://www.cs.ukc.ac.uk/people/staff/saf/patterns/. Accessed Feb 2003.

Gamma, E., Helm, R., Johnson, R. and Vlissides, J. (1995): *Design patterns: Elements of reusable object orientated software.* Reading, MA, Addison Wesley Longman.

Griffiths, R. and Pemberton, L. (2001): Use Pattern Languages not Guides. http://www.it.bton.ac.uk/staff/lp22/guidelinesdraft.html. Accessed 10 Dec 2002

McInerney, P. (2002): UI Patterns. *Patterns Workshop, CHI2002*. 13.1-13.5. Philidelphia. http://www.welie.com/patterns/chi2002-workshop/index.html Accessed Oct 2003.

Mullet, K. (2002): Structuring Pattern Languages to Facilitate Design. *Patterns Workshop, CHI2002*. 15.1 -15.11. Philidelphia. http://www.welie.com/patterns/chi2002-workshop/index.html Accessed Oct 2003.

Norman, D. A. and Draper, S. W. (eds.) (1986): *User Centered System Design: New perspectives on Human-Computer Interaction*. Hillsdale, New Jersey, Lawrence Erlbaum Associates.

Pemberton, L. (2000): The Promise of Pattern Languages for Interaction Design. Human Factors Symposium, Loughborough, UK. http://www.it.bton.ac.uk/staff/lp22/HF2000.html. Accessed Oct 2003

Phillips, C. and Kemp, E. (2002): In support of User Interface design in the Rational Unified Process. *Proc. Australian User Interface Conference*. Melbourne.

Salingaros, N. (2000): The Structure of Pattern Languages. *Architectural Research Quarterly*, **4**:149-161.

Tidwell, J. (1998): Interaction Design Patterns. *PLOP'98*, Illinois.

Welie, M. v. (2001): Interaction Design Patterns. http://www.welie.com/patterns/ Accessed Sept 2002..

Welie, M. v. and Traetteberg, H. (2000): Interaction Patterns and User Interfaces. *Proc. Pattern Languages of Programming*. **7**. Monticello, Illinois, USA. http://monkey.icu.ac.kr/sslab/proceeding/PLoP2000/. Accessed Oct 2003.