

# Patterns for Technology Companies

(c) Allan Kelly - [allan@allankelly.net](mailto:allan@allankelly.net)

<http://www.allankelly.net>

## 1 Abstract

This paper builds on the author's earlier work (Kelly, 2005a, Kelly, 2005b) by adding two more business strategy patterns and positioning business patterns. As such the paper splits into two distinct parts.

The first part deals with *pattern theory* with respect to *Wholeness* (aka *Quality without a name*) and the role of patterns as knowledge management tools while contrasting patterns with the emerging story telling movement.

The second section presents two patterns describing common strategies used by technology companies; specifically software companies but the strategies should be extensible to other technology sectors. These patterns are:

- *Products and Services*
- *Certified Experts*

## 2 Audience

These patterns are intended to codify several common business practices in a pattern language so they may be communicated and studied more clearly.

The patterns given here are intended for those interested in how corporate strategies may be applied. This group includes both students of the subject and new managers.

The author is interested in the applicability of the pattern form to business domain; whether the form works, what insights it can offer and what value it offers in codifying and communicating business practice.

## 3 Positioning business patterns

### 3.1 *Patterns and their environment*

“There is a central quality which is the root criterion of life and spirit in a man, a town, a building, or a wilderness. The quality is objective, precise, but it cannot be named. ...

In order to define this quality ... we must begin by understanding that every place is given its character by certain patterns of events that keep happening there. ...” (Alexander, 1979, p.ix)

In writing patterns individuals attempt to codify and document the events that create the quality Alexander speaks of. The result is a pattern or set of patterns (*a pattern language*), however these are not the patterns themselves but a human interpretation of the events and the results. As such these patterns reflect the context in which they are written: the time, the political

climate, the economic environment and, significantly, the authors own views and understanding of the world around them.

For example, in an age when global warming threatens the planet, were pollution in cities endangers health and public opinion dislikes concreting over swathes of greenery what are we to make of *Ring Roads* and *Nine Percent Parking* (Alexander, 1977)?

Patterns are created in an environment and in a period of time that brings with it its own set of assumptions, norms and standards. Patterns both document their environment and exist within it. Indeed in documenting and naming events we may actually change that which is described.

One of the primary reasons for writing patterns is to communicate our understanding. Naturally, it is easier to communicate ones own understanding to another who shares a similar environment, the greater the difference in environment the more difficult it is to communicate the idea. This is not just a matter of understanding, it is a matter of applicability.

The pattern *Nine Percent Parking* would have made little sense to any individual living before the middle of the twentieth century. This is not because the idea was badly communicated, nor does it reflect the validity of the pattern, simply, before about 1950 there was simply no environment in which this pattern would exist.

### **3.2 *Quality without a name: Wholeness***

For Alexander patterns are not an end in themselves, they are only a means to end, his objective is to generate *quality without a name* referred to above:

"To reach the quality without a name we must then build a living pattern language as a gate." (Alexander, 1979, p.xi)

More recently Alexander has chosen to name the *quality without a name*, the name is: *Wholeness* (Alexander, 2002). As this term is relatively new much of existing literature still refers to *quality without a name*, or QWAN for short. In this paper the two terms will be considered interchangeable.

Alexander is not alone in emphasising the importance of *the quality*. The comments of Dick Gabriel apply equally to software and business patterns:

"I believe we cannot come to grips with Alexander in the software community unless we come to grips with this concept [wholeness]."  
(Gabriel, 1996, p. 34)

Therefore, if business patterns are to stand alongside architectural patterns and software patterns we must consider the role of *wholeness*, or *quality without a name*.

A full discussion can be found in Alexander (1979). For the moment we may abridge Gabriel:

"The quality is an objective quality that things like buildings and places can possess that makes them good places or beautiful places. Buildings and towns with this quality are habitable and alive. ...

Alexander proposes some words to describe the quality without a name, but even though he feels they point the reader in a direction that helps comprehension, these words ultimately confuse. The words are *alive, whole, comfortable, free, exact, egoless, and external.*" (Gabriel, 1996, p.34-36)

Gabriel questions if the writers of software patterns share the same goal:

"When I look at software patterns and pattern languages, I don't see the quality without a name in them, either. Recall that Alexander said that both the patterns themselves and the pattern languages have the quality." (Gabriel, 1996, p.69)

However, even if software patterns lack the quality without a name they can still be useful and play a positive role. Gabriel outlines one of the ways in which software patterns are useful:

"Patterns provide several benefits to programmers and system developers. One is as a common language. ...

Another benefit is as a common base to understanding what is important in programming. ...

A third benefit is that with a corpus of patterns a programmer is able to solve problems more rapidly by having available a storehouse of solutions - a cookbook, if you will." (Gabriel, 1996, p.51-52)

We may generalise Gabriel's discussion from software patterns to encompass our consideration of business patterns. So, we may ask three things of a pattern:

- Is the pattern useful?
- Does the pattern possess the *wholeness* itself?
- Does the pattern contribute to the generation of the *wholeness*?

The answer to the first of these questions largely depends on the reader's point of view and experience. Answering the second and third is difficult because we do not have an objective standard for determining if a pattern actually possesses *wholeness*.

As noted above, patterns both describe and exist in a changing environment, this a pattern possessing and generating wholeness in one environment, say 1970's America, might not be viewed the same way in another environment, say, twenty-first century Europe. So, while Alexander and Gabriel claim the quality itself is objective the changing environment makes it hard to validate this claim.

This author believes that taken together these arguments render the second and third questions subjective. Who is to be the final arbiter of pattern wholeness? Who is qualified to judge whether a given pattern creates wholeness?

Given these limitations all we may ask a pattern writer is that they aspire to answer *yes* to these three questions.

### 3.3 *Sustainability and, living and dead patterns*

Now we understand the need to aspire to wholeness we need to understand the limits of pattern writing. Obviously some things are simply not patterns: unique events with unique solutions, but what of events that reoccur and produce reoccurring solutions but which do not contribute to wholeness? For example, the smog creating highways that blight cities such as Los Angeles?

Alexander draws a distinction:

“These patterns of events are always interlocked ... The specific patterns out of which a building or a town is made may be alive or dead. To the extent they are alive, they let out inner forces loose, and set us free; but when they are dead, they keep us locked in inner conflict.” (Alexander, 1979, p.x)

Simply, some patterns, dead patterns, perpetuate the conflict. This is not to say all alive patterns resolve all forces, far from it. After the application of a living pattern some forces are resolved, some mitigated, some unresolved and some new ones created. This creates the environment for the next pattern, this is a living process where each solution flows from the next.

Conversely, a dead pattern fails to resolve forces, enhances others and creates new ones in ways that restricts change and resolution. Our ability to create wholeness is reduced.

For example, we might see:

- Architectural pattern describing the creation of slum housing.
- Software pattern describing the creation of a highly coupled monolithic application.
- Business pattern describing Enron style accounting.

We could write patterns about these subjects, but the resulting patterns would not contribute to wholeness, they would be dead patterns.

Things become clearer if we substitute *sustainability* for *wholeness*. Living patterns contribute to the sustainability of architecture, software applications and businesses; conversely, dead patterns reduce sustainability.

So, a business run according to the *Enron pattern language* might well grow, employ thousands of people, increase shareholder value and reward the business leaders but eventually prove to be unsustainable.

### 3.4 *Worse is better: path dependency?*

If our goal is to address all forces and create wholeness we must indeed address all forces: technical, social and economic. In some cases it may simply not be possible to address all forces, in other cases a *good enough* solution may suffice to resolve some of the forces, or at least mitigate those that remain.

Gabriel's *Worse is Better* (-, 1996, 1990) argument suggests that all forces are not equal, the forces of economics and the market can overwhelm technical forces. Similar arguments are made in the pattern *Big Ball of Mud*

(Foote and Yoder, 1999). This describes how shantytowns and monolithic software applications can come to be when the immediate forces of architecture and software design are overwhelmed by the environment we find on the ground.

On the one hand *Big Ball of Mud* cannot contribute to *Wholeness* because it describes how we break every rule of good architecture. On the other hand, for those living in the shantytown or earning a living from maintaining monolithic software it is increasing *Wholeness* - it is better to live in a shantytown than a doorway.

Again, substituting *sustainability* for *wholeness* sheds some light. The monolithic application and the shantytown are sustainable for a period of time but not indefinitely. Eventually the application will fail or become unmaintainable; in case after case shantytowns fail because of natural or social phenomenon such as earthquake, landslides, social breakdown and riots.

One might imagine a situation were a dead pattern, one that does not contribute to wholeness and has limited sustainability becomes the dominant solution to a problem. Such a position would be analogous to the economic theory of *path dependency* and “vendor lock-in”.

According to the path dependency theory the QWERTY keyboard and VHS video recorders achieved market dominance not because they were technically superior but because they were first to market. Following this line of argument, the *Big Ball of Mud* exists because once it has been brought into being the cost of replacing it not justified.

A fuller discussion of this theory and the analogy is beyond the scope of this paper. For the moment it is sufficient to say that we may draw the analogy and that the theory of *path dependency* is disputed (e.g. Margolis and Liebowitz, 1998).

### **3.5 Knowledge management and Patterns**

Patterns are written with the objective of communicating some piece of knowledge from the author to the reader. A long list of pattern publications demonstrate some degree of success here. Actually, a second objective less often mentioned is to allow the author themselves to better understand the pattern.

The idea that Patterns are a form of knowledge management is not new:

“A relatively new approach known as *patterns* has created new possibilities for those who wish to manage knowledge in a more efficient and effective way.” (Manns, 2001)

“the really big opportunity for patterns and pattern languages, an opportunity for exponential growth, comes to light when we perceive patterns as an important part of a larger challenge - that of valuing, exploiting, and managing the knowledge available all around us, in both tacit and explicit forms, ripe for the picking.” (May and Taylor, 2003)

Patterns both embody knowledge and describe its practical use. Each pattern describes a solution to a problem with supporting evidence of use

and applicability. There is an implied chronology to the pattern that gives it a somewhat story like nature, for example:

“we had a problem, there were these forces... therefore, we did X, and as a result we got here. And now we know of some other folks who did similar things.”

There is an emphasis in pattern writing on *what you do* rather than what you know, as such, the pattern is a call to action (Alexander’s events). The pattern narrative calls for the pattern to be used - the search for *wholeness* directs us to create and use patterns that live, patterns that are not only knowledge but action too.

That patterns concern themselves with action as well as knowledge gives them a role in closing the so called *Knowing-Doing Gap* (Pfeffer and Sutton, 2000).

A pattern is a form of structured story. Like poetry the structure may take many forms, indeed, authors are free to devise their own structures. However, there must be a recognisable structure, and it must help express the core elements of a pattern: context, problem, problem forces, solution and solution consequences.

Patterns are not just structured stories. In detailing a set of actions and choices they outline a design. People must consciously choose to build that described in the pattern and modify the pattern to their specific environment. Implementing the pattern is a conscious design oriented action by one or more individuals.

Without the structure, core elements and design imperative a problem-solution pair may occur again and again and in common parlance it may be called *a pattern*, but it is not a pattern in the terms of Alexander, Gabriel and the wider pattern community.

The structured form of the pattern, the emphasis on design and construction, the description of actions, the core elements and tradition of review within the pattern writing community all serve to help capture in depth knowledge. In particular, as May and Taylor point out, it is important to capture the less obvious tacit knowledge within the pattern in addition to the better known explicit knowledge. The structured form challenges us to capture the tacit knowledge.

Patterns in the Alexandrian tradition do not exist in isolation. True, anyone may pick up a book and read some patterns but the creation of patterns is seldom done in isolation. A “community” exists around most writers to review the work, guide the author and socialise the pattern knowledge.

The community helps the written form and more importantly helps authors capture the nuances and intricacies of the pattern, that is, helps the author capture and document much of the tacit knowledge that hides inside the problem-solution.

The pattern community also serves to keep the writer honest: a writer who ignores *wholeness* will eventually be questioned and dead patterns exposed.

### 3.6 Story telling and Patterns

Recently, another form of knowledge management has appeared that also seeks to combined action with knowledge: *Story Telling* (Denning, 2001, Brown et al., 2005).

“Stories are one of the ways knowledge is transmitted, especially social knowledge.” (Prusak in Brown et al., 2005, p.47)

“Once we understand what knowledge is, where it resides, and how knowledge in communicated, we discover that narrative plays an unexpectedly large role.” (Brown et al., 2005, p.53)

Brown continues to describe the importance of capturing *living knowledge* that almost to echo’s Alexander’s call for living patterns:

“There are a lot of ways to capture knowledge that kill it stone dead, and it’s very hard to spread knowledge when it’s dead.” (Brown et al., 2005, p.54)

Reading Denning and Brown’s works it is hard not to draw parallels with the pattern community: values and concerns about change, community, abstraction (or *compression* depending on your point of view) and the importance of practice re-occur.

There are also differences between the two techniques, for example Denning emphasis the role of verbal story telling while patterns emphasis the written form; stories are spread by socialisation where patterns (outside of shepherding and conferences) are spread via books and the Web.

Denning (2001, p.197) provides an appendix describing “Elements for Developing a Springboard Story” and it is interesting to contrast these elements with accepted a pattern norms - Table 1.

Denning’s Characteristic	Denning’s explanation	Parallel in patterns
The explicit story should be relatively brief and textureless	The story can be brief and should have only enough texture and detail for the audience to understand it. ...	Patterns should be short and concise. Patterns should not include needless information.
The story must be intelligible to the specific audience	The audience needs to understand enough about the protagonist and the initial incident for them to be <i>hooked</i> ...	Writers should understand who their audience is. Pattern must pose a real, non-trivial, problem, writers should build tension to be resolved later in the pattern.
The story should be inherently interesting	To capture interest, the <i>actions</i> described might be difficult, with a <i>predicament</i> that cannot be handled in a routine manner, and some tension between	Patterns should describe a real problem with multiple forces creating tension in the problem statement. The solution should be non-obvious, at least at first,

	characters in the story; or <i>unexpected events ...</i>	and should describe implementation action.
The story should spring the listener to a new level of understanding	... it must epitomise or embody the change idea, almost like a premonition of what the future will be like. ...	The reader should learn something from the pattern. By naming the pattern we move to a new level of abstraction.
The story must have a “happy ending”	The story needs to spring the listener out of the typical negative, questioning, skeptical frame of mind ... the listener must [achieve] a positive <i>aha!</i>	The solution is an essential element of the pattern, reading it should give the reader an “ <i>aha!</i> moment”.
The story should embody the change message	Stories can help persuade people to change ... which the audience can discover on its own and make into its own change message.	Patterns do not necessarily embody a change message. The solution may imply some degree of change but often the solution describes a technical fix.
The change message should be implicit	... one lets the listener discover the implicit change message so that it becomes the audience’s own idea and meaningful to them.	Patterns contain <b>explicit</b> messages that should be clearly communicated. Readers are encouraged to relate the pattern to but everything is described clearly.
The listener should be encouraged to identify with the protagonist	Much of the power of the springboard story derives from the fact that the audience identifies with the protagonist ...	No direct parallel. Forces and consequences encourage the writer and reader to consider the other protagonists/stakeholders effected by the problem or solution.
The story should deal with a specific individual or organization	People are more likely to project onto single individuals ...	Examples are often, but not always, used to build the pattern through a single case.
The protagonist should be prototypical of the organization’s main business	The individual should be a central figure in the business of the organization ...	No direct parallel. Patterns are written for a specific audience.
Other things being equal, true is better than invented	Our confidence in the veracity of a supposedly true story is usually not on very solid ground.	Patterns should not be works of theory or conjecture. Patterns should be observed in use, in the real world before documentation.

		Multiple known uses are valued.
Test, test, test	... A story can be tested on individuals or small groups before being tried out on a large group in a high-risk setting.	Patterns are not directly tested themselves but are expected to document known (and therefore tested) solutions.  Shepherding, writers workshops and pre-publication review may provide some degree of pattern testing.

**Table 1 - Parallels and differences between Denning's story elements and Pattern writing**

The approach taken by Denning and others to knowledge management through storytelling seems parallel the approach taken by pattern writers but the two are not the same. Hopefully the two communities will be able to learn from one another.

## 4 The Patterns

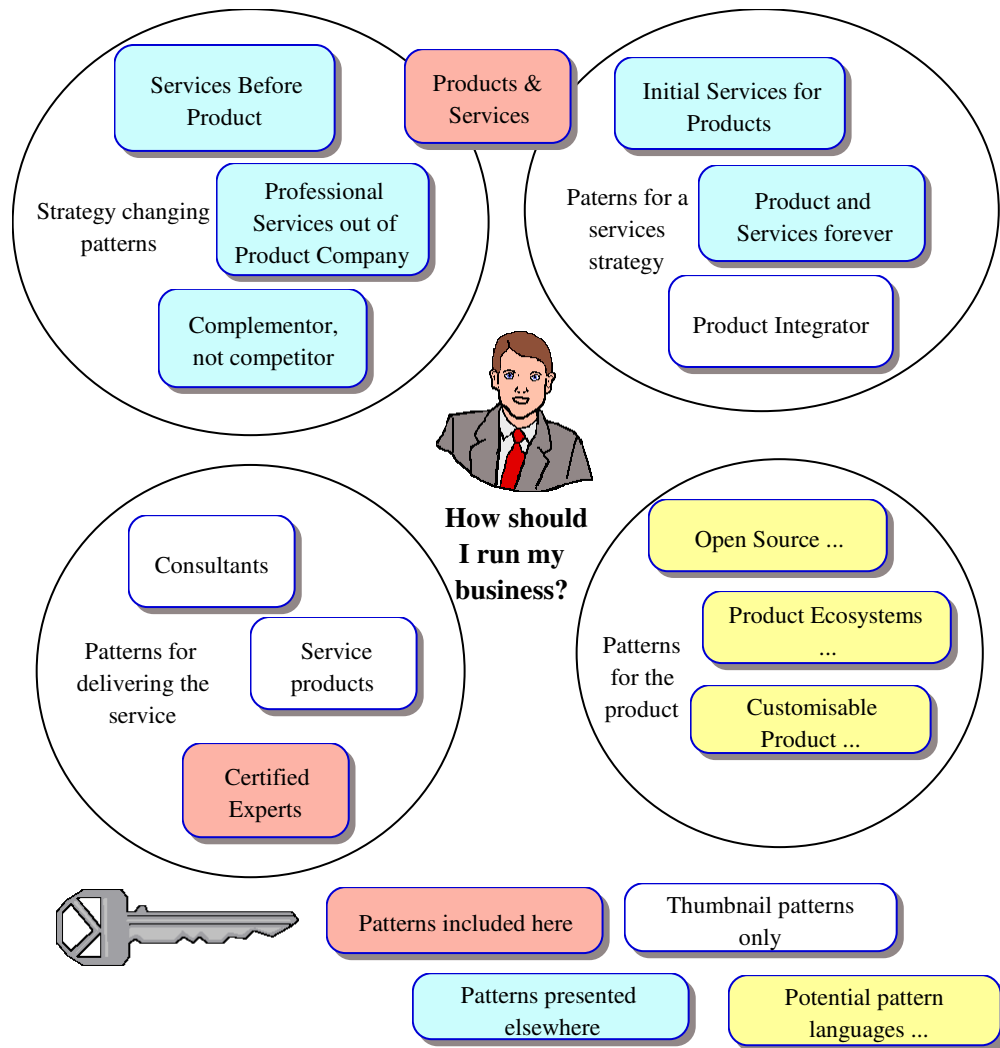


Figure 1 - Pattern map

<i>Product and Services</i> <i>Page 12</i>	Technically complicated products are not commodities, they can be hard to use. Therefore, offer services in addition to the product to help the customers, e.g. a support desk and training courses.
<i>Start-up Services for Products*</i>	Your product serves a complicated market, consequently your product is complicated. Customers need help to get the most from the product. Therefore, create a professional services group within your organization and sell consultancy services to help the introduction of your product.
<i>Continuing Services for Products*</i>	Complex products often require ongoing maintenance and support. The company that makes the product already knows a lot about the product so is well positioned to do this activity too. By sharing

	knowledge between services and products operations both can be improved.
<i>Complementor, Not Competitor*</i>	Choosing to compete in multiple product categories against multiple competitors' means you sometimes compete against companies who could help sell your other products. Therefore, withdraw weaker and less strategic products, you can now complement your former competitors and increase sales of your leading products.
<i>Services Trump Products*</i>	Your company has been successful selling products but you are running out of growth, you may already be loosing money. Therefore, use your knowledge of the products to move up the value chain and sell services instead of or in addition to products.
<i>Services Before Product*</i>	You are creating a start-up company but you are short of money and/or need a better understanding of the market. In order to get a better understanding of the market you need to get into the market. Therefore, sell consultancy services to start with, you will generate money and get a feel for the market before you start work on your product.
<i>Certified Experts Page 15</i>	You and your customers want to know who is competent to work with the product in depth. Your staff can't do all the work. Therefore, segment the user base by offering to certify experts who know the product in depth.

**Figure 2 - Thumbnails of related patterns**

\*Patterns presented at VikingPLoP 2005 (Kelly, 2005a)

## 4.1 Products and Services

*Being a product company is hard. Most companies have support desks and offer some training services.*

<b>Context</b>	You produce a technically advanced product, e.g. enterprise software.
<b>Problem</b>	<b>Sometimes your customers get stuck with your product. They don't know how to use it, they don't know what it expects of them, or it doesn't seem to be working as they expect - it may even have developed a fault.</b> <b>How do you help your customers overcome the barriers to using your product?</b>
<b>Forces</b>	Technically advanced products solve many customer problems but bring problems of its own, e.g. software contains bugs, manuals need reading, customers require training, installation is hard.  Customers may have bought your product but they are not using it to the full. Some users may be unaware of valuable features, others may have encountered problems on first use and not tried again.  You want to sell more product but customers don't see the value in your product because of difficulty using the product. You may miss a sale entirely or be forced to discount the product to make a sale.  You want to improve the product but without customer feedback you will be guessing. What do customers find difficult? What features do they find missing? What features will they need next year?
<b>Therefore...</b>	
<b>Solution</b>	<b>Supplement your product with services to help the customer maximise their value from your product.</b>  In the technology sector even pure product companies will find the need to provide technical support for their products - this is the most basic service to be offered. This may be offered free for an introductory period, say, the first year, after which customers may be expected to pay either a fixed sum - perhaps 10% of list price - or pay per call.  Training services can help users get more from your product. New users can be helped along the initial learning curve and more experienced users can benefit from learning new and advanced features.  For some products, in some markets, support and training may be the only services required. With other products, in other markets it can help to provide pre-sales services to assist the sales force.  Post-sales services can take many forms: onsite consultants helping customer staff day-to-day or occasional visits to hear about problems and observe how the product is being used.

Other services may be offered during initial adoption of the product and throughout the life-time of usage - see related patterns below for more details.

**Consequences** Providing a support desk will allow customers to make you aware of their difficulties. Even simple queries (“How do I do...”) can provide valuable information on how the product is used and perceived. More complicated problems (“It crashes when I do...”) may demand an immediate fix and highlight issues for future products.

Support and other services all help you understand your customer in more depth and extend your knowledge of the domain.

Sending trainers and consultants to visit customers will allow you to identify and fix problems with product use. This might be as simple as helping a user with a difficult action or simplifying features in a future release.

Resolving customer problems will allow you to demonstrate usage and value. Increasing the product value to the customer will increase the chances of further orders.

By having your people engage with customers in service roles you can learn what customers find difficult, what features the product lacks and what future opportunities exist for your product(s).

You can charge for support, training and other services. This provides for an alternative revenue stream. However, services can be expensive to provide, support desks must be staffed even when there are no calls. A low quality product may result in more calls and thus more expense.

For a company that considers its core competency to be in creating technical products it may be hard to justify support services and to position them in the organization.

Training can be a substitute for making an easy to use product. Companies that rely on training users rather than making simpler products are vulnerable to competitors with such products.

**Variations** If you are not able to provide the services yourself you may choose to partner with an organization that can provide some, or all, of the services. You must ensure a feedback mechanism is in place so you hear about customer problems and needs.

**Examples** Most software companies provide help desk support for their product range that will answer technical questions and help users with difficulties.

**Also known as**

**Related patterns**

Kelly (2005a) gives several related patterns that may follow:

*Start-up Services for Products* describes how services can be provided to help hold a customer's hand during initial stages of

adoption.

*Continuing Services for Product* describes how services can continue to be offered for a product after this initial adoption.

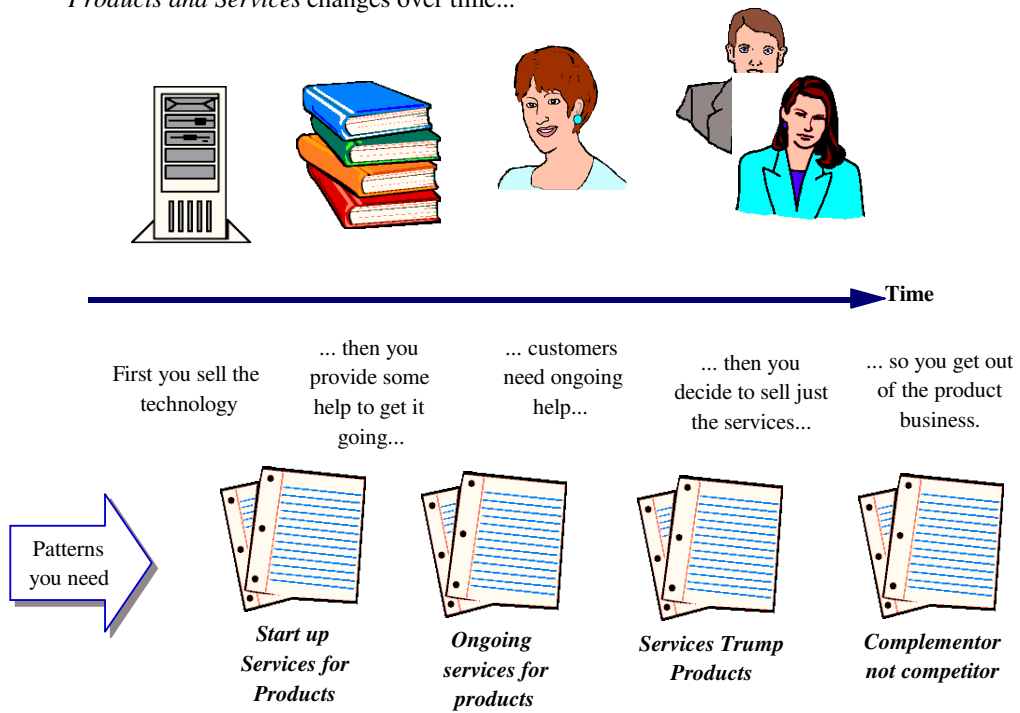
Sometimes it is more profitable to supply the services than products, *Services Trump Products* describes how companies change. This may involve using *Complementor, Not Competitor*.

This sequence is shown graphically in Figure 3.

**Sources**

Authors observations and experience

*Products and Services changes over time...*



**Figure 3 - Products and Services is the root of several other patterns**

## 4.2 Certified Experts

*During the late 1980's and early 1990's Novell Netware was the most successful networking system. But it was difficult to install, configure and administer. So, Novell created the Certified Novell Engineer programme. If you needed a Novell network you hired a CNE to look after it. Individuals found they were worth more if they were certified and service organizations added CNE services to their offerings.*

**Context** You are marketing a technically complicated product that requires a significant level of understanding to operate correctly.

**Problem** **How do you and your customers know who is proficient to use your products?**

**Forces** The product may expose several different interfaces but at least one requires in-depth knowledge, e.g. vending machine appear simple to casual users but very different to service technicians who repair machines.

You need to be able to distinguish between expert and non-expert users of your product.

Documentation and interface designed for advanced users will not help basic users, but advanced users have little or no need of the documentation and features needed by the basic users.

Your service organization can undertake advanced work (e.g. installation and configuration) for customers but this makes customers dependent on your service organization.

You want to concentrate on products but the need for services dilutes your focus.

Some organizations may not want to be dependent on your service organization, and if your product is successful your service organization may not be able to service all potential customers - resulting in lost sales.

Users outside of your service organization may be capable of performing the necessary work but how do you, or your customers, know who has the advanced skills and experience necessary and who doesn't?

**Therefore...**

**Solution** **Segment your users base by creating a certification programme for advanced users.** You will be able to target documentation, training and features at different levels of user, your customers will know who is capable of undertaking work on your product.

You will need to create a test for those seeking certification. You will need produce materials for those studying for the test and ensure those certified are up to date with changes, e.g. certification might be tied to a product version, or re-certification may be required every

year.

It may be necessary to create multiple certification programmes for different products or different aspects of the product, e.g. Novell certifies Netware Administrators, Netware Engineers and SUSE Linux Professionals. You may also want to create certification for trainers who can train others in your products.

**Consequences** By distinguishing between users you can target different interfaces and materials (e.g. documentation and training) at different levels of users.

When material is clearly marked for different customers then people will be able to choose for themselves or you can restrict access to qualified people only.

Your service organization is no longer the only people who can undertake complex work. Customers can now choose your service organization, a third party organization or train their own people and do the work themselves.

Even when your service organization is fully utilised sales can be made to customers who can source the necessary services elsewhere.

Your company can focus on its products again rather than supplying services.

Third party organizations may decide to add your product(s) to their service portfolio. This may result in reduced revenues for your service organization but increased sales of your product.

Individuals may find that certification makes them more employable and allows them to command higher wages.

The existence of individuals and organizations who can support your product will enhance your market credibility and make your product a less risky purchasing decision thereby increasing sales.

## **Variations**

**Examples** Novell's Netware certification programme helped Netware become the leading network operating system in the late-1980's and early 1990's. After purchasing SUSE Novell expanded the scheme to provide Linux certification.

Microsoft and other enterprise software providers offer similar schemes for their advanced products and technologies.

A technology certification programme is akin to the Driving Test that most countries require drivers to pass. While anyone can be a passenger in a car only certified drivers can take the controls.

**Also known as**

**Related patterns**

**Sources** Authors observations and experience

## ***Acknowledgements***

Thank you to Klaus Marquardt for shepherding this paper to EuroPLoP 2006. Klaus was good enough to shepherd my VikingPLoP 2005 submission, why he came back for more I don't know!

## ***History***

June 2006	Post shepherding version submitted for EuroPLoP
May 2006	Interim version posted on web
February 2006	Submitted to EuroPLoP 2006, shepherding begins.
January 2006	First draft

## ***Bibliography***

- Alexander, C. 1979 *The Timeless Way of Building*, Oxford University Press, New York.
- Alexander, C. 2002 *The nature of order : an essay on the art of building and the nature of the universe*, Center for Environmental Structure, Berkeley.
- Alexander, C., et al. 1977 *A pattern language*, Oxford University Press.
- Brown, J. S., Denning, S., Katalina, G. and Prusak, L. 2005 *Storytelling in Organizations*, Elsevier Butterworth-Heinemann.
- Denning, S. 2001 *The Springboard: How storytelling Ignites Action in Knowledge-Era Organizations*, Butterworth-Heinemann.
- Foote, B. and Yoder, J. 1999 *Pattern Languages of Program Design volume 4* (Eds, Harrison, N., Foote, B. and Rohnert, H.) Addison Wesley.
- Gabriel, R. P., 1990, *Worse is Better*, EuroPAL, Cambridge,
- Gabriel, R. P. 1996 *Patterns of Software: Tales from the Software Community*, Oxford University Press.
- Gabriel, R. P. - <http://www.dreamsongs.com/WorseIsBetter.html>,
- Kelly, A., 2005a, *Business strategy patterns for knowledge based products and services*, VikingPLoP 2005, Espoo, Finland,
- Kelly, A., 2005b, *A few more business patterns*, EuroPLoP 2005, Irsee, Germany,
- Manns, M. L., 2001, *Patterns: A Promising Approach to Knowledge Management*, ABIT (Association of Business and Information Technology), Monroeville, Pennsylvania,  
<http://www.eberly.iup.edu/ABIT/proceedings%5CPatternsAPromisingApproach.pdf>.
- Margolis, S. E. and Liebowitz, S. J. 1998 *Look, I understand too little too late*, <http://www.utdallas.edu/~liebowit/palgrave/palpd.html>,
- May, D. and Taylor, P. 2003 Knowledge Management with Patterns, *Communications of the ACM*, 46, <http://www.thedanielmay.com/publications/p94-may.pdf>.

Pfeffer, J. and Sutton, R. 2000 *The Knowing-Doing Gap*, Harvard Business School Press.