[This note has a parallel existence as a draft of an article invited for trial issue of IEE Informatics, aimed at a general computer-literate audience. Suggestions for improvement in the next few days are most welcome.]

Web site wanderer

"Pattern" – one of the latest of many words to be taken over and given a completely new technical meaning. Not that it didn't have technical meanings enough already of course; ask the artificial intelligence practitioners and the functional programmers.

In software engineering, design patterns are another twist in the reuse tale. A design pattern can be seen as a reusable chunk of design expertise. The idea is that patterns help novice designers produce more expert designs, by clearly explaining common design problems together with solutions which are known to work. Think of dress patterns: the seamstress still has to decide what variant of the pattern to follow, make minor adjustments and make up the garment (implement the design), but the pattern makes the process of design much easier. (Where the analogy breaks down is that a software design pattern normally covers only a small part of the design of a system.) In practice the usefulness of design patterns is not limited to novice designers; the names of design patterns provide a high-level addition to the design vocabulary, which is useful to experts as well.

Probably, though, you'd heard of software design patterns before; perhaps you've read the Gang of Four book. You know there's a lot more information out there on the web. Where do you start?

Of course, you go to your favourite search engine and search for "pattern" – but in this case, the word is too overloaded for that to be all that useful; and perhaps you don't know what to add to make the search more specific. In this column, I'll take you on a tour through some of my favourite sites.

So what is a pattern, really? A good place to begin if you're impatient is with Brad Appleton's really short (11 slides) introduction to patterns
(`http://www.enteract.com/~bradapp/docs/patterns-nutshell.html`).
For more detail, try his nearby *Patterns and Software: Essential Concepts and Terminology*
(`http://www.enteract.com/~bradapp/docs/patterns-intro.html`),
or go on to "the" Patterns Home Page
(`http://hillside.net/patterns/patterns.html`)
and start with the section **About Patterns**. This site is one of the main starting places for all kinds of information about patterns; another good one is Brad Appleton's own collection of patterns links,
(`http://www.enteract.com/~bradapp/links/sw-pats.html`)

By now you'll have seen several definitions of patterns. My favourite single sentence one is *A pattern is a structured description of a good, well-understood solution to a common problem in context.* One of the things this has in common with many others is that it doesn't say *what kind of* problem a pattern helps solve. The best known patterns are *software design patterns.* Patterns of this kind were made famous by Erich Gamma, Richard Helm, Ralph Johnson and John Vlissides in their book *Design Patterns: Elements of Reusable Object-Oriented Software*[GHJV95], known by pattern afficionados as the Gang of Four or GoF book.

History of patterns

Patterns began, however, in a quite different field, with the architect Christopher Alexander, whose books (especially *A Timeless Way of Building*[Ale79] and *A Pattern Language*[AIS+77]) are essential reading for the serious pattern enthusiast. Oddly, though, a randomly chosen architect seems much less likely than a randomly chosen software developer to have heard of Alexander. There's much less discussion of Alexander's work on the web than you might expect, but a good

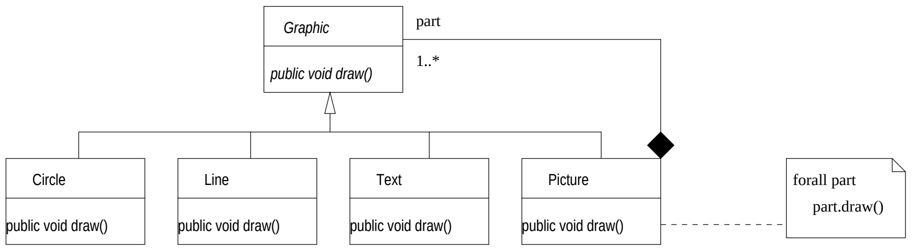

Figure 1: The structure of the Composite design pattern.

starting point is *Some Notes on Christopher Alexander*
(`http://www.math.utsa.edu/sphere/salingar/Chris.text.html`)
by Nikos A. Salingaros, who describes himself intriguingly as "mathematician and architectural theorist". There's also a brief review of the main books by Susan Stepneys at
(`http://public.logica.com/~stepneys/bib/nf/alexandr.htm`)
Alexander was the keynote speaker at OOPSLA'96 where he talked about his reactions to the way the software design community has taken up his idea; a transcript of his talk is at
`http://dlib.computer.org/so/books/so1999/pdf/s5071.pdf`.

You'll now be getting to grips with some patterns jargon, such as Quality without a Name (QWAN) (the property of being Right in some ineffable sense), generativity (the idea, of which more anon, that, between them, patterns generate coherent designs with desirable emergent properties) and the Rule of Three (the idea that a pattern should have been demonstrated to be useful in three independent cases before being trusted).

Software design patterns

In software design, a pattern captures and describes a solution to a common design problem. Usually there's an underlying assumption that the design is object-oriented; this is more to do with which community took up the idea of patterns than with any feature of patterns. For example, there's often a need to model a hierarchical structure of parts and containers for parts – think about a GUI in which some widgets, frames and the like, can contain other widgets, even other frames. The design pattern Composite (included in the GoF book) describes how to do this using what's essentially an object oriented implementation of a recursive datatype, which we can describe in UML as in Figure 1.

The pattern itself is more than just this structural description of course. The section for this pattern in the GoF book includes a discussion of the pros and cons of this pattern, its variants, how it relates to other patterns, as well as sample code for an implementation.

What is a pattern *language*? The Alexandrian idea is that patterns are generative; in software this has been interpreted to mean that whole architectures are generated from the patterns that are used in the architecture. Several patterns interacting to form designs which are more than the sum of their parts are considered analogous to words forming grammatical sentences with emergent meaning. In practice, though, most pattern "languages" are little more than collections of patterns applicable in the same domain, and most designers use only a few patterns in each system. Whether, or to what extent, this will continue to be the case as software design and its patterns are better understood is unclear and controversial.

New patterns

Design patterns didn't end with the publication of the most famous books; there are several conferences every year which include writers' workshops on patterns. A collection of announcements and links is

`http://hillside.net/patterns/conferences/`.
(ChiliPLoP, the one in Arizona, probably wins the prize for silliest conference name ever.)

By contrast with the Patterns Home Page, hillside.net/patterns/, the WikiWikiWeb at the Portland Pattern Repository is more interactive: anyone can add content to it (go to the Recent Visitors page and add your own details). The best of many possible places to start is probably
`http://c2.com/cgi/wiki?WelcomeVisitors`.
The advantage of being able to add your own content is that anyone can add patterns or proto-patterns, or descriptions of new pattern groups, or whatever. Unfortunately, the resulting design-by-committee leads (as you would expect, and as described in Jim Coplien's pattern **Architect-ControlsProduct**
`http://www.bell-labs.com/cgi-user/OrgPatterns/OrgPatterns?ArchitectControlsProduct`
to a rather hard-to-navigate web site.

Patterns in software engineering outside design

You may have noticed that **ArchitectControlsProduct** is not likely to be a pattern describing a solution to a software design problem as such. This leads on to the next point: that patterns have spread widely through software engineering as a way of codifying and transferring knowledge; they are no longer confined to software design. There are many patterns concerning *organizational patterns* at the OrgPatterns site
(`http://www.bell-labs.com/cgi-user/OrgPatterns/OrgPatterns`)
for example. Personally I'm interested in using patterns to capture expertise in reengineering legacy systems
(`http://www.reengineering.ed.ac.uk`).
There's a useful collection of patterns on user interface design at Jennifer Tidwell's site
(`http://www.mit.edu/~jtidwell/interaction_patterns.html`)
and many other areas also have active pattern communities.

Mailing lists and their archives

If you want to participate actively in discussions on patterns, there are various mailing lists, some with archives; a partial list is at
`http://hillside.net/patterns/Lists.html`
The main one is perhaps **patterns-discussion** which was once archived at
`http://www.DistributedObjects.com/portfolio/archives/patterns/index.html`
but the archives haven't been updated since last June. That's the web for you...

Patterns groups

There's a UK Patterns Special Interest Group run as a subgroup of the BCS OOPS group, but judging from its web page
`http://homepages.tesco.net/~jophran/UKPatterns/patternsUK.html`
it's not very active at the moment. There are various local patterns reading groups, which generally meet once a month or so to discuss a given pattern or pattern language. The patterns mailing lists frequently carry announcements of groups' meetings.

As with fiction, writing patterns requires another level of skill. John Vlissides' *Seven Habits of Successful Pattern Writers*, at
`http://hillside.net/patterns/papers/7habits.html`
is a valuable starting point (and might tempt you to buy his excellent book *Pattern Hatching*[Vli98]). There's even a *Pattern Language for Pattern Writing* at
`http://hillside.net/patterns/Writing/pattern_index.html`.

References

[AIS$^+$77] Christopher Alexander, Sara Ishikawa, Murray Silverstein, Max Jacobson, Ingrid Fiksdahl-King, and Shlomo Angel. *A Pattern Language: Towns, Buildings, Construction.* Oxford University Press, 1977.

[Ale79] Christopher Alexander. *The Timeless Way of Building.* Oxford University Press, 1979.

[GHJV95] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. *Design Patterns.* Addison Wesley, Reading, MA, 1995.

[Vli98] John Vlissides. *Pattern Hatching: Design Patterns Applied.* Addison-Wesley, 1998.