Institute of Architecture of Application Systems

A Process for Pattern Identification, Authoring, and Application

Christoph Fehling, Johanna Barzen, Uwe Breitenbücher, Frank Leymann

Institute of Architecture of Application Systems,
University of Stuttgart, Germany
{nachname}@iaas.uni-stuttgart.de

BibTeX

```
@inproceedings {INPROC-2015-50,
    author = {Christoph Fehling and Johanna Barzen and Uwe
Breitenb{\"u}cher and Frank Leymann},
    title = {{A Process for Pattern Identification, Authoring, and
Application}},
    booktitle = {Proceedings of the 19th European Conference on Pattern
Languages of Programs (EuroPLoP)},
    publisher = {ACM},
    institution = {Universit{\"a}t Stuttgart, Fakult{\"a}t Informatik,
Elektrotechnik und Informationstechnik, Germany},
    pages = {1--9},
    type = {Konferenz-Beitrag},
    month = {Januar},
    year = {2015},
    language = {Deutsch},
   url = {http://www2.informatik.uni-stuttgart.de/cgi-
bin/NCSTRL/NCSTRL_view.pl?id=INPROC-2015-50&engl=0}
 }
```


Universität Stuttgart
Germany

A Process for Pattern Identification, Authoring, and Application

Christoph Fehling, Johanna Barzen, Uwe Breitenbücher, Frank Leymann

Institute of Architecture of Application Systems
University of Stuttgart
Stuttgart, Germany
{fehling, barzen, breitenbuecher, leymann}@iaas.uni-stuttgart.de

ABSTRACT
The process to identify, author, and apply patterns is mostly performed manually by pattern experts. When performing pattern research in large domains involving many persons, the current state of the art of pattern research techniques, such as shepherding and writers' workshops, should be extended to a larger organizational process coordinating the work of all involved participants. This paper presents the process we followed to identify, author, and apply patterns in various domains involving multiple industry partners. Due to the diversity of these domains, we claim that the process is general enough to be applicable in other domains as well. This paper documents this process for use, discussion, further refinement, and evaluation in a larger pattern research community.

Categories and Subject Descriptors
D.2.1, D.2.2, D.3.3

Keywords
pattern authoring, pattern languages, pattern research method

1. INTRODUCTION
Patterns are a well-established concept to document proven solutions to reoccurring problems in well-structured documents. Each pattern captures an abstract solution to such a reoccurring problem in the considered domain. Patterns are interrelated through references in order to guide users during their application. Through these interrelations the patterns of a certain domain form a so-called *pattern language* that specifies an order of consideration during the application of the contained patterns.

In the remainder of this introduction, the domains are covered in which we identified patterns. From these works, the process covered in this article has been generalized to be applicable by other pattern authors and pattern users. The process, thus, supports pattern authors during the identification and abstraction of patterns.

EuroPLoP '14, July 09 - 13, 2014, Irsee, Germany Copyright is held by the owner/author(s). Publication rights licensed to ACM. ACM 978-1-4503-3416-7/14/07…$15.00 http://dx.doi.org/10.1145/2721956.2721976

Pattern authors and pattern users alike can follow the covered aspects regarding pattern application.

In the past years, we identified, researched, authored, and applied patterns in various domains [5] [17] [18] [21]. As a means to structure this research, we developed an overall process to guide (i) the collection of information from which patterns may be deducted, (ii) the extraction of patterns themselves, as well as their (iii) later application in concrete use cases. The domains we considered during our pattern research are as follows:

Cloud Computing Patterns [4] [5] [6]: in collaboration with several industry partners, we reviewed existing applications that had been developed for cloud environments and those that should be migrated to it. The resulting cloud computing pattern language contains patterns to characterize the cloud environment by describing different deployment options, service models, and offerings provided by clouds. In the diversified market of cloud computing, this helped our industry partners to characterize cloud providers and their offerings on an abstract conceptual level. Especially, this made providers comparable with each other as well as existing IT infrastructures of these companies. As these patterns for cloud offerings describe how offerings behave, they help to solve the architectural problem when to select a certain cloud provider and offering. Additional patterns describe how to design, build, and manage cloud applications on top of those offerings and, especially, how to cope with the provider-specific properties expressed as pattern implemented by the provider. These patterns have been extracted from existing applications as well as provider documentation and have been refined multiple times to create new cloud applications and to restructure existing ones.

Cloud Data Patterns [22] [23]: Providing best practices to deal with challenges that may arise when migrating the data layer of an application to cloud environments is the purpose of these patterns. Utilizing cloud technology leads to challenges such as incompatibilities that may refer to inconsistencies between the functionality of an existing traditional database layer and the functionality and characteristics of an equivalent database layer in the cloud. In addition the challenges include data privacy issues such as avoidance of accidentally disclosing of critical data by e.g., moving them to the public cloud. In order to address these challenges a set of Cloud Data Patterns dealing with functional, non-functional, and privacy-related aspects have been identified. Through the use of decision support systems, users of these patterns are guided during the pattern selection depending on the application at hand and the cloud provider to be used etc. Based

on user selections, patterns to be implemented during the migration, for example, to obfuscate data, can be recommended [24].

Application Management Patterns [6] [7]: as part of our work on the IT management standard TOSCA [19], we researched how management processes are handling IT applications during runtime can be developed using a pattern-based management approach. TOSCA assumes applications to be componentized and modeled regarding their hosting infrastructure stack, i.e., deployment dependencies of application components on middleware, usage dependencies when application components interact with each other, etc. Management patterns presented in [6] [7] are refined semi-automatically to runtime infrastructure-specific management processes through orchestrating so-called *Management Planlets* [3]. Each of these reusable planlets captures the refinement of management activities used by the management patterns for one specific IT component. The refinement of management patterns for a certain infrastructure may then be guided using the application models created by developers and the management patterns they wish to use.

Green Business Process Patterns [16] [17] [18]: these patterns describe how business processes of organizations may be adjusted to be more environmental friendly. These patterns not only consider new patterns in this domain but also green versions of the above mentioned patterns and those from other domains, which have not been written with their environmental impact in mind. The green versions of these patterns contained in this pattern language focus on this neglected aspect. They show how the original pattern can be applied to achieve this new ecological goal or how the abstract solution captured in the original version of the pattern may even hinder reducing the environmental impact.

Costume Patterns [2] [21]: as part of our involvement in digital humanities – the use of IT concepts and technologies in the humanities research domain – we investigated the use of costumes in films. This work aimed at the extraction of patterns for proven solutions to communicate character traits of film roles through costumes, so called *vestimentary communication*. After an initial extraction of patterns for western characters, significant work is currently ongoing to document costumes from films in a data format that can be queried and processed by IT tools. This may then be used to prove the existence of reoccurring combination of pieces of clothing, coloring, or status (worn, dirty, etc.) in order to communicate character traits.

For each of these research domains, in which we conducted pattern research, the research methodology has been documented in the respective sources. In this paper, the practices employed in these domains shall be homogenized to obtain a general pattern identification, pattern authoring, and pattern application process. We argue that the diversity of the domains in which this methodology has been applied ensures the generality of the presented approach to a significant extend – after which it may then be applied to other domains by researchers of the pattern community. Still differences of other domains than those considered by us may make slight alterations of the process reasonable.

The remainder of this article is structured as follows. After the related work in Section 2, Section 3 is structured according to the phases of the process: pattern identification (Section 3.1); pattern authoring (Section 3.2); and pattern application (Section 3.3). The paper is concluded by a summary and outlook in Section 4.

2. RELATED WORK

As a starting point to our pattern research in each of the above mentioned domains, we considered other works on pattern writing, pattern review, and pattern organization.

Wellhauser and Fießer [25] describe how to craft pattern documents in a way that they are accessible to readers. In a conversation-based presentation, the authors mainly target first time pattern writers by describing how to structure the pattern document, what the semantics of contained sections should be etc. This helps the pattern author to consider the way how a pattern reader will access the pattern document and how he or she can quickly identify whether or not the pattern is applicable in the concrete use case at hand.

Meszaros [14] gives similar advice to pattern authors in the form of best practices on how to write patterns. Practices are themselves captured as a pattern language, thus, forming "a pattern language for pattern writing". These patterns guide pattern authors to obtain a good pattern document structure, to find good and well-understandable names for patterns and references, to make the pattern itself comprehensible, and to structure the pattern language containing all the created patterns. Pattern authors should, therefore, follow these patterns while they write and organize patterns in order to form their own pattern language.

Harrison [11] describes patterns for the iterative review cycle used for pattern documents by the pattern research community. This *shepherding* involves a shepherd – an experienced pattern researcher - and a *sheep* – the pattern author. These two parties iteratively review and improve the pattern document prior to its presentation at a pattern research conference. As a matter of fact, this paper will have undergone the same process at the time it is published. The "language of shepherding" described by Harrison captures best practices during this review phase again in a pattern format. It covers patterns how the shepherd and the sheep should interact, what properties of the pattern documents they should focus on in each iteration of shepherding etc. Therefore, these patterns describe what should be done after the initial drafting of a pattern document.

Lucrédio et al. [13] capture patterns to guide students during PLoP conferences. They describe the situations that may occur, the problems a participant new to the pattern research community may encounter, and how to handle them. They also help attendees to prepare for the conferences regarding their own work that is being discussed in writers' workshops as well as the other work that an attendee himself or herself has to give feedback on. These patterns are, therefore, ideal to prepare for PLoP conferences, which are in fact very different to other research conferences following a less interactive and more presentation-style mode of operation.

While these sources helped us significantly during the authoring of patterns and the organization of pattern languages, their review during shepherding, as well as their discussion and presentation at conferences, we argue that the following factors should also be considered when researching patterns in a domain:

(i) *Domain Coverage*: the domain in which patterns shall be identified and abstracted should be considered completely. Thus, the most significant and most often occurring problems of the domain shall be addressed by the patterns.

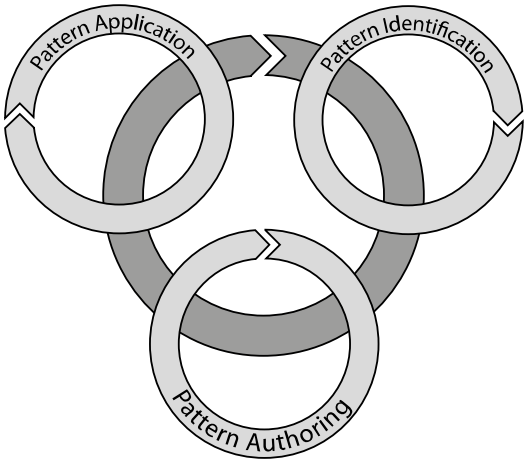

Figure 1: The Pattern Identification, Authoring, and Application Process © Christoph Fehling

(ii) *Domain Language Unification*: patterns of a domain tend to refer to similar elements in their abstract solution, i.e., a "server" in the domain of IT architecture patterns or a "hat" in the domain of costume patterns. The semantics of such textual and graphical elements should be clarified in order to homogenize their use in pattern documents by different pattern authors. These elements are, however, not patterns themselves as they do not solve reoccurring problems. Instead, they are commonly referred to as pattern primitives [26]. These primitives should be unified for patterns of a domain to provide a better reading experience for users.

(iii) *Pattern Search and Recommendation*: the structure of the pattern language and additional information annotated to patterns should be considered in order to create human-usable recommendation tools to find patterns quickly for the problems at hand.

(iv) *Pattern Refinement*: especially, for our industry partners using IT architecture patterns a manual refinement of patterns upon each application to a problem was inefficient. Often, these companies had undergone significant work to homogenize their IT infrastructure stack. Thus, the application of abstract patterns to this standardized environment had to be constrained further to ensure that the obtained architectural solutions remain manageable.

(v) *Author Group Coordination*: the guidelines presented in the related work section are mainly aimed at pattern authoring: (i) workshops discussing patterns – the writers' workshops, and (ii) mentoring of authors – the shepherding of one pattern author by a more experienced author. While this helps to organize interactive pattern writing and review, we argue that additional means are necessary to organize and structure a pattern language, enable recommendation of patterns, etc. in order to support a larger pattern research community.

In the following sections, the iterative pattern identification, authoring, and application process will be discussed. For its three phases, we will cover the pattern domains in detail where the activities comprising each phase have been used extensively. However, this does not mean that the phase has not been used in the other domains mentioned in the introduction.

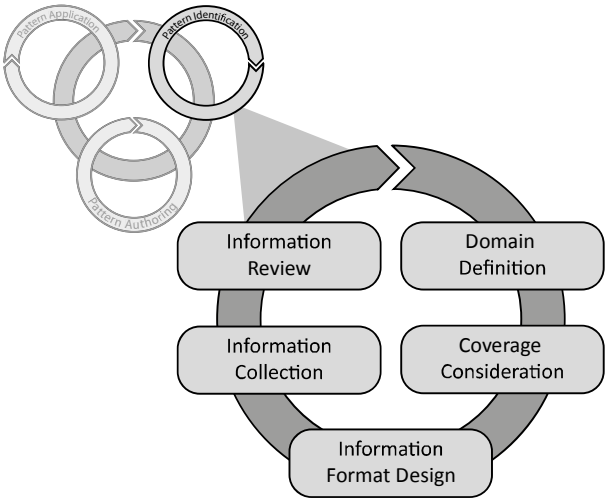

Figure 2: Detailed sub-activities of the information collection phase © Christoph Fehling

3. PATTERN IDENTIFICATION, AUTHORING, AND APPLICATION

The iterative process shown in Figure 1 consists of the three phases (i) *pattern identification*, (ii) *pattern authoring*, and (iii) *pattern application*. Each phase comprises multiple *sub-activities* that define the steps to be performed.

The overall process as well as each phase is performed in iterations to continuously drive the improvement, discussion, and adjustments of created results. The phases *pattern identification* and *pattern authoring* are responsible for creating patterns and are repeated as long as new patterns are found in the considered domain. All of the phases form an internal iterative cycle, meaning that the sub-activities comprising the phase are executed multiple times in order to refine their output continuously. The *pattern application* phase considers the refinement of patterns authored in the first two phases to a concrete use case or to a certain application environment, for example, for the specific IT infrastructure stack used by a company.

3.1 Pattern Identification

The pattern identification phase structures and collects the relevant information of a domain in which patterns shall be found. It is depicted in detail in Figure 2. The main goal of this phase is to structure the domain in which patterns shall be identified. Furthermore, the terminology and graphical elements used in pattern descriptions shall be defined in order to be used in each pattern and documented solution in a homogeneous fashion. These steps are especially necessary, if larger teams of pattern researchers shall be coordinated.

3.1.1 Domain Definition

Problem: if a larger team of pattern researchers work collaboratively on the identification of patterns in a domain, they need to have a common knowledge about this domain.
Solution: the domain is described by written text and with references to well-accepted definitions and related documents. Fundamental and defining information sources are identified and communicated to the group. If needed, definitions of domain concepts are collaboratively created for use within the group.

Result: the persons collaborating in pattern research have established a common knowledge about the domain and its specific definitions and terminology. During the following iterations, these definitions and terminology are continuously revised. This effort ensures that patterns are written with a similar mindset.
Examples: in the domain of cloud computing the NIST definition [27] has been identified and communicated as a common denominator for all participating persons. Furthermore, terms such as "server" or "application component" have been defined in order to be used homogeneously by pattern authors. In scope of patterns for costumes in films, the genre of films to be investigated has been defined.

3.1.2 Coverage Consideration

Problem: a domain can be very large, i.e., many information sources should be considered for pattern identification. Covering all of these sources may be unfeasible depending on the size of the pattern research group.
Solution: the pattern research group identifies relevant topics in the domain or formulates constraints to manage the amount of information that has to be considered. The previously defined domain characteristics often lead to characteristic problems that have to be handled in each solution of the domain, which may be used to introduce some structuring. Furthermore, all information sources of the domain may be constrained to consider only a subset of the information sources. Especially, if the pattern research group is too small to review all of them.
Result: characteristic problems of the domain are identified for which existing solutions may be searched. The information sources to be considered are well-defined in order to coordinate the effort of the pattern research group. After this activity, it is clear what topics and information sources the pattern identification shall be based on.
Examples: cloud applications have been identified to follow certain architectural principles to profit from this domain [6]: *isolation of state* – data should only be handled by small parts of the application; *distribution* – the application has to be enabled to use distributed cloud resources; elasticity – resources used by the application need to be added and removed; *automated management* – the application should react autonomously to changes in workload or resource failures. Information sources could be investigated for solutions fulfilling these architectural principles. Furthermore, we described a set of relevant topics, such as availability, resiliency, data management billing etc. that are affected by the use of cloud computing in applications [5] to coordinate the pattern research of multiple authors. Constraints were introduced as a list of concrete cloud providers and applications of a company that should be considered for review during this activity. In the domain of costume patterns, a concrete list of films was defined during this activity that had to be watched in detail in order to document the costumes contained therein.

3.1.3 Information Format Design

Problem: if multiple persons collaborate on the review of information sources and summarize their individual findings, they likely use different forms of representations. This can hinder the later review of collected information.
Solution: The format in which information is collected should be homogenized in order to process the findings more easily. Members of the pattern research group agree on one format to express individual findings. Furthermore, the used data format, i.e., document structures, is defined in order to homogenize the work of all collaborating persons.
Result: in each domain, certain modeling languages and concepts are prevalent. For example, in the domain of IT Architecture, the *Unified Modeling Language (UML)* [32] can be used to describe components of software architectures. Finally, each domain uses specific language elements, referred to as *pattern primitives* [26]. By agreeing on the format to be used, collected findings from information sources look similar to ease perception [28]. Such similar descriptions can also enable the automated processing of documented findings or the querying of documented solutions.
Examples: in the domain of IT architecture, the concepts of a "server", "node", "component", and others have been used in solution descriptions. In order to homogenize them during the information collection and later pattern description, their graphical representations has been homogenized in [29] and provided as an authoring toolkit [4], which is also available online[1]. This authoring toolkit also includes Word and Excel template to homogenize the collection of information sources and later pattern writing. In the domain of costumes, the elements of a costume, i.e., the pieces of clothes it comprises are modeled explicitly as ontologies [2] to homogenize their use during the documentation of costumes. Furthermore, terms to describe these costumes, such as colors or status ("clean", "dirty", "bloody" etc.), had to be defined as well. Through this model of the terms to be used during the costume documentation, information could be queried afterwards, for example, to research the use of the color "red" in documented costumes.

3.1.4 Information Collection

Problem: pattern identification can be driven by domain experts who capture their experience. However, the knowledge of a domain may also be persisted in other forms than human memory. It is visible in every created artifact, such as an IT application, a building, or a costume in a film. Therefore, patterns could be identified from the existing solutions rather than in human memory.
Solution: existing solutions of the considered domain are captured by multiple persons in the defined information format. The aspects defined during the coverage consideration phase can be assigned to different persons in order to document existing solutions as a collaborative effort.
Result: based on these captured existing solutions, the research group may identify patterns even if the solutions have not been created by members of the group itself. In this activity, the pattern research group puts the previously defined format and terms into use. Information sources are reviewed and solutions are summarized.
Examples: cloud provider documentation and information about existing cloud applications has been collected and summarized in table-centric documents to collect information about how individual providers and existing applications enable the architectural principles of cloud applications. Costumes in films have been collected in a custom web-based tool by viewing films scene-by-scene.

[1] http://www.cloudcomputingpatterns.org/authoringtoolkit.zip

3.1.5 *Information Review*

Problem: during the *coverage consideration activity*, the domain has been structured by characteristic problems for which existing solutions should be found in information sources. If many information sources have been documented, this structuring may be insufficient in order to make the identification of patterns feasible: too many existing solutions would have to be considered at once.
Solution: the domain structuring is refined to create manageable sets of existing solutions considered for pattern identification. Based on the information format, queries may be realized on the set of documented existing solutions to find similar ones.
Result: smaller sets of existing solutions may be considered to identify similarities easier. However, this grouping may also lead to unidentified similarities among existing solutions that were classified differently.
Examples: in cloud applications the concept of availability could be divided regarding existing well-established strategies: avoiding faults altogether and reacting quickly to faults. Existing solutions could then be grouped with respect to the strategy followed in order to identify patterns. In scope of costume patterns, documented costumes could be further classified by character trait of the role, i.e., good or bad. Furthermore, the custom tool in which costumes have been documented could be queried to find costumes that are similar in color, style etc. to identify relations of these elements and the character traits, for example.

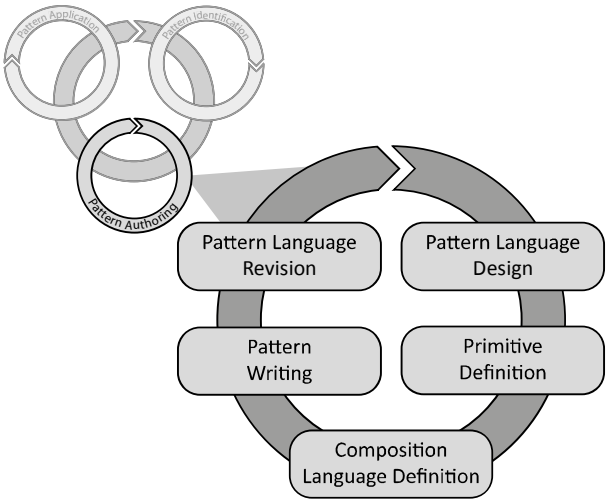

Figure 3: Detailed sub-activities of the pattern authoring phase © Christoph Fehling

3.2 Pattern Authoring

The *information collection* phase creates the information basis in which similarities of existing solutions can be identified. Based on these similarities, patterns are written during the p*attern authoring* phase detailed in Figure 3.

3.2.1 *Pattern Language Design*

Problem: while many patterns follow a similar document format comprised of sections such as "intent", "forces", "driving question", and "context" [1] [9] [10] [26], there is no generic format that fits all domains. Nevertheless, the patterns written by the author group shall be structured and organized similarly.
Solution: based on the well-established pattern formats, the concrete format used in the considered domain may be different and is, thus, adjusted to the domain's specific needs. Also, the references that may be used among pattern documents is defined to express combinations, alternatives, compositions etc.
Result: the work of multiple pattern authors is homogenized by defining and communicating two aspects to the group: (i) the pattern sections and their concrete semantics, i.e., what information should be included in which sections, how long each section should be to ensure readability etc. (ii) the semantics of pattern interrelations, i.e., is only a "related to" link used between patterns or are there additional types, such as "alternative", "composition" etc. This format and the interrelations may have to be refined during further iterations of this phase as patterns are created.
Examples: the cloud computing patterns, cloud data patterns, green business process patterns, and costume patterns use a very similar format. Each domain, however, uses some adjustments. For example, the cloud computing patterns use a leading intent statement at the beginning of each pattern that is, especially, reused in an overview poster[2]. A similar section of the green business process patterns summarizes the ecologic aspects of the patterns. During the iterations of the process, the pattern format may be adjusted. Differences can be seen in the first publication of a subset of the cloud computing patterns [5] and their final version [6].

3.2.2 *Primitive Definition*

Problem: pattern descriptions may use additional and possibly different primitives as those that have been formulated during the *information format design* activity of the *pattern identification* phase. If these primitives are used differently by pattern authors, the readability of pattern documents is reduced.
Solution: the defined primitives are revised and extended.
Result: terms and graphical elements used in pattern documents are homogenized for different pattern authors. This may, especially, subsume the creation of different versions of the existing primitives, for example, smaller graphical elements to be used in pattern icons.
Examples: graphical representations of concepts like "servers", "components", or "communication links" have been standardized within scope of the cloud computing patterns [6]. Since each pattern of this pattern language has an icon, graphical representations also have been resized to be included in these icons in a homogeneous manner. This especially ensured the same size of graphical elements in all occurrences to be easier identified [28]. In the costume domain, the modeled colors and other terms used to describe existing costumes were also used in costume pattern descriptions.

3.2.3 *Composition Language Definition*

Problem: many pattern languages use a sketch to describe the abstract solution as a graphic. While the primitive refinement step homogenized the graphical elements used in these sketches, the composition of these elements to form sketches may be handled differently by pattern authors.
Solution: the pattern research group agrees on guidelines to

[2] http://extras.springer.com/2014/978-3-7091-1567-1/pattern_overview_A3.pdf

create sketches or maybe even on a formal specification of a composition language. Especially, this agreement can describe the composition of existing patterns to form a new solution sketch, for example, through the use of their pattern icons.

Result: the composition of graphical elements into a sketch is standardized and possibly formalized. Sketches, therefore, obtain a common look and feel as graphical elements are used in a similar manner. If the composition is formalized, the modeled sketches may even be verified for correctness.

Examples: in the domain of costumes, the language to describe that garments are "worn above" or are "attached to" each other to form a costume has been formalized [30]. In the IT architecture domain, pattern authors could decide to use a certain modeling language, such as UML [32] or ACME[3], to describe pattern solutions. The cloud computing patterns use human readable guidelines on how to combine the graphical elements and pattern icons in sketches.

3.2.4 Pattern Writing

Problem: it can be extremely difficult for the domain expert to author a pattern at the correct level of abstraction. If the existing solutions are abstracted too much in order to obtain a general solution to the often reoccurring problem, pattern users will have problems applying the pattern as not enough information is contained in the pattern document. If the pattern document does not abstract enough from existing solutions, is may be more difficult to apply the captured knowledge to new occurrences of the problem.

Solution: after its initial creation, the pattern document is discussed and revised by other pattern authors and pattern users. This group of persons reviewing the pattern document may even be unfamiliar with the considered domain to ensure that the pattern may be used as a learning resource. During further iterations of this activity already existing patterns should be revisited with respect to the existing solutions that have been documented since the first definition of the pattern.

Result: the aim of the prior two activities was to homogenize the created pattern documents in order to ensure an easier accessibility for readers. During the pattern writing activity, pattern authors use the identified similarities in documented solutions to write patterns using the defined pattern format. The patterns and guidelines for pattern writing, which were mentioned in the related work section are considered during this activity in order to improve and verify patterns in a larger community. Therefore, *shepherding* and *writer's workshops* are held in order to improve the pattern document.

Examples: the information about costumes could be used to find garments worn by all documented sheriffs in western films. These pieces of cloth could then form the basis for a sheriff pattern [31]. Similarly, the documented information about IT architecture was reviewed to find information how a concrete cloud provider or documented application addresses availability and resiliency [6]. Patterns from both domains have been reviewed during shepherding and writers workshops.

[3] http://www.cs.cmu.edu/~acme/

3.2.5 Pattern Language Revision

Problem: after multiple patterns have been extracted from a certain domain, they are already interconnected through references. These interrelations are commonly used to express that patterns are "often used together", form "alternatives", "compose" other patterns, etc. Through these references, a user of the resulting pattern language should be guided during the search for applicable patterns to his or her concrete use case (see pattern *search and recommendation* activity of the following *pattern application* phase). However, due to the iterative nature of pattern identification, patterns written early tend to have fewer references to other patterns as patterns written later. This is due to the fact that patterns written later have simply a lot more patterns to reference.

Solution: during this activity, the overall structure of a pattern language should be investigated using the same collaborative activities as are used during *pattern writing*. Especially, bidirectionality of references should be considered: if one pattern is, for example, often used with another pattern, is the same true for the inverse direction? If so, the reference should be added in the document of the referenced pattern as well.

Examples: we did several workshops with employees of collaborating companies that were users of the cloud computing patterns, i.e., employees who were not involved in the pattern writing. This helped us to identify how accessible the pattern language was and where its overall structure could be improved to help readers.

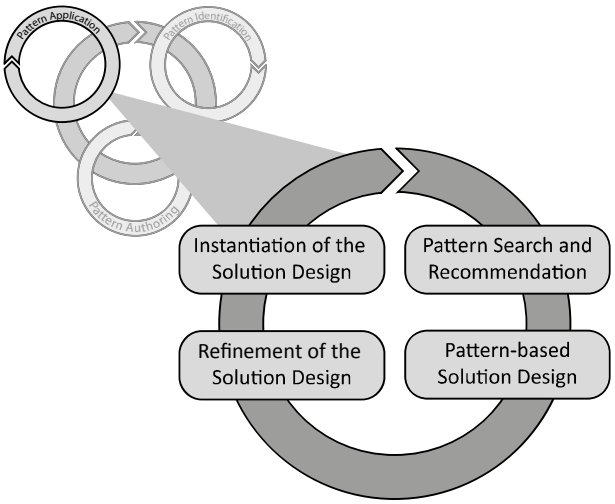

Figure 4: Detailed sub-activities of the pattern application phase © Christoph Fehling

3.3 Pattern Application

The pattern application phase shown in Figure 4 can be performed independently of the other two phases once there are patterns to be applied. It is comprised of the activities shown in Figure 1. Just like the other phases it can be performed iteratively in itself to create multiple solutions.

3.3.1 Pattern Search and Recommendation

Problem: the number of patterns identified during the previous phases or by other pattern researchers is very large. Given this set of patterns, a user has to find pattern suitable for his or her

problem at hand.
Solution: the pattern user review summarized information about patterns, such as icons, questions answered by the patterns, or intents in order to identify possibly applicable patterns. Then, the concrete pattern documents are read and references between pattern documents are followed to obtain the actually applicable pattern.
Result: the pattern users access summarized information and then refines his or her search by considering the whole pattern document. In many cases, this activity can be supported using recommendation tools that enable querying the pattern language and navigating through the interrelations among patterns.
Examples: for cloud data patterns [22], recommendation tools have been created [24] in the form of questionnaires that can be completed by users wishing to migrate their applications – and especially the data handled by them – to a cloud environment. Based on the answers given by users and annotations of possible answers to patterns, the users can be presented with a list of applicable patterns sorted regarding the likelihood of their applicability in the given use case. The cloud computing patterns are made accessible online[4] in form of a wiki to ease navigation between them.

3.3.2 Pattern-based Solution Design

Problem: patterns capture abstract solutions to be applicable in a general fashion. Therefore, a pattern user has to transfer these abstract solutions to his or her specific problem domain. Thus, the pattern needs to be implemented for the given use case.
Solution: the generality of the pattern achieved through the abstraction from concrete solutions is again refined to describe the tailored solution. During this activity, the pattern user may rely on the documented solutions from which a pattern has been abstracted.
Result: documented existing solutions or specifically created *reference implementations* are provided to the pattern user. He or she, thus, accesses the documented solutions that have been captured during the *information collection* phase to develop new solutions.
Examples: the cloud computing patterns contain references back to the known uses from which they have been abstracted. An Apache Maven[5] repository is used to host references implementations for these patterns [20]. A cloud computing patterns user may, thus, access code artifacts from this repository that may form the basis for pattern implementations using different cloud providers.

3.3.3 Refinement of the Solution Design

Problem: manual implementations tend to be different each time a pattern is refined to a concrete solution. Especially for large companies, it may be beneficial to control the pattern implementation to a larger extend by guiding and constraining the implementation and technology decisions made during the refinement of the pattern. One challenge faced by these companies regarding the management of IT infrastructures is to reduce the level of heterogeneity of infrastructure stacks powering their business applications. This heterogeneity is a major cost driver for IT management [12]. Unconstrained manual pattern implementations would lead to such heterogeneity.
Solution: patterns used by large companies are constrained with respect to the environment in which they shall be applied. For example, restrictions could state that only one or two types of IT infrastructure stacks and programming languages may be used for all implementations of a pattern.
Result: a company could decide to support one open source software stack and one proprietary software stack, each for different type of projects and their requirements. While this leaves the implementation of the pattern to the pattern user, he or she may significantly be supported by automated infrastructure management, deployment functionality, and code templates all of which was provided by the IT management departments. But even if the environment for which a pattern shall be implemented is not as constrained as those company-internal ones, the use of code template-based reference implementations is also helpful for other programmers.
Examples: pattern refinements of our industry partners may not be described here due to corporate regulations. The above mentioned Maven repository is used by us to manage reference implementations of the cloud computing patterns for different providers.

3.3.4 Instantiation of the Solution Design

Problem: if patterns are used based on reference implementations and constrained pattern refinements, the decisions to be made by pattern users are similar for each use case. The same is true for the following configuration of the homogenized environment in which a pattern shall be deployed.
Solution: techniques and tools for the management, configuration, and deployment of software applications are used to configure and deploy pattern refinements, thus, alleviating redundant manual tasks for pattern users.
Result: supporting tools may provide configuration of obtained hard- and software forming the runtime environment for the application. The configuration of the code templates themselves is also possible in this scope. Tooling for the guided configuration of these IT artifacts has, for example, been presented in [15] [20].
Examples: for the cloud computing patterns, Maven has been extended to handle the configuration of pattern implementations and the deployment of an implementation at a cloud provider. This especially, subsumed the configuration of reference implementation for different user credentials used to access cloud providers.

While these last two activities are clearly related to the refinement of pattern for the IT domain, similar support of pattern refinement could be realized in other domains. In the domain of costumes, this activity would subsume the ordering of clothes and existing costumes from rental agencies or similar institutions and then fitting them to the measurements of the concrete actor playing a role.

4. SUMMARY AND OUTLOOK

The presented pattern identification, authoring, and application process has been used in several pattern research domains. Cloud computing patterns describe how to design, build, and manage cloud applications. Cloud data patterns focus on providing best practices for challenges that may arise when migrating the data base layer of an application to the cloud. Application management

[4] http://www.cloudcomputingpatterns.org

[5] http://maven.apache.org

patterns cover how to handle applications during runtime using automated process models that are generated from planlets in a semi-automated fashion. Green business processes cover the environmental impact of a company's process models and runtime infrastructure by describing new patterns and green variants of already existing patterns. Finally, costume patterns describe reoccurring best practices to design costumes in movies in order to communicate certain character traits of roles. We argue that the diversity of these domains already ensures the generality of the presented pattern identification, authoring, and application process to some degree. With this paper we would like to document it for use in a larger pattern-research community where it is likely to be refined further. Our current ongoing work subsumes the creation of tools to support the different phases of this process. In the domain of costume patterns, work is ongoing on a repository for capturing costumes extracted from certain movies. Based on the data models and terminologies of this domain – the primitives used in the costume pattern language – we aim at finding reoccurring combinations of pieces of clothes in order to identify patterns. With a special focus on cloud computing patterns, we currently develop wiki-based tools to capture and manage patterns [8]. Especially, this wiki captures the semantics of interrelations between patterns and makes them accessible using special queries. This shall enable users of the wiki to find related or alternative patterns for a currently considered pattern more easily. Thus, the pattern language is managed as a fully navigable graph in the wiki. Finally, existing tools guiding the decision support aiming at the initial selection for applicable patterns from the cloud data patterns domain [24] should be integrated with this wiki in order to find suitable entry points to pattern languages using a wizard-based analysis of users' requirements.

5. ACKNOWLEDGMENTS

The work published in this article was partially funded by the Co.M.B. project of the Deutsche Forschungsgemeinschaft (DFG) under the promotional references SP 448/27-1, LE 2275/5-1 as well as the SitOPT project (Research Grant 610872, DFG) under the promotional reference LE 2275/9-1.

6. REFERENCES

[1] Alexander, C. 1978. *A Pattern Language: Towns, Buildings, Construction*. Oxford University Press.

[2] Barzen, J. 2013. *Taxonomien kostümrelevanter Parameter: Annäherung an eine Ontologisierung der Domäne des Filmkostüms (in German)*. Technical Report No. 2013/04. University of Stuttgart.

[3] Breitenbücher, U., Binz, T., Kopp, O., and Leymann, F. 2013. Pattern-based Runtime Management of Composite Cloud Applications. *Proceedings of the International Conference on Cloud Computing and Service Science (CLOSER)*.

[4] Fehling, C., Ewald, T., Leymann, F., Pauly, M., Rütschlin, J., and Schumm, D. 2012. Capturing Cloud Computing Knowledge and Experience in Patterns. *Proceedings of the IEEE International Conference on Cloud Computing (CLOUD)*..

[5] Fehling, C., Leymann, F., Retter, R., Schumm, D., and Schupeck, W. 2011. An Architectural Pattern Language of Cloud-based Applications. *Proceedings of the Conference on Pattern Languages of Programs (PLoP)*.

[6] Fehling, C., Leymann, F., Retter, R., Schupeck, W., and Arbitter, P. 2014. *Cloud Computing Patterns: Fundamentals to Design, Build, and Manage Cloud Applications*, Springer.

[7] Fehling, C., Leymann, F., Ruehl, S. T., Rudek, M., and Verclas, S. 2013. Service Migration Patterns - Decision Support and Best Practices for the Migration of Existing Service-based Applications to Cloud Environments. *Proceedings of the IEEE International Conference on Service Oriented Computing and Applications (SOCA)*.

[8] Fürst, N. 2013. *Semantisches Wiki zur Erfassung von Design-Patterns*. Diploma Thesis No. 3527. University of Stuttgart.

[9] Gamma, E., Helm, R., and Johnson, R. E. 1994. *Design Patterns. Elements of Reusable Object-Oriented Software*. Addison-Wesley.

[10] Hanmer, R. 2013. *Pattern-Oriented Software Architecture for Dummies For Dummies*. Wiley.

[11] Harrison, N. B. 1999. Language of Shepherding. *Proceedings of the Conference on Pattern Languages of Programs (PLoP)*.

[12] IBM. 2007. Corporate strategy analysis of IDC data.

[13] Lucrédio, D., de Almeida, E. S., Alvaro, A., Garcia, V. C., and Piveta, E. K. 2004. Student's PLoP Guide: A Pattern Family to Guide Computer Science Students during PLoP Conferences. *Proceedings of the Latin American Conference on Pattern Languages of Programs (SugarLoafPLoP)*.

[14] Meszaros, G. 1997. Pattern Language for Pattern Writing. *Pattern languages of program design 3*. Addison-Wesley.

[15] Mietzner, R. 2010. *A method and implementation to define and provision variable composite applications, and its usage in cloud computing.* Ph.D. thesis. University of Stuttgart.

[16] Nowak, A. and Leymann, F. 2013. Green Business Process Patterns - Part II. *Proceedings of the IEEE International Conference on Service Oriented Computing and Applications (SOCA)*.

[17] Nowak, A. and Leymann, F. 2013. Green Enterprise Patterns. *Proceedings of the Conference on Pattern Languages of Programs (PLoP)*.

[18] Nowak, A., Leymann, F., Schleicher, D., Schumm, D., and Wagner, S. 2011. Green Business Process Patterns. *Proceedings of the Conference on Pattern Languages of Programs (PLoP)*.

[19] OASIS. 2013. Topology and Orchestration Specification for Cloud Applications.

[20] Schraitle, A. 2013. *Provisioning of Customizable Pattern-based Software Artifacts into Cloud Environments*. Diploma Thesis No. 3468. University of Stuttgart.

[21] Schumm, D., Barzen, J., Leymann, F., Ellrich, L. 2012. A Pattern Language for Costumes in Films. *Proceedings of the European Conference on Pattern Languages of Programs (EuroPLoP)*.

[22] Strauch, S., Andrikopoulos, V., Bachmann, T., and Leymann, F. 2013. Migrating Application Data to the Cloud Using Cloud Data Patterns. *Proceedings of the International*

Conference on Cloud Computing and Service Science (CLOSER).

[23] Strauch, S., Andrikopoulos, V., Breitenbücher, U., Kopp, O., and Leymann, F. 2012. Non-Functional Data Layer Patterns for Cloud Applications. *Proceedings of the IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*.

[24] Strauch, S., Andrikopoulos, V., Bachmann, T., Karastoyanova, D., Passow, S., and Vukojevic-Haupt, K. 2013. Decision Support for the Migration of the Application Database Layer to the Cloud. *Proceedings of the IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*.

[25] Wellhausen, T., and Fießer, A. 2011. How to write a pattern? *Proceedings of the European Conference on Pattern Languages (EuroPLoP)*.

[26] Zdun, U. 2007. *Systematic pattern selection using pattern language grammars and design space analysis*. ACM Software.

[27] Mell, P., and Grance, T. 2011. The NIST definition of cloud computing. Available at: http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf.

[28] Petre, M. 1995. *Why looking isn't always seeing*. Communications of the ACM.

[29] Fehling, C., Leymann, F., Mietzner, R., and Schupeck, W. 2011. *A collection of patterns for cloud types, cloud service models, and cloud-based application architectures*. Technical report No. 2011/05. University of Stuttgart.

[30] Barzen, J., and Leymann, F. 2014. Kostümsprache als Mustersprache: Vom analytischen Wert Formaler Sprachen und Muster in den Filmwissenschaften (in German). *Jahrestagung der Digital Humanities im deutschsprachigen Raum (DHd 2014)*.

[31] Barzen, J., Leymann, F., Schumm, D., and Wieland, M. 2012. Ein Ansatz zur Unterstützung des Kostümmanagements im Film auf Basis einer Mustersprache (in German). *Modellierung 2012, GI*.

[32] Object Management Group (OMG): Unified Modeling Language (UML) Standard Version 2.4.1. Available at: http://www.omg.org/spec/UML/2.4.1/

[33] Bowman, M., Debray, S. K., and Peterson, L. L. 1993. Reasoning about naming systems. *ACM Trans. Program. Lang. Syst.*

All links were last followed on 30 January 2015.