# Patterns as Structure, Process, and Community

Mary Lynn Manns, University of North Carolina Asheville
Joseph W. Yoder, The Refactory, Inc. – USA

*The modern patterns movement, with its roots in the work of Christopher Alexander, has made impressive strides since the 1990s. Most of this work has focused on the creation of patterns as structures for documenting knowledge. This paper argues that two other aspects of patterns—process and community—have received less attention. However, the time is prime to increase the discussion of how the patterns community can put more emphasis on process and the creation of wider communities. The goal of this paper is to propose a manifesto for the patterns community to make better use of all three features in patterns: structure, process, and community.*

## Categories and Subject Descriptors
**• Software and its engineering ~Patterns • Social and professional topics ~ Quality**
**• Human-centered computing ~ Collaborative and social computing**
• *Software and its engineering ~ Software design tradeoffs* • Software and its engineering ~ Design patterns

## General Terms
Human Factors, Design, Community, Languages, Patterns, Architecture, Society

## Additional Keywords and Phrases
Patterns, Pattern Literacy, Complex Systems, Pattern Language, Societal Changes, Quality, Software Quality

## ACM Reference Format:
Manns, M.L., Yoder, J.W., 2017. Patterns as Structure, Process and Community. HILLSIDE Proc. of Conf. on Pattern Lang. of Prog. 24 (October 2017), 22 pages.

Author's email address: manns@unca.edu, joe@refactory.com

.

## Introduction

The concept of patterns draws inspiration from Christopher Alexander. Recognized as an influential building and urban planning architect [Sali00], Alexander used patterns to document successful design practices in architecture. His focus on proven solutions rather than new and unique ones was motivated by his belief that modern day buildings and towns do not approach the beauty of the historical past. He observed that the vast majority of architecture since the end of World War II has been dehumanizing, of poor quality, and lacking all sense of beauty and human feeling [OCal01]. This created his distaste for simply fashionable architecture and a preoccupation with the search for a design approach that generates beautiful structures [Grab83].

His life mission to make architecture as emotionally rich as the people who live in it has been guided by the belief that design must come from ordinary experience [Brow00]. His patterns describe "the obvious" which, he observes, is usually ignored because people are so often caught up in fashion and trends. In architecture, new and unique work is often rewarded even though it is not comfortable to reside in. Therefore, he argues for the "one timeless way of building" and use of "patterns" for capturing this timeless way [Alex79]. In his *Timeless Way of Building* book, he introduced the concept of the "quality without a name" (QWAN), and argued that we should seek to include this nameless quality in things we build. In his effort to add beauty and life, much of his work highlighted pattern languages, focusing on wholeness and centers, that are created with an unfolding process that can be found throughout nature.

This process begins with describing each individual pattern as "a thing, which happens in the world, and the rule which tells us how to create that thing, and when we must create it" [Alex79]. While each pattern captures a timeless practice, this is not enough. Alexander argues for the importance of a timeless processes for how the individual patterns are structured into a pattern language, for how they are used to create things, and just as importantly, for how the individual patterns and for how the languages are kept alive by the community who uses them.

The concepts proposed by Alexander have caught the eye of others who are attempting to use patterns to document successful practices in other disciplines, such as software development, organizational culture, community development, beauty, health, and more. However, these modern day authors primarily focus on the structure or the thing that is the end result of applying the patterns. While we are good at recording the process that creates the thing each pattern describes, the work often stops here. Individual patterns are usually written, workshopped, published, and then remain static in their current form. They are frequently not combined into a pattern language and into a system for using them. In addition, the patterns, and any pattern languages if they exist, are often not curated and kept "alive" by the community who owns them. Therefore, the authors of this paper propose taking the work of the existing patterns community to the next level in order to achieve all the benefits of patterns -- we must also build the language connections between individual patterns, the process for building things with the patterns, as well as the process for keeping the patterns alive.

There has been some discussion about the process aspect of patterns and some noteworthy attempts to create community through the PLoP conferences. However, in order to advance the patterns movement forward, it's time to pay more attention to process and community. We will describe all three qualities (structure, process, and community), with the intention of opening a discussion on how we can build communities that can both create and make use of patterns as evolving living entities.

## The Foundation for Patterns

We begin by summarizing the history of Alexander's philosophy of construction in order to examine its significance to patterns. Alexander graduated from Cambridge University where he studied mathematics and architecture, and later received a PhD in architecture from Harvard University. He has published more than one hundred books, papers, and monographs, three of which are well known in the software patterns community, *The Timeless Way of Building*, *A Pattern Language*, and *The Oregon Experiment*. Most recently, in his four volume Nature of Order series Alexander has explored other aspects of patterns [Alex02].

The roots of Alexander's patterns philosophy can be found in his earlier publication, *Notes on the Synthesis of Form* [Alex64]. It presents a critique of modern design, contrasting what he referred to as the failure of the professional, rational "self-conscious" design process with an approach which he calls an "unselfconscious" design process. He explained that modern design is distinguished from traditional craftsmanship by its "self-conscious" separation of design from the final product[1] and its reliance on rules and formal models to produce abstract designs. While facing the modern conditions of increasing complexity and accelerating change, society has specialized and spun off design into a separate profession. Alexander argues that because the cognitive burden of highly complex design is great, it is counterproductive to place the responsibility of dealing with all of the multiplicity and changeability of forces impacting a project on the shoulders of a single individual ('designer' or 'architect'). Instead, he argues for embedding design in a highly social process.

This social process is possible because the knowledge of how to build, and therefore to design, is embedded in culture and tradition. Alexander gives modern examples of the huts built by the Mousgoum tribe in Cameroon and the igloos of Eskimos. Traditions and cultures dictate how each of these kinds of structures are built. Those who live in these kinds of houses are the experts in building them. When a design failure is experienced, for example by a river flooding a Mousgoum village, or when changing temperatures require igloos to be ventilated or blocked, the same design culture which dictated how the structure should be built also determines how they can be modified. There are no architect specialists in these societies. There is no separate theory of design. Instead, there is praxis, the result of many years of accumulated experience in building structures that has infused the design culture. In these societies, adaptation is relatively easy because the design failures, which require changes to be made, tend to happen one at a time and are typically familiar. Because adaptation is part of the culture, rather than a separate profession, the feedback loop is immediate. The dweller often knows best what he needs and makes the change.

[1] This can be considered analogous to the Waterfall Methodology

In contrast to this social process that continually reestablishes equilibrium between form and context, failures in modern society are often caused by multiple forces, resulting in repair that is highly complex and requires a design professional to be called. Change is experienced as crisis rather than a natural part of the evolving design process. Contrasting the two design alternatives, Alexander claims that successfully designed products or systems need to be homeostatic, that is self-adjusting. This is the quality that he argues is often absent in modern structures.

As an example, consider an individual tree as an homeostatic system. It presents a form that is optimally fitted to its environmental context. Its height is partly determined by its need to compete for sunlight with other trees in the canopy. The number of leaves and branches it presents is determined by the amount of moisture it requires in its specific situation. Even its shape is fashioned by prevailing wind conditions. A tree has no designer. Rather, its genetic code (and other ontogenetic mechanisms) enables it to take account of and adapt to its specific environment. Alexander's philosophy is concerned with finding the modern equivalent of a 'genetic code' for building. He does not seek to return to primitive methods of building, but rather proposes a new approach that captures some of the qualities of self-adjusting design. It is one that creates well-fitting form through adaptation and through the creation of a new design culture captured in the patterns. The idea is that the patterns, created by the community, can supply that genetic code for design.

Alexander's patterns work took shape in *A Pattern Language* [Alex77] and *The Timeless Way of Building* [Alex79]. The former presents the concrete details in a collection of 253 related patterns that Alexander terms 'a pattern language' for creating architectural form [Alex77]. Each encapsulates a solution to a problem in urban architecture and design at a variety of levels of scale, from the construction of floors and walls, the placement of windows, and the details of gardens, to the design of city buildings, streets, and surroundings. The patterns evolve from the community's culture and are designed to be used collaboratively by the builders and the community inhabitants. It is through the documentation and use of this language that this same society can obtain quality in structural forms. Therefore, pattern languages are designed to replace what has been lost in more current approaches to design [Blum96]. Their purpose is to capture the practices that will rebuild the quality once found in traditional architecture, but lost in modern structures, and to create a genuine culture of design.

In *The Timeless Way of Building, Alexander* further describes this philosophy and rationale for design that makes use of traditions, captured in patterns—in a piecemeal approach[2] to creating well-fitting form. He stresses that the patterns are applied during the construction process when a problem in a given context creates the need for one. The application of a pattern to correct a problem results in a change in the system's state, thus creating a new context, perhaps with a new problem to which a new pattern can then be applied. Alexander explains that, each of these acts is "done to repair and magnify the product of previous acts", which slowly generates "a larger and more complex whole than any single act can generate" [Alex79].

---

[2] Alexander expanded on this concept of piecemeal growth in his first rule of his "Seven Detailed Rules of Growth" in his book *A New Theory of Urban Design* [Alex87]

Alexander uses a concept of *structure preserving transformation* to gradually introduce a gradual process of differentiations [Alex96]. He relates it to the emergence of organic life which is generated, not through a plan that dictates where cells should be placed, but rather through a subtle organized cooperation of parts. Therefore, a living order is formed purely by the interaction of cells guided by their genetic code. He compares patterns in a language to seeds in a genetic system which, through millions of small acts, have the power to create form [Alex79]. He argues that, as in biology, the structure of a town can and should be woven from the interaction of individual acts of building. This piecemeal approach should be guided by the culture's traditions rigidly maintained in a common language. The language, a collection of related patterns, is what governs the construction of the parts and, in turn, the orderly emergence of the whole [Alex79].

Alexander claims that pattern form and the form of final whole structures should be developed organically by a social process in the community. This should be the result of a long sequence of tiny acts and transformations which, if they are repeated often enough, have the power to create a pattern and eventually a language of patterns [Alex79]. The organic nature also calls for the community to continually review and update the patterns and the pattern language.

However, creating and updating the patterns and pattern language is not enough. While some ordering constraints are visible when the patterns contain pointers to related patterns that help complete them, the community must also understand how to apply the patterns one at a time, causing transformations towards the creation of the final form. Alexander uses the idea of a "sequence" to teach designers and builders how to construct a coherent artifact. Without some ideas on how to sequence one's design thoughts, the underlying pattern language is likely to be mostly a diagnostic tool.

Imagine a puzzle ring. It is a set of 4–7 individual rings which, when manipulated the right way, create a beautiful and mostly traditional ring. A pattern language describing the final ring is not very helpful when trying to put it together, and a pattern language with individual transformations that can be made to the unformed ring might not be enough either if each pattern describes a single transformation. What is required is a *sequence* to follow, pointing out the order of different design decisions that will utilize the patterns to create different forms.

Four Ring Puzzle

This approach to design and building that allows the details to be fitted to the overall, evolving structure is best explained in Alexander's own words as follows:

> *The fundamental philosophy behind the use of pattern languages is that buildings should be uniquely adapted to individual needs and sites; and that the plans of buildings should be rather loose and fluid, in order to accommodate these subtleties … Recognize that you are not assembling a building from components like an erector set, but that you are instead weaving a structure which starts out globally complete, but flimsy; then gradually making it stiffer but still rather flimsy; and only finally making it complete stiff and strong.* [Alex77].

To envision this philosophy, Alexander compares the construction process of a novice to that of a master carpenter. While the novice's inexperience and fear prompts his desire and need for a blueprint, the master carpenter has the ability to make decisions about details and correct problems with small, incremental steps while the construction is being done [Alex77]. This is because, unlike the novice, the master has learned a pattern language for building and has the ability to combine these patterns to form a structure. Therefore, his actions are guided, not by a master plan, but "according to the processes given by the pattern language in his mind". Alexander points out that the master's approach allows the production of a well-fitting form through a continuous analysis and repair of failures and a commitment to detail, variety, experimentation, and wholeness [Alex79].

This method of construction, based on the piecemeal correction of misfits, is markedly different from modern architecture practice. Therefore, in the third of the patterns trilogy books, *The Oregon Experiment* [Alex75], Alexander describes, by way of example at the University of Oregon, practical details for how his ideas for an entirely new attitude in architecture and planning may be implemented. This includes the creation of organic order, the role of community participation, the process for piecemeal growth, the use of patterns, and the importance of coordination and regular diagnosis in the planning process[3] [Alex75].

In summary, the piecemeal approach governed by interdependencies between patterns is the cornerstone of Alexander's philosophy of building. It avoids the totalitarian order of a strict master plan that hinges on a view of an environment that is static and discontinuous. Instead, it recognizes an environment that is dynamic and continuous and therefore promotes moving forward in small steps. Ultimately, this permits organic order to arise, defined by Alexander as the perfect balance between the needs of the individual parts and the needs of the whole. The core principles of a community are captured by the members in a pattern language of general building practices. These patterns form the basis for shared agreement and are therefore used by all stakeholders, through small acts of building, to create communities. As a result, each community has its basic requirements met with the quality the inhabitants desire. The structures are usable, adaptable, and ultimately improve the human condition.

[3] This piecemeal growth and inspect and adapt philosophy is very similar to what is known as Agile practices.

## The Relevance of Alexander’s Philosophy to Current Patterns Movement

Christopher Alexander inspired Kent Beck and Ward Cunningham to write their first small pattern language in 1987 for designing user interfaces. In 1993, Kent Beck and Grady Booch sponsored a mountain retreat in Colorado which was the start of the nonprofit Hillside Group. The original vision of this group was to help uplift the software community through pattern writing. Toward this end, the Hillside Group started the Pattern Languages of Programming (PLoP) conference series; variations are now being held around the globe. PLoP conferences follow a highly collaborative style based on “shepherding” before submission and peer based feedback workshops during the conference. Many successful pattern papers and books have emerged from this process [Hohp13].

In 1994, Erich Gamma and his colleagues wrote *Design Patterns* which launched the concept of patterns to a broader audience; as of this writing, it has sold more than 700,000 copies in 14 languages [Gamm95]. It is interesting to note that in a discipline that is known to experience continual change, these principles are still useful and this book continues in its popularity more than 25 years after publication.

This success of patterns within the software community led to a lot of early hype but has diminished as people realize that patterns neither replace design skills nor solve all problems. Still, well crafted patterns provide valuable nuggets of relevant advice based on actual experience. Because learning by doing (learning from making mistakes) often isn’t an option for real world projects, patterns can provide a way to learn from others’ experiences (including mistakes).

In these early years of the patterns movement, Alexander observed that patterns were primarily being used as a “neat format,” a tool for communicating good ideas about software design [Alex96]. Even though he encouraged the industry to think about patterns as much more, at that time there was no visible evidence that the pervasive view of patterns was anything more than an attempt to capture successful practices. The consistently reported benefits of patterns were seen primarily as an effective way to capture expertise and pass it along to others in the form of a standardized vocabulary.

Therefore, most pattern authors focused on structure in the form of various pattern formats or templates while authoring individual patterns or small collection of patterns. The development of languages and a systematic process for using the patterns was rarely considered; in addition, the patterns community was more interested in pattern creation than in the practice of refinement. While this notion of using individual patterns as a means to communicate successful practice is part of Alexander’s philosophy, a focus on structure only misses the important dimension of the process offered by pattern languages and the role of community in keeping the patterns “alive”.

Those who recognize this vision in the midst of a majority who see patterns primarily as structure has spawned some disagreement in the patterns community. Some believe that because the pattern structure allows the industry to work towards capturing and reusing its best practices, it is enough, at least for now. Others see that it is vital to follow Alexander’s philosophy of patterns in order to address some of the critical issues in designing and building complex software [Copl96; Gabr96].

Those who believe the latter have criticized the view of patterns popularized by Gamma. Despite the impressive sales of this book, there has been continuing debate about whether these artifacts should indeed be referred to as patterns. In contrast to Alexander, Gamma patterns do not capture knowledge that is as long-term and well tested as that found in building and city architecture and do not contain sequences that define the order in which the patterns should be used. Instead, Gamma patterns focus on the basic object structure with solution examples. Jackson, in his book *Problem Frames*, also makes note of the Gamma patterns' emphasis on the solution, rather than the problem as Alexander originally intended [Jack01]. These missing characteristics—a specified order (sequences) and the ability to create complete structure (language)—reveal that the Gamma patterns are related only loosely, are not part of a language and therefore do not define a process for using them.

The popularity and claimed usefulness of this book causes it to be the foundation for many developers' notion of what a pattern is. For them, the concept is derived from Gamma and has little to do with anything called an Alexandrian pattern. In general, such people do not have a deep understanding of Alexander's work or don't view his work as relevant to software.

Concern over this attitude was discussed in October 2000 at an OOPSLA conference panel titled "Sequel to the Trial of the Gang of Four". One of the authors of *Design Patterns*, John Vlissides, stressed that the purpose of the book was to "plant a stake in the ground", arguing that it is better to take incremental steps rather than attempt to wait until you can get it completely right the first time. Frank Buschmann appeared to agree when he reminded the audience of a "do it, reflect, start over again" approach. However, it was argued that if the authors had begun with a system perspective, we'd be better off today. Instead of creating individual techniques (patterns), it was argued that they should have looked at how each structure could be part of a larger whole that contributes to the quality of life. Alexander emphasized, in his keynote at OOPSLA'96, that software developers have a social responsibility to do this because, unlike building architects, they touch everything.

Dave Ungar challenged the assumption that software can apply building metaphors to their discipline because the constraints are so different. However, the underlying theory, such as the process of creating organic structure can map into software construction. Alexander's keynote in 1996 gave the software industry a wake-up call telling them that they were in bad shape and had a reason to reflect. Just as Alexander noticed that the quality in architecture has virtually disappeared due to a lack of system perspective that puts the production of the environment in the hands of the people who use it, problems in software are system problems. Although the work in *Design Patterns* is useful, it can be argued that only patterns that are part of a pattern language can work together and give developers the ability to build software with a system perspective.

Even though the panel ended with Ungar's suggestion that it is now time to take this system perspective, there is no clear evidence, many years later, that this is happening. Even though the software patterns movement was prompted by similar observations that prompted Alexander's life work, the software industry has given only some consideration to the writings of Alexander as a means to explore how his philosophy of a patterns and the underlying design and construction process that stems from it can be useful in developing software. Alexander observed poor quality in architecture that he argues exists because of the lack of shared knowledge for timeless, successful traditions in building and urban architecture. But he also saw the need for a system-based process that supports the use of this literature, one that is able to build quality despite the need to handle the complex architecture demands. Similarly, quality in software has suffered, to some degree, from the lack of a consistent use of its successful practices, and this has created more interest in reusing proven practices throughout the industry. In addition, there is the nagging need to handle the growing complexity and the need to improve quality in present day software with a development process that can cope with this reality.

As seen in the growing Agile movement, Alexander's philosophy resists a linear, master plan development process and raises concerns about artificial models that separate the designer and the user. This is likely the reason Agile developers were among the first to include a pattern-based design approach in order to encourage a piecemeal, participatory approach that integrates, rather than separates, software development roles and activities. Rather than a master plan previously found in a waterfall approach, the stakeholders in an Agile project adopt a process that is similar to an order governed by pattern interdependencies. Each step in the construction process involves an analysis of the current problems presented within the structure and the misfits with its environment. This can be followed by an application of a pattern (or a set of patterns) that corrects the problem and repairs the misfits. In this way, the final form of the structure is transformed, strengthened, and brought to a closer equilibrium with its environment [Lea98].

This piecemeal construction based on the stepwise application of patterns, is an alternative to the formal modeling, master plan approach often seen in software engineering. Piecemeal construction recognizes continual analysis, design, and adaptation as an inevitable part of construction, a characteristic some have argued is central to handling the complexity of present system development projects. It is supported by [Blum96] and [Laws97] who are among those calling for a design process that is able to manage change instead of one that requires knowledge of the complete product at the beginning. They point to the reality that information is never complete, and changes to resolve one problem often affect the choice of solutions to other problems. Therefore, Blum states that design is always "a contingent process" and must provide for "perpetual discovery".

Henry Petroski, an American engineer specializing in failure analysis, would agree with this need for perpetual discovery. Similar to Alexander's approach to building form through stepwise correction of misfits, Petroski explained, in a 2001 OOPSLA conference keynote, that the continual observation of failures and the effect of their correction on the complete system is a fundamental underlying principle that effective designers follow. In software, Gabriel points out that software development work is rarely done with a thorough abstract design, but instead is accomplished through piecemeal growth [Gabr96]. In fact, it can be argued that the early agile movement was influenced by many of these ideas. Most of the early agilists were directly involved in the early patterns movements, many of them attending and publishing at early PLoPs.

In this piecemeal growth process, Alexander emphasizes the role of the community surrounding the project. Participation is encouraged from all levels during the creation of the patterns, the building of the structures from the patterns, and the decision-making about future growth. Collective development is made possible by a common pattern language of practices that all stakeholders in a project can use to create quality form [Lea98]. A common language allows all stakeholders to integrate, rather than separate, their roles. This is necessary in software development because human communication is a vital part of software development. Therefore, patterns have the potential to facilitate better communication between people involved with all aspects of software including the  community of software developers and their clients.

In his OOPSLA'96 keynote presentation, Alexander pointed to the "abundant connections" that can be drawn between his field and software development [Alex96]. He asserted that his lessons are something that can and should be adopted by software engineers, proposing that the idea of a piecemeal design process forms the core of the computer science field and can become the natural process because software design methods are perfectly designed for it. He stressed that, similar to living architectural structures, computer science has the means to view their software as a natural, genetic infrastructure in a living world. This, he claims, could "turn the world around, and make living structure the norm once again, throughout society, and make the world worth living again" [Alex96].

Three years later, when the presentation was published in *IEEE Software*, the foreword reported that the patterns discipline has become one of the most widely applied and important ideas of the past decade in software engineering [Copl99]. It has even been suggested that Alexander has perhaps had an even greater impact on computer science than on architecture [Sali00]. As Coplien writes:

> *The curious parallels between Alexander's world of building and our world of software construction helped the ideas to take root and thrive in grassroots programming communities worldwide.*

Although Alexander's vision is not a complete theory, it does provide an evolution of thought in which the concept of patterns and a pattern language has remained a continuous element. The particulars of building architecture knowledge may not be as important to software development as what Alexander teaches about design thinking-- it is not simply patterns thinking, but a broad approach to design that embraces the creation and use of patterns.

## Where are we?

The patterns community has continued to evolve. It is difficult to determine the exact number of documented patterns. Linda Rising's The Pattern Almanac 2000 listed more than 1,000 patterns [Risi00]. The various PLoP conferences around the world, sponsored by the Hillside Group (www.hillside.net), have accepted more than 2,000 papers over the years. The submission rate to those conferences has been constant, at over 100 papers annually. Using a conservative estimate of four patterns per paper, plus all the books and face design, mobile app development, adaptive systems, sustainable architectures, domain-specific patterns, meta-architectures, workflow, fault-tolerant systems, and security patterns, along with over 1000 human action patterns by the IBA lab [Sasa16] would put the total over 10,000 patterns published to date. Some patterns have been catalogued into collections or pattern libraries. Many companies, including Amazon, Google, IBM, Lucent, Microsoft, Oracle, and Siemens, have written similar pattern collections, some of which are available in books and on websites [Hohp13].

Pattern authors accept the definition of an individual pattern as a proven solution to a problem in a context. They recognize that each one documents a reusable solution, encapsulates knowledge about successful practices, and provides information about its usefulness and tradeoffs. Some authors have attempted to weave their collection of patterns into a language. Connected sets of interrelated patterns building on each other can start to form a pattern language, which support a generative, domain specific development process [Hohp13]. More recently there has been even more work toward building pattern languages and improving pattern mining and validation. For example, the Iba lab has outlined Patterns 3.0 pattern languages [Iba11] for guiding human action. Additionally the IBA lab has done novel work on pattern mining [Sasa16]. Another example can be found in the *Fearless Change* patterns [Mann05; Mann15] which began to build a process for evolving an organization during times of change. Fearless Change began to connect their individual patterns into a language by resolving negative consequences in each pattern with a recommendation for new patterns to use.

Patterns are also being successfully used in other fields, such as design, media, arts, IT, management, innovation, music, pedagogy, social activism, social innovation, and grassroots movements. The wider PLoP community is starting to expand by focusing on specific topics with conferences such as PurpleSoc[4] that is concentrating on patterns for social change. Moreover, there are groups such as the Pattern Science Community that supports the growing pattern research community by working "in the tradition of Christopher Alexander". They include practitioners of many application domains, academic disciplines, communities of practice and social movements. This community is integrated with PLoPs (EuroPLoP et al.), PUARL, and PURPLSOC conference series. Some of their initial topics are the mutual support and cross-pollination of ideas, networking for pattern projects of all types, with the goal of improving the shared scientific standards of pattern science.

But where do we need to go from here? The patterns community has yet to take advantage of all aspects of patterns as proposed by Alexander. We have focused on the structural aspect by writing, workshopping and publishing many individual patterns. Primarily through the PLoP conferences, we have worked towards building a community. Even though this is much more than what Alexander did, we need to reach out and widen our community. We also need to encourage the community to develop a method for keeping the patterns alive. Regarding the process aspect, we have done only limited work in structuring our individual patterns into languages and have not concentrated on developing sequences to show how patterns can be used

[4] http://www.purplsoc.org/ Patterns for Social Change

to create a timeless way of building our products or systems. Yet, as Alexander has taught us, it is important to weave together all three aspects, structure, community, and process. In the following section, we describe these three aspects in more detail with the goal toward encouraging the community to think more deeply about each one as well as how to weave them together to realize their full benefits.

## Patterns as Structure for Capturing Knowledge

We have shown that the structure of patterns has its roots in the design philosophy of Christopher Alexander who used patterns to capture successful traditions in building quality structures. His definition of a pattern is widely cited throughout the software discipline (e.g. [Saun98; Busc96; Gamm95; Hohp13]):

> *Each pattern describes a problem which occurs over and over again in our environment, and then describes the core of the solution to that problem, in such a way that you can use this solution a million times over, without doing it the same way twice* [Alex77].

The problem and solution are the essential content in a pattern template. In addition, a pattern provides information such as the conflicting forces that create the problem, the context in which the pattern is applicable, and the rationale for and consequences of using the solution. A number of approaches for writing this information can be found in the literature. Some advocate the use of clearly marked sections to make it easy for the reader to find key elements of the pattern [Mesz98] while others make use of a more free-form format that is closer to the original one used by Alexander (e.g. [Olso02; Foot00; Harr99; Yode14]). Various pattern template formats have evolved over the years. However, many feel it is more important to explore the content and use of patterns rather than to define a formal representation for them.

While an individual pattern documents a successful solution to a recurring problem, building relationships between them into what is known as a ‘pattern language’ provides the resource to handle truly complex problems. A pattern language is a collection of patterns that are related, and thus are primed to work together as a system in various sequences to build a variety of whole forms.

Alexander compares this to the English language, a system that allows the creation of “an infinite variety of one-dimensional combinations of words, called sentences”. Just as the English language provides the words and the grammatical rules for arranging the words to generate various legitimate sentences, a pattern language provides the patterns and the structural connections that specify how the patterns can be used to generate various types of designs. Alexander’s architectural language gives builders the power to create an “infinite variety” of buildings, gardens, towns [Alex79]. As an example, Alexander lists a sequence of ten patterns from *A Pattern Language* that were used to create a farmhouse in the Bernese Oberland and a sequence of eight patterns that were used to create stone houses in the South of Italy [Alex79].

While single, unrelated patterns are used in isolation to solve isolated problems, building relationships between them into a ‘language’ gives some indication of how the patterns can work together to solve complex problems. To make this possible, an individual pattern, as part of its structure needs to document its relationship to other patterns in the language. These relationships can manifest themselves in a variety of ways, showing complements such as specializes, generalizes, parallels, uses or completes, follows or precedes [Mesz98]. The resulting structure becomes a language which reveals the meaningful order in which the patterns can be applied in a variety of sequences, building on each other to create various whole forms. In the midst of this exploration, patterns start to offer a structure for capturing abstractions that are not easily captured otherwise [Gabr96].

*Therefore* we believe that the patterns community needs to not only write individual patterns, but to also work harder at integrating these into pattern languages. Pattern authors need to take responsibility for documenting the relationships between patterns as well as the alternatives among them. This can be done as initial drafts are written and further developed as the patterns are shepherded and workshopped.

One of the practical ways to illustrate the structure of relationships and alternatives is to create some type of a pattern map showing the potential big picture relationships and alternatives. Some authors that have held PLoP workshops to assist them in finding relationships with the hope of making their patterns more understandable. This has shown to be a useful start because it builds the structure that can lead to the process of how the patterns can be used. But it's not enough.This focus on using patterns solely as a structure for capturing knowledge still does not realize the full benefits of the process for how patterns and languages can build form. This is where, as explained in the next section, a collection of pattern sequences provide concrete scenarios on how to use them.

## Patterns as Process for Building Form

Imagine inheriting a fabulous toolbox from a master builder. It contains useful, well-tested tools but imagine that you don't know how to best use them to create anything such as a beautiful bookcase. You might know how to use some of them, such as a screwdriver and hammer (individual patterns) to do simple things, but you still don't have the knowledge on how to best use all the tools (patterns) together, in the right order with a process to create something of more value. A collection of sequences illustrating how the tools can actually build various types of bookcases would be useful.

In *The Timeless Way of Building*, Alexander describes how the life and beauty of great cathedrals arise from pattern languages:

> *... the rules which formed the great cathedrals were, to some extent, common rules of thumb, which defined the general form of "a" cathedral. ... And it is not only the obvious large scale organization which was composed of common patterns. At a smaller scale, there were patterns too. ... Indeed the most beautiful details were patterns too. ... There were hundreds of people, making each part within the whole, working, often for generations. ...each person in the whole had, in his mind, the same overall language. Each person executed each detail in the same general way, but with minor differences. ...*

> *the builders themselves knew enough of the shared pattern language to make the details correctly, with their own individual flair. But still the power and beauty of the great cathedrals came mainly from the language which the master builder and his builders shared. ... The building grew slowly, magnificently, from the impact of the common pattern language from which it was made, guiding its individual parts, and the acts which created them, just as the genes inside the flower's seed guide and then generate the flower. All the great buildings in history have been built like this, by language* [Alex79].

This excerpt refers to the shared language, the collection of related patterns that guides the process of building. The process stems from the structural relationships between the patterns. These structural relationships guide the creation of sequences showing how complex patterns can be solved with the use of individual patterns and the relationships between them. Each pattern solves a problem, which then creates a new condition with new conflicts that must be addressed with the application of another pattern. For example, when building the bookcase (a complex problem), the sequence would show the collection of tools (patterns) that would be used in a particular order to create this form.

Alexander believes that quality cannot be built with an isolated pattern, but rather with an entire system of patterns that are interdependent at many levels. In Alexander's language, the structural relationships between the patterns prompt sequences. Alexander has outlined importance of sequences with examples[5] that move from larger to smaller patterns, such as those that create regions (e.g. Identifiable Neighborhood (14), Activity Nodes (30)) and the buildings in those regions (e.g. House For A Couple (77), Individually Owned Shops (87)), to those that are concerned with various levels of details that embellish the structures (e.g. Alcoves (179), Final Column Distribution (213)).

Despite the capacity for pattern languages that define relationships and sequences that define various processes for creating form, it is difficult to find modern-day pattern authors who pay close attention to this. As a result, there is a scarcity of complete pattern languages. Pattern writers are creating languages to help build parts of systems to address various individual issues; therefore, languages appear at more of a component level than a system level—they do not yet define the complete design process.

Even though there is a scarcity of pattern languages and sequences that define larger processes for complex building, an element of process can also be found in each individual pattern. Alexander explains that a pattern is both a thing and a process for creating that thing. It describes what you have to do to generate the entity which it defines. A popular view recognizes that a software pattern focuses on the structure it creates and the process for building that structure [Winn02]. Therefore, the selection of a pattern prompts the use of a process. As Shaw explains, patterns are used in practice by developers who adopt one or more of them to help shape the design of their application [Shaw95]. It can be argued that this act of looking up a pattern to find a solution for a development problem is a very different process than inventing from scratch.

[5] http://www.patternlanguage.com/sequences/sequences.html and http://www.patternlanguage.com/sequences/otherinfo/intro.html

Therefore, even individual patterns have an element of process, but it is internal to the pattern rather than in the way the individual patterns could be used in a larger process.

Some people in the pattern community are starting to look at how patterns can be used for a larger process. For example, each *Fearless Change* pattern contains relationship to other patterns to set-up the context, to complete the solution and to resolve the negative consequences. One might argue that this allows a pattern language to unfold. *Fearless Change* also includes "experience reports" that provide some examples of sequencing, showing how the patterns can be used as process to build complex organizational change [Mann05; Mann15]. This idea of inspect and adapt with an open feedback loop can also be seen in the Agile movement. The agile mindset focuses on piecemeal growth by continually adding value through constant attention to adaptation and improvement.

Although pattern collections and libraries still have value, pattern languages that include sequences give us the best value and alternatives for going through the design space. Unfortunately as a patterns community, we've had a tough time getting people to look at pattern languages rather than just individual patterns and collection of patterns. Recently, there has been more work toward languages [Iba16]. Enhancing this with sequences which include many scenarios will provide the most benefit for those using patterns.

*Therefore* our design practices must evolve to include a fundamental process as described by Alexander. This calls for strong analogies/metaphors that provide scenarios or micro-stories that are sequences which help users to understand how they can apply the individual patterns to create a new form. A global sequence, perhaps at the beginning of a patterns book, can also be very useful. Even though there are so many possible sequences, which makes it impossible to show them all, a good subset of the high value ones can help illustrate ways to think about using the patterns—this begins to make use of the process feature in patterns. If we begin to create useful sequences, we can encourage others to continue with this, thus having languages that are "alive" and continuing to grow. The authors of the *Fearless Change* language [Mann05; Mann15] made some progress in keeping a language alive as they evolved and edited their patterns in the ten years between the first *Fearless Change* book through the second publication of *More Fearless Change*. The authors are currently working on creating sequences.

Pattern languages with sequencing will give us a grammar, syntax and process for good design that creates beauty. It will also allow us to make better use of the patterns and likely have more interest in keeping them "alive" in a way we don't now. However there is still more to realizing success with patterns and pattern languages. This includes the community aspect of patterns.

## Patterns as Community

Alexander stresses that community participation is an essential feature in the patterns philosophy. In order for the language to be used in the building process, all stakeholders, not just the architects, must take part in creating a pattern language. It is only then that it can become a communal language. He explains it in this way:

> [A pattern] *forms the basis for a shared agreement in a community. Each one is, therefore, a statement of some general planning principle so formulated that its correctness, or incorrectness, can be supported by empirical evidence, discussed in public, and then, according to the outcome of these discussions, adopted, or not, by a planning board which speaks for the whole community* [Alex75].

This shared agreement is important because, as explained earlier, order is drawn from a pattern language that belongs to the community in which it is used. The language supports an approach in which projects move forward through local acts performed by members of the community. Alexander claims that this level of participation is practical because those who will inhabit the structure know most about what is needed. It also provides a way for individuals to become involved in their community, giving them a sense of ownership and some degree of control. Therefore, it is the community that builds the language and constantly evaluates and improves it.

Alexander believes that it is possible to replace the master plan approach with patterns because the tools and theory are worked out in the language. Although he talked about the vital role of community in building patterns (and tried in the Mexicali project [Alex85]), there is little evidence he was successful in bringing a community together. On the other hand, the patterns community has been evolving from the work of the Hillside Group[6] over a period of almost 30 years.

The Hillside Group started in 1993 as a nonprofit dedicated to improving human communication about software by encouraging people to codify common programming and design practice. Over time, the group expanded to include a much larger patterns audience such as social networks and community. Currently the Hillside vision and mission is as follows:

> The Hillside Group was founded on the observation that innovation and design is one of the most difficult human endeavors, requiring the creation of novelty under pressure, sometimes without the benefits of a long tradition to fall back on. The Hillside Group endeavors to help others recognize that for disciplines in early stages of development, artistry and invention are as much a part of creation as good engineering.
>
> The mission of the Hillside Group is to improve the quality of life and society as a whole. This includes architects, developers, managers, owners, workers, educators, students, and more. Understanding and helping the human element is critical for achieving success. The Hillside Group believes in making processes and design more humane by paying attention to real people and existing practices.

This community was formed with the goal of creating a body of patterns literature. Over the years, much of the activity has centered around the Pattern Languages of Programming (PLoP) conferences. Within the framework of these conferences, the patterns community has defined a process for writing and reviewing patterns. It includes "shepherding", a phase in which a paper is assigned to an experienced patterns author in order to provide feedback that will help improve

[6] http://hillside.net

the paper [Harr99]. This is followed by a “writers’ workshop”, a technique borrowed from the writing community that gathers a collection of authors together at PLoP conferences to discuss ideas for further developing their patterns [Risi98; John95; Gabr02].

As a result of this activity, the community has evolved since its early inception and there has been some progress towards realizing the community aspects proposed by Alexander. For example, there is a growing group of individuals working on social patterns as well as patterns for building interactive communities and helping with societal change, with conferences that are dedicated to this such as PURPLSOC – In Pursuit of Pattern Languages for Societal Change.

The stakeholders within the patterns community vary. There is the community that carries the pattern language forward -- those evolving the pattern language within the community or organization (inner source or open source). Then there is the community that is using the patterns. It is important that these two communities work together. For example some [Mizu15; Mizu16] have put together ways to evolve the patterns and work with the community using the *Fearless Change* patterns. Some are bridging the gap between these two communities through the use of pattern cards and other ways to use the patterns. This is a highly interactive process that helps various people see how to use the patterns, the sequences that might be useful to them, and the patterns that may be missing.

But we can go further. For example, there is little or no community for those who could benefit from locating and using the large collection of patterns created at PLoP conferences. Individuals within the patterns community usually know about their own patterns and those in the conferences they attend. But it is very hard for those outside our community to find and use these patterns. It is also difficult for all of us to give feedback on existing patterns. The patterns community mostly focuses on writing, workshopping and publishing patterns. Following this, they are not easily found, and as a result seem to get lost. Only when authors publish a well-known patterns book does this resource become easily accessible. However, even when published in a book, they become static. Presently, there is no effective way for the community to participate with feedback to help patterns evolve and grow. Although we have attempted to build a community, we fall short in the kind of community involvement Alexander proposes.

This community aspect is key because it is the community that builds the patterns, keeps them updated and “alive” and uses them to create beauty. The community needs to build the structure of the individual patterns and pattern languages, propose scenarios that illustrate the sequences on how they can be used in a process and participate in the evolution of the languages. Also, the community needs to provide means to distribute and share this knowledge.

It can be argued that the community aspect is the area that will have the most impact and probably needs the most improvement. In an attempt to strengthen our community, Norm Kerth and other leaders within the community presented, at the 2001 Hillside meeting, ideas from Amy Jo Kim’s book -- *Community Building on the Web: Secret Strategies for Successful Online Communities* [Kim00]. Kim outlined three core principles for building your community: 1) Design for Growth and Change, 2) Create and maintain feedback loops, and 3) Empower your members over time. The book proposes methods for welcoming new members, for keeping and utilizing experienced elders and for members to transition to service responsibilities. Kim also outlined the following nine design strategies:

1. Define and Articulate your PURPOSE
2. Build flexible, extensible gathering PLACES
3. Create meaningful & evolving member PROFILES
4. Design for a range of ROLES
5. Develop a strong LEADERSHIP program
6. Encourage appropriate ETIQUETTE
7. Promote cyclic EVENTS
8. Integrate the RITUALS of community life
9. Facilitate member-run SUBGROUPS

The patterns community has done well in some aspects of these principles and strategies whereas in other aspects we have an opportunity to make our community better. For example, the Hillside Group has defined and articulated its purpose, has built nice gathering places at cyclic PLoP events, has encouraged appropriate etiquette in shepherding and workshopping, and has integrated the rituals of community life. Also, the community is good at welcoming newcomers and keeping experienced elders involved.

However, the other aspects are lacking in many ways. For example it is often not clear on how well the feedback loops work. Although the community is open for growth and change, there is not a clear design for this and often it is not promoted enough within the community. Also, although there is a transition to service responsibilities, it is more ad hoc and often members of the community do not know how to get involved. It is commonly done through encouragement from elders or a shadow cabal.

One of the goals of the Hillside Group has been for a project that will help with creating and evolving members' profiles. There is no clear profile of most of the members and what ones are there do not seem very meaningful. Creating this needs to be a priority of the patterns community and can help the community to grow as well as share experience and knowledge within the community.

Additionally there needs to be an effort to better define roles that can and should evolve with time and a method for encouraging new members to take responsibilities for these roles. This can be facilitated with a strong leadership program. Presently, a few of our leaders are often over-burdened with doing many of the tasks which has often kept the community from growing in ways that it can.

*Therefore* the patterns community needs to work harder at creating an interactive community. We can evolve the more static web pages and mailing lists into interactive social networks. The community has made some small progress towards creating an online searchable catalogue. However this needs to be extended with ways to enter feedback on patterns and pattern languages. The feedback should come from both the patterns community, which includes the authors, those using the patterns, and experts in the domains. Presently, the community mostly consists of pattern authors with a few pattern users. The community needs to find ways to include others, especially those using patterns and experts from the pattern domains. This can be done through overlapping PLoP with other conferences and being more intentional about inviting people from those conferences. We can hold focused events for these people such as "Using Patterns" or "Pattern Mining" events. Also, authors of pattern books should be encouraged to join the community by inviting them to PLoPs and to also promote their books on the patterns websites, mailing lists, and social networks.

## Conclusion

A pattern has been described as a thing and a process for building that thing. It offers a structure for documenting knowledge and two other features that have been weak in past efforts – process and community. The process aspect can be strengthened by putting more emphasis on the creation of pattern languages as well as the documentation of common scenarios for using the patterns. This should include vehicles for keeping the pattern language alive and evolving. The community aspect is rather strong among a limited number of people who attend PLoP conferences. In fact, we believe that the patterns community evolved from a community aspect much more than Alexander ever did. However it is still fairly weak outside of the PLoP communities. It would be good if we can help individual organizations create their own communities. Also, there is still a lot of work that can greatly benefit the existing patterns community. This will not only serve to grow the wider patterns community but also has the potential to help make the process aspect more important as organizations define ways to create processes that use the patterns and keep them alive.

Patterns have aspects of structure, process, and community. When we focus primarily on structure, we lose the benefits that process and community offer. The modern day patterns community's attempts to capture its best practices in a reusable and effective form does not consider all three aspects. Alexander and his ideas were the gate for the current patterns community. We have travelled through that gate and in some ways have been more successful with these ideas than his original work. As we continue, it is important that we evolve and continue to grow without limitations from the gate. Similar to being on the shoulder of giants we are able to see and do things beyond the original vision.

These are some ideas that the authors hope will prompt a discussion on how the patterns community can pay more attention to the process and community features of patterns. To this end, we would like to propose the following manifesto:

> Writing patterns is much more than just writing about the structure. The most value happens with patterns that evolve to a language that shows the connections between the patterns. Also there is value in describing various sequences and a process for how to use the language. Pattern authors need to pay attention to the process aspect of patterns by creating languages and sequences. Patterns should be created with input from a community that is continually growing through invitations to people who are both writing patterns and using them in various domains. The community should also create a feedback loop for keeping the patterns and the languages alive, as well as adding new sequences to encourage more pattern use.

This manifesto can and should evolve in a piecemeal growth process within the patterns community.

## Acknowledgements:

We thank our shepherd Richard P. Gabriel (rpg) for his valuable comments and feedback during the PLoP 2017 shepherding process. We also thank our 2017 PLoP Writers' Workshop Group, Chris Richardson, Helene Finidori, Rebecca Wirfs-Brock, Maria Lydia Fioravanti, Aracele Fassbinder, Kotomi Nozaki, and Richard P. Gabriel for their valuable comments and suggestions.

## References:

[Alex64] Alexander, C., *Notes on the Synthesis of Form*. London: Harvard University Press, 1964.

[Alex75] Alexander, C., *The Oregon Experiment*. New York: Oxford University Press, 1975.

[Alex77] Alexander, C., *A Pattern Language*. New York: Oxford University Press, 1977.

[Alex79] Alexander, C., *The Timeless Way of Building*. New York: Oxford University Press, 1979.

[Alex85] Alexander, C., Davis, H., Martinez, J., Corner, D., *The Production of Houses.* New York: Oxford University Press, 1985.

[Alex87] Alexander, C., Neis, H., Anninou A,, King I.,. *A New Theory of Urban Design.*
New York: Oxford University Press 1987.

[Alex96] Alexander, C., Keynote address, *Object-Oriented Programming, Systems, Languages and Applications Conference*. San Jose, October 1996.

[Alex02] Alexander, C., *The Nature of Order: An Essay on the Art of Building and the Nature of the Universe, Book 1 - The Phenomenon of Life.* New York: Oxford University Press 2002.

[Blum96] *Beyond Programming: Toward a New Era of Design*. Oxford University Press, 1996.

[Brow00] A Design Controversy Goes Cozy.com. *The New York Times*. 23 Nov 2000.

[Busc96] Buschmann, F., Meunier, R., Rohnert, H., Sommerlad, P., Stal, M. (1996). *Pattern-Oriented Software Architecture: A System of Patterns*. Chichester: John Wiley & Sons.

[Copl96] Coplien, J.O. (1996). *Software Patterns*. New York: SIGS Publications.

[Copl99] Coplien, J.O. (1999). The Origins of Pattern Theory. *IEEE Software*. September/October 1999, 71-82.

[Foot00] Foote B., Yoder J., "Big Ball of Mud," Pattern Languages of Programs Design 4 Harrison N., Foote B., and Rohnert H., editors. Addison Wesley, 2000.

[Gabr96] Gabriel, R. (1996). *Patterns of Software*. New York: Oxford University Press.

[Gabr02] Gabriel, R. (2002). *Writers' Workshops & the Work of Making Things: Patterns, Poetry…*Boston, MA: Pearson Education.

[Gamm95] Gamma, E., Helm, R., Johnson, R., Vlissides, J. (1995). *Design Patterns: Elements of Reusable Object-Oriented Software*. Reading, MA: Addison-Wesley.

[Grab83] Grabow, S. (1983). *Christopher Alexander: The Search for a New Paradigm in Architecture*. Stocksfield: Oriel Press.

[Hohp13] Hohpe, G., Wirfs-Brock, R., Yoder, J., Zimmermann O (2013). Twenty Years of Patterns' Impact. *IEEE Software*. Volume: 30, Issue: 6, Nov.-Dec. 2013.

[Harr99] Harrison, N.B. (1999). The Language of the Shepherds. *Proceedings of the 6th Annual Conference on the Pattern Languages of Programs*. August 1999, Monticello, Illinois, USA. 15-18.

[Iba11] Iba, T. 2011. "Pattern Language 3.0 Methodological Advances in Sharing Design Knowledge," International Conference on Collaborative Innovation Networks 2011 (COINs2011).

[Iba16] Iba, T., Isaku, T., "Creating a Pattern Language for Creating Pattern Languages: 364 Patterns for Pattern Mining, Writing, and Symbolizing," 23rd Conference on Patterns of Programming Language (PLoP 2016), Monticello, Illinois, USA, 2016.

[Jack01] Jackson, M. (2001). *Problem Frames*. Harlow, England: Addison-Wesley.

[John95] Johnson, R, Cunningham, W. (1995). *In* Coplien & Schmidt (Eds.). *Pattern Languages of Program Design I*. Reading, MA: Addison-Wesley.

[Kim00] Kim A. (2000). *Community Building on the Web: Secret Strategies for Successful Online Communities*. Berkeley CA: Peachpit Press a division of Addison-Wesley, 2000.

[Laws97] Lawson, B. (1997). *How Designers Think: The Design Process Demystified*. Oxford: Architectural Press, Butterworth-Heinemann.

[Lea98] Lea, D. (1998). Christopher Alexander: An Introduction for Object-Oriented Designers. In Rising, L., *The Patterns Handbook*, Cambridge: Cambridge University Press, 1998.

[Mann05] Manns, M.L., Rising, L. (2005). *Fearless Change: Patterns for Introducing New Ideas.* Boston: Pearson Education.

[Mann15] Manns, M.L., Rising, L. (2015). *More Fearless Change: Strategies for Making Your Ideas Happen.* Boston: Pearson Education.

[Mesz98] Meszaros, G., Doble, J. (1998). A Pattern Language for Pattern Writing. *In*: Martin, Riehle, Buschmann (Eds.). *Pattern Languages of Program Design 3*. Reading, MA: Addison-Wesley. 1998, 529-574.

[Mizu15] Mizutani, M., Takahash, M., "Patterns for a Company to Collaborate with Local Community on Social Issues," 23nd Conference on Patterns of Programming Language (PLoP 2015), Pittsburgh, Pennsylvania, USA, 2015.

[Mizu16] Mizutani, M., Wakui, M., Ozaki N., "Patterns for Collaboration between Companies and Local Communities on Social Issues," 23rd Conference on Patterns of Programming Language (PLoP 2016), Monticello, Illinois, USA, 2016.

[OCal01] O'Callaghan, A. (2001). Architecture, patterns and components. *In:* Graham, I. *Object-Oriented Design Methods*. London: Addison-Wesley.

[Olso02] Olson, D., Stimmel, C. (2002). *The Manager Pool: Patterns for Radical Leadership*. Boston: Addison-Wesley.

[Risi00] Rising, L. (2000). *The Patterns Almanac 2000*. Boston: Addison-Wesley.

[Risi98] Rising, L. (1998). Pattern Writing. In Rising (Ed.). *The Patterns Handbook*. UK: Cambridge University Press.

[Sasa16] Sasabe, A., Kaneko, A., Takahashi, K., Iba, T., Pattern mining patterns: a search for the seeds of patterns," 23rd Conference on Patterns of Programming Language (PLoP 2016), Monticello, Illinois, USA, 2016.

[Shaw95] Shaw, M. (1995). Patterns for Software Architectures. *In*: Coplien, J., Schmidt, D. (Eds.). *Pattern Languages of Program Design*. Reading, MA: Addison-Wesley.

[Sali00] Salingaros, N.A. (2000). Some Notes on Christopher Alexander. http://www.math.utsa.edu/sphere/salingar/Chris.text.html. (20 February 2000).

[Saun98] Saunders, D. (1998). Patterns: The New Building Blocks for Reusable Software Architectures. *In* Rising (Ed.). *The Patterns Handbook*. UK: Cambridge University Press.

[Winn02] Winn, T. Calder, P. (2002). Is This a Pattern?. *IEEE Software*. Jan/Feb 2002, 59-66.

[Yode14] Yoder J. and Wirfs-Brock R., "QA to AQ Part Two: Shifting from Quality Assurance to Agile Quality," 21st Conference on Patterns of Programming Language (PLoP 2014), Monticello, Illinois, USA, 2014.