International Journal of Technology and Innovation Management Education 1(1): 57-68.

A Pattern Language for the Teaching and Practice of Technology Management

Andres Fortino
School of Management, Marist College

Abstract: The work of architect Christopher Alexander in articulating the field of pattern languages for architecture has been extended to several other fields, notably object oriented software development and project management. Alexander calls for the identification of patterns in areas of creativity (e.g. architecture, programming) and of human activity (e.g. project management) which when taken together form a comprehensive and interrelated set of entities comprising a language that defines best practices in that area. Alexander has identified a well-defined process to identify patterns and criteria for selecting and creating patterns beneficial for action by the practitioner. Most often the discovery of patterns and development of a language for a field of practice is carried out by a community of practitioners. This paper presents the application of Alexander's pattern language approach to teaching technology management practice. Specifically it proposes the development of a community of technology managers, practitioners and educators to discover patterns and create a pattern language for technology management. Examples of patterns in technology management and their derivation from industry laws (such as Moore's Law) are presented as a staring point in the discussion.

Keywords: patterns, pattern language, best practices, teaching, IT laws, community of practice.

1. Introduction

Technology management as a business endeavor has become a powerful business asset. It has developed rapidly and has gained acceptance. More importantly, business executives have elevated the importance of technology management competencies in new hires to a premium. A recent survey of executives of major corporations in the Philadelphia was conducted by Villanova University business school (Najdawi, 2003). One gratifying result is that these executives ranked competency in technology management as most important in graduates five years after graduation, behind number-one ranked competencies in international business.

With such a premium on technology management competencies, it is useful to consider ways to contribute to the building of a body of knowledge in technology management best practices as well as providing technology management students with the opportunity to develop competencies in these best practices.

The creation of a pattern language for technology management best practices may be such an opportunity. The competency in discovering patterns in technology management practice is complementary to language creation and one

that would be useful to teach technology management students. It is advocated here that the development of a community of practice (Brown, 2000) of technology management practitioners be created to discover patterns in best practices, document them and make these available as a body of knowledge.

The pattern language techniques of Christopher Alexander (1979) are a fundamental basis for such an endeavor. This process has been successfully translated from architectures of buildings to architectures of computers (Coplien, 1999). Best practices in managing enterprise architectures would yield well to this documentation approach as well.

The Alexandrine process is also particularly useful in developing analysis skills in technology management students. They in turn, through their educational efforts, discover and contribute patterns to the community of practice while building competencies in industry analysis and best practices.

This paper develops the process of crafting patterns and creating a pattern language. Then it explores its application to pedagogy showing how this process was applied successfully to the education of technology managers at one university. Finally it proposes the creation of community of practice of technology managers for the purpose documenting technology management best practices in a pattern language, develop a database of patterns and thereby document the growing body of knowledge in the field.

2. Building Quality Architectures

2.1. Patterns as Rules of Architecture

Christopher Alexander (1979) argues that architectural design rules may be codified through use of pattern languages. He calls the process the "the timeless way of building" (Alexander, 1977). He urges architects to build buildings that are alive: good to live in and to work in. This quality of aliveness or livability as apposed to "dead" or difficult to live in he says is hard to quantify, even to define. He calls it the quality without a name. But that quality can be recognized when it is present. Living architectures, that evolve gracefully, that serve people, that "feel" good have that quality.

Alexander goes on to demonstrate that the gateway to building with the quality without a name is the use of patterns. Patterns are fundamental dimensions of architecture. These patterns can be sought, discovered, documented and used by other workers. And one of Alexander's major contributions is teaching how to discover and document patterns (Alexander, 1979, 2003).

The process may be applied equally well to other areas of technologies, especially ones that are organized in architectures. Information technology is one

such area, and technology managers as information technology architects can derive some benefit from using the Alexander techniques.

3. Patterns

Patten documentation has taken several forms. The method used below follows the format originally proposed by Alexander (1979), and the resulting patterns are termed Alexandrine patterns. Other methods are shorter and more succinct (Coplien, 1997) or more suitable to specific applications such as software development (Coplien, 1999). It may well be that as workers in the field of technology management adopt this method of documenting best practices; the form of the pattern evolves to a more convenient embodiment. The Alexandrine form is used here as a starting point of discussion.

3.1. Elements of a Pattern

Name of Pattern. Usually several descriptive words in title case written as one word.

Context. Background material that sets the stage for understanding the forces and problem. The statement of the context of the problem space and the pattern that follows.

Problem. The problem solved by the pattern.

Forces. The architectural forces in tension behind the pattern.

Solution. The description of the pattern that resolves the tension in the forces.

Resulting Context. The specific context the pattern will operate in, as there may be other patterns to solve this problem with other parallel contexts.

Examples. Some specific instantiations or applications of the pattern and how it resolves forces by solving the problem. Real-life examples.

Known Uses. General class of problems the pattern can solve.

Pattern Connections. Other patterns in the pattern language directly associated with this pattern.

4. Sources of Technology Management Patterns

Patterns are derived from observation of industry forces in play which eventually create problems. They in turn may find recurrent solutions. Industry laws are a fruitful source of well known forces at play with known solutions. They are the wisdom of the elders of the industry and have accumulated over the years. It is useful in the education of technology managers to turn their attention to both documenting the laws, which in and of itself is an excellent educational activity, as well as to discerning and documenting patterns derived from the laws just documented.

4.1. Laws of the IT Universe

In any profession and area of scientific inquiry there appears to accumulate over the years a collection of practical wisdom, a distillation of the pronouncements of the elders of the tribe. In the IT industry these rules seem to be codified in "laws" that seemingly spring up industry-wide. They contain hard won truths and best practices, as well as insightful observations by foremost practitioners, which could be of much use to fellow practitioners as well as to those learning the profession. A recent IEEE Spectrum article documented five of the most common laws (Ross, 2003).

These IT laws, over thirty-tree at last count (Fortino, 2002), cover the entire spectrum from business practices (Barksdale's Effect) to scientific principles (Snell's Law). Some are well-known (Moore's Law) and some are more obscure (Kerckhoff's Criterion). They are laws to govern software development (Brooks' Law); and to give us rules of thumb for the cost of large systems (Grosch's Law) and smaller systems (Machrone's Law). Some are obsolete while others are still applicable, albeit with modifications. Some are timeless, such as the Law of Unintended Consequences. Others are the basis of entire segments of the industry (fiber optics is based on Snell's law of the physics of total internal reflection); while still others predict the future direction of the industry (Nielson's and Moore's).

What makes a law? Some are incontrovertible physical principles and can rightly be called laws (Nyquist's and Snell's). Most are insightful observations by prominent industry observers that, through the test of time, have proved to be relatively true and are accepted by all in the industry (Moore on chip making), with many imitators in related areas (Nielsen on internet bandwidth). Many are industry trend curves predicting the future (today called road mapping), and have been reduced to mathematical models (Bass Curves for diffusion of innovations). In all instances, two factors elevate these observations to the levels of "laws": (1) the test of time and (2) the adoption of the principle by the majority of workers in

the industry, most of them recognizing it as a law and writing and speaking of it as such.

Why study these laws at all? They provide rules of the road for technology manages and executives in the IT industry to properly and safely acquire or build and deploy IT solutions for their organizations. If managers drive safely by paying attention to these laws they stand a good chance of being successful. Break these laws and risk failure. The laws also embody the practical wisdom necessary to craft patterns. If patterns are derived by observing problems in action and the resulting solutions to these problems, what better place to look for patterns than in the collected wisdom of those who have made deep and often timeless observations of the workings of our information architectures.

5. Pattern Discovery and Documentation

Patterns are discovered, not invented. They are evident solutions, tried and true, that are gleaned from long-standing practice. They are not clever inventions that solve a problem but solutions that stand the test of time. Even in dealing with innovative twenty-first century technologies, one may manage them successfully using patterns based on centuries-old practices. The faculties that work best in discovering patterns are keen powers of observation and of discrimination. Observation reveals the underlying pattern and discrimination allows for selecting life-giving (or beneficial) patterns that advantage the enterprise architecture.

5.1. Process to Discover a Pattern

Let's try to observe an architecture, a building, or a town as a pattern maker would. The pattern maker looks to discover any recurring themes, patterns, dimensions, "things" that work, "spaces" that makes a person feel good being in them, while using that space. They ask: What is fundamental about that "thing"? Why does it work? They extrapolate to other things that produce similar results and life-like feelings to discover the fundamental pattern in the group of things being observed.

Take for example a fence. Fences solve the fundamental problem of division of space. They help keep the peace between neighbors, mark legal boundaries of land, keep livestock in the proper side of ranches. These forces are in tension, the pattern represented by the prototypical fence would resolve them very neatly and efficiently. Thus architectures that are alive, that have that quality of goodness necessary for life to go on, use fences. Fences come in many sizes and shapes, and have many applications, but they, as a unit, constitute a fundamental pattern in buildings that are alive.

It is the same with firewalls in information architectures, for example. Their use constitutes a pattern of security used to separate networks. Firewalls are a pattern in data networks. They resolve many competing forces. They come in many shapes and sizes, many different embodiments. And there are many known uses for firewalls. It is one architectural element that may be rightly classified as a pattern.

It is easy to see that a firewall is a pattern because it is physical, but what about intangible patterns? Software and its development is one such intangible which may be analyzed and worked into a pattern language. Our colleagues designing object oriented (OO) programs (Gamma, 1995; Coplien, 1996) have pioneered the use of patterns in successful OO design projects.

This technique may be extended to business practices as well. Project management, business process reengineering and the creation of best practices compendiums in some business disciplines have yielded to this approach (Coplien, 2000). For the technology manager, all aspects of the enterprise architecture, its infrastructure, both hardware and software, as well as operational processes dealing with the architecture may be analyzed and benefit from the pattern analysis and the development of a pattern language.

The technology manager would be urged to look at architectural elements in enterprise architectures, especially in the technical infrastructure components. What are the current recurring themes of IT application (everyone seems to have websites these days for every information need, for example); what patterns, dimensions, and/or things are observed that make an enterprise architecture work well, that makes information flow better, and make business processes operate more efficiently and effectively? These observations would be extrapolated to analyze similar circumstances and things that work well. What problem is being solved, what are the competing forces being resolved, what general pattern could be delivered from this set of common solutions to a common problem? Analyzing the solution to these problems would lead to discovering patterns.

5.2. Crafting a Pattern

The first step in crafting a pattern (as a literary creation documenting the discovery) is to observe that there is a problem to be solved. Then, after some analysis and investigation and observation, all the forces in tension behind the problem are discovered and documented. These become the givens with the hard work documenting the pattern to follow. The next step is one of design – the forces need resolution. There must be a solution to the forces in tension that solves the problem they bring about. This is the imaginative step. The practitioner observes what solutions have been crafted by other practitioners, indeed what is the best practice solution to the group of forces creating the problem. The observed solution is expressed as a pattern of action that embodies the solution in

as general a way as possible: "Under these circumstances it is recommended the practitioner should do this." The discovered pattern resolves the tension in the forces.

The rest is documenting the resulting context of the pattern with some example applications given to show how it is instantiated. Some known uses, which may have been the specific cases from which the solution was originally surmised, are also sought and documented to assist other practitioners in applying the pattern. Lastly it may be observed that the pattern is linked to other patterns. These are then given as pattern connections.

Naming the pattern is the last step, as with any title. The name selected uses a catchy one or two word (and occasionally three) sequence that pithily describe the pattern. The component words of the pattern name are usually written together as one word using title case format.

5.3. Process to Craft a Pattern Based on an IT Law

One useful approach to discovering patterns is for the practitioner to start out by identifying an IT architectural area of application. Then the IT laws applicable to that area of interest are studied. The technical manager should try to determine the *context* of the law. What *problem* does it appear that the law addresses? If the law is a straight forward statement of a physical phenomena (such as Snell's Law) one should look for applications where the law solves IT architectural problems. For example, Snell's law is fundamental to the development of fiber optics, which in turns solves many communication (wiring) problems, each being a possible basis for best practices based on application of fiber: security, long distance/low loss transmission, in high noise environments.

The IT law, or its application, is concerned with *problems* resulting from the tension of *forces* in the industry or in the work environment, such as Brook's Law, for example (Brooks, 1995). The next step would be to analyze and document the forces behind the problems the law addresses. These forces have often been resolved via a best practice or a design solution (hardware or software), and the technology manager may observe these best practices as specific embodiments of a pattern.

Collecting several of these *solutions* to the *problem,* and which also resolve the *forces* in tension, is the precursor step to discerning a *pattern*. The technology manager is not so much designing a solution but observing solutions that have already been crafted and that work–best practices. Here is where the power of discrimination comes in. Some solutions are better than others, more successful, more efficient and effective. They give life to our system making the technology manager more successful in managing the architecture. Collecting these solutions and observing that they are a *pattern* of best practice is the critical point

in the analysis. This is the nugget of truth the technology management is seeking. From here the analysis turns into documentation.

The *pattern* is expressed as a single *solution* in the form of a best practice and it is also expressed in the pattern notation of Alexander. The *resulting context* is documented together with some specific e*xamples* of where the pattern has been applied successfully. It is further generalized by describing as many known uses of the pattern as possible. When constructing a set of related patterns, as in a language, then other *pattern connections* may be noted.

5.4. Example Use of an IT Law to Create a Pattern

IT laws are a very good source of practical knowledge from which to glean patterns. The laws that deal with IT architectural issues have a very good correlation to architectural patterns in civil engineering, as first applied by Christopher Alexander. Consider creating a pattern from the well known Moore's Law.

Moore's Law

> The observation by Gordon Moore of Intel that the logic density of silicon integrated circuits has closely followed the curve *(bits per square inch) = 2(t - 1962)* where *t* is time in years; that is, the amount of information storable on a given amount of silicon has roughly doubled every year since the technology was invented. (Schaller, 1996)

Pattern. PeriodicUpgrades

Context. CIOs and IT managers are constantly challenged to keep their fleet of computing platforms (servers and workstations) upgraded and are often under tight budgetary constraints.

Problem. Reduce risk of workstation obsolescence (and that of other computing platforms such as servers) while spending the least possible amount to maintain worker productivity

Software manufacturers take continuous advantage of the advances in chip capability by improving their software (Gate's Law, Fortino, 2000). User workstations should get upgraded on a periodic basis and at the correct interval, to keep workers productive, making the firm more competitive. If the CIO waits too long to replace the fleet of workstation the cost to upgrade the entire fleet is a major expense that could hurt the firm financially at the time of upgrade.

Competing Forces. The law shows that chip manufacturers relentlessly drive to make CPU chips denser and faster and memory chips denser with greater capacity. Competition in the industry drives their cost down at the same time as their capabilities increase. When the new chips first appear on the market new memory and CPUs technology is normally too expensive to roll out for all user workstations – bleeding edge.

Solution as a Best Practice. Replace one-third of the fleet of workstations every twelve months with machines that have memory and CPU chips a generation older than the bleeding edge technology. Use Moore's law to estimate the proper rate of replacement to align with the pace of change in chip technology. Servers are excellent candidates to be replaced by the newly released high-performing machines sporting the new chipset.

Resulting Context. Managing technology churn such that the risk is minimized and the organization remains efficient and competitive at the minimum cost.

Examples. A medium-size company may have anywhere from 250-500 workstations and a dozen servers. A new version of the Pentium chip is slated to be released by Intel. Three months later this chip is available in workstations. At this point it is too expensive for all but the most demanding of power users. The appearance of this chip in workstations considerably reduces the price of less powerful machines. And these are the models that should be considered for purchase to replace a part of the fleet. The just-released chip may be at the point of purchase for fleet replacement within twelve months or more depending on how aggressive the CIO decides to be.

Upgrading one-third of the fleet (or may be 25%) on a rotating basis (every twelve months or more) keeps the upgrade process simple and the fleet never more than three years old. The same may be done for servers except servers are candidates to be replaced with machines sporting the new chip.

Known Uses. Server replacement, workstation replacement, laptop replacement.

Pattern Connections. Rock's Law (Brix, 1998).

6. Community of Practice

Coplien demonstrates (1999) that pattern writing is a literary process. It is best done under similar circumstances as the documenting of knowledge creation: the peer review process of publishing. The process is even more public and interactive, with patterns offered for review at writer's workshops, including

public critique and rewrite as an interactive activity. Coplien has derived a pattern language for pattern documentation (1999).

The process has strong similarity and many elements of a community of practice advocated by John Seely Brown (2000). A community of pattern writers composed of experienced technology managers would create a starting pattern database to which other practitioners could add over time. In turn this would become a pattern language and a compilation of a body of knowledge of best practices i9n technology management.

7. Educating the Technology Manager

Pattern writing as an analytic activity used to document best practices in technology management is also a pedagogical tool. As part of a class in technology management the student would receive some training in the theory and practice of pattern writing. Selected required readings in pattern languages would immerse the student in the field, together with an additional rich bibliography for those who wish to explore further.

Assignments may involve the creation of patterns. As a starting point, the student is directed to the laws of the IT industry as a rich source of existing patterns. All students in the class would be assigned to create a pattern from the same law at first. There should be as many examples available as possible for the student to emulate and do a good job on their first attempt at pattern writing. The crafted patterns would be brought back to class for discussion in a writer's workshop atmosphere. The educator would be the expert, guiding and helping students develop competence in analysis and documentation.

The second assignment would be the creation of a pattern based on another law. This time each student selects a different law, and they bring back to the class a rich set of documented best practices based on many aspects of IT architecture practice. The best patterns are then published on a virtual repository available to all students as a working database of best practice patterns. Experience at our university with this process shows the students are motivated to excel in pattern writing as much by the fact that the best patterns are selected for inclusion in the database as with the grade they get on the assignment.

8. Conclusions

Architectural pattern building appears to be a productive metaphor to codify a best practices body of knowledge has found fertile application in many areas of technology. The most developed application has been in software development in general, and object oriented design in particular. It could yield equally good

results if applied to documenting and codifying technology management best practices.

In particular it is proposed that a community of practice of technology managers may undertake as a virtual community to create patterns for technology management best practice. Enterprise architectures offer an excellent field of study for such a purpose. Patterns based on laws of the IT industry may be mined for possible patterns.

Technology management graduate students may be taught how to analyze existing architectural situations using the pattern language framework. They should be challenged to seek patterns of best practice that are fundamental and solve problems in a timeless way. The pedagogical effort may yield a rich set of patterns for use by the community of practitioners.

References:

Alexander, C. (1979), *The Timeless Way of Building*, Oxford University Press, Oxford.

Alexander, C. (1977), *A Pattern Language*, Oxford University Press, Oxford.

Alexander, C. (2003), *The Phenomenon of Life: The Nature of Order, Book 1.* Center For Environmental Structure, Oxford University Press, Oxford.

Brix, T. (1998), "Retrofit, Don't Replace, Your Semiconductor Equipment", *Semiconductor FabTech, 9th Ed.,* pg. 57.

Brooks, F.P. (1995), *The Mythical Man Month (Anniversary ed.).* Addison-Wesley, Reading, MA.

Brown. J.S. and Duguid, P. (2000). *The Social Life of Information*. Harvard Business School Press, Boston, MA.

Coplien, J.O. (1996), *Software Patterns*, SIGS Books & Multimedia, New York.

Coplien, J.O. (1997), "Idioms and Patterns as Architectural Literature", *IEEE Software Special Issue on Objects, Patterns, and Architectures.*

Coplien, J.O. (1999), "A Pattern Language for Writers' Workshops, Bell Laboratories", Retrieved form the World Wide Web January 10, 2004, http://www1.bell-labs.com/user/cope/Patterns/WritersWorkshops/

Coplien, J.O. & Devos, M. (2000), "Architecture as a Metaphor", OT2000 Conference.

Fortino, A.G, (2002), *Laws of the IT Industry*, ISBN: 1586924095, Courier Custom Publishing, XanEdu Publishers, Ann Arbor, MI.

Gamma, E., et al.(1995), *Design Patterns: Elements of Reusable Object-Oriented Software*.: Addison-Wesley, Reading, MA.

Najdawi, M.K. (2003), Presentation at the *AACSB Associate Deans/Data Management Conference,* Orlando, FL.

Ross. P.E., (2003), "5 Commandments. Some technology laws and rules of thumb have stood the test of time, but not all", *Spectrum, IEEE , Volume: 40 , Issue: 12 , Dec. 2003 pp:30 – 35.*

Schaller, Robert (1996), "The Origin, nature, and Implications of "Moore's Law", Retrieved from the World Wide Web", January 10, 2004. http://www.iso.gmul.edu/~rschaller/moorelaw.htm