Towards Extending Online Pattern Repositories: Supporting the Design Pattern Lifecycle

CHRISTIAN KÖPPE, HAN University of Applied Sciences, Netherlands
PAUL SALVADOR INVENTADO, School of Design, Carnegie Mellon University, USA
PETER SCUPELLI, School of Design, Carnegie Mellon University, USA
UWE VAN HEESCH, HAN University of Applied Sciences, Netherlands

Design patterns offer proven solutions to well-known problems and are an accepted way of knowledge preservation in various domains. Many design patterns have been documented and are published in a variety of places and formats. This makes it difficult for people to find patterns that offer solutions for a particular problem. Publishing the patterns in a repository seems to be a promising solution and, to date, many repositories have been set up containing patterns of various domains and offering a wide variety of functionality to support pattern users. However, none of these repositories seem to fulfill all user needs and there is certainly not “the one" common repository that serves as starting point for pattern search. We believe that the main reason is that all repositories are strong in certain aspects, but miss functionality in other aspects. In this work, we try to identify the most important parts of all relevant aspects, based on the existing work on pattern repositories. Building on that, we describe an initial set of high-level requirements for a generic online design pattern repository.

Categories and Subject Descriptors: D.2.10 [**Software Engineering**]: Design—*Design Patterns*

General Terms: Design, Languages, Requirement

Additional Key Words and Phrases: Pattern Repository

Reference Format:

Köppe, C., Inventado, P.S., Scupelli, P., van Heesch, U. 2016. Towards Extending Online Pattern Repositories: Supporting the Design Pattern Lifecycle – Proceedings of the 23rd Conference on Pattern Languages of Programs, PLoP ’16(October 2016), 26 pages.

1. INTRODUCTION

In 1987 Howard G. (Ward) Cunningham, then with Tektronix, and Apple Computer’s Kent Beck co-published the paper “Using Pattern Languages for Object-Oriented Programs" [Beck and Cunningham 1987]. This paper about programming patterns was inspired by Christopher Alexander’s architectural concept of "patterns". It was written for the 1987 OOPSLA programming conference organized by the Association for Computing Machinery. Cunningham’s and Beck’s idea became popular among programmers, because it helped them exchange programming and design solutions in a format that was easy to understand and contains all relevant information. In 1994, the Portland Pattern Repository[1] (PPR) became the first repository for computer programming design patterns. It was accompanied by a companion website, WikiWikiWeb[2], which was the world’s first wiki and allowed people to collaborate on pattern writing.

[1] `http://c2.com/ppr/`
[2] `http://c2.com/cgi/wiki?FrontPage`

Corresponding author: christian.koppe@han.nl

Also in 1994, the first Pattern Language of Programs (PLoP) conference was held. It evolved into an international conference series[3] with conferences such as EuroPLoP, AsianPLoP, SugarloafPLoP, and VikingPLoP, which are held all over the world. The main focus of these conferences is on promoting, improving and publishing patterns, and many patterns have been written since the first conference. The Pattern Almanac by Linda Rising [Rising 2000], published in 2000, contained an overview of 1.200+ patterns. In 2013, Hohpe and his co-authors estimated over 7.500 published patterns [Hohpe et al. 2013].

Patterns are published in conference proceedings, journals, and books, but also websites and online repositories or libraries. Each of these publication types has pros and cons. Peer-reviewed publications are usually of higher quality than patterns published on websites. However, these pattern publications stop evolving once they are published. They cannot be changed and adapted to changing environments or new insights. Existing websites and online repositories (partly) address this issue by supporting the evolutionary character of patterns [Reiners et al. 2013]. Publishing pattern versions online in a repository, in combination with a review process and quality gates, can support their aliveness, without compromising the quality of the pattern descriptions.

A long list of pattern repositories emerged in the last years, but most of them silently disappeared again. Repository systems such as the Portland Pattern Repository, PATONGO , PatternPedia, or the Open Pattern Repository offer extensive support for pattern authoring and other parts of the pattern lifecycle, but none of these are widely used (or not used at all) by the community around the various PLoP pattern conferences. However, all of them contain features that are relevant requirements for a system that would support the pattern community in all activities related to the design pattern lifecycle (pattern writing, application, evaluation, evolution etc.).

We think that there are two main reasons for this phenomenon of failing attempts to establishing a pattern repository as supporting system in the pattern community: (1) an experienced lack or inappropriateness of functionality required for supporting *all* activities in the pattern lifecycle and the shepherding process [Harrison 1999] and (2) the lack of a community of stakeholders that actively evolves the repository and keeps it alive through adding or updating content. Certainly, the latter reason is impacted by the former.

In this paper, we address the first reason with the goal to identify requirements and features for a pattern repository which

—support all phases of the design pattern lifecycle (such as pattern writing, evolution, validation, application),

—can additionally support the processes as established in the pattern community around the PLoP conferences (such as shepherding/review),

—cover the valuable features of existing repositories (such as browsing/searching, scenarios, examples, feedback possibilities, versioning, continuous improvement/refinement, interlinking between patterns etc.),

—can be used for adapting existing repositories so that they could be used for xPLoPs, Hillside etc. or as meta-repository with links to (patterns in) other repositories.

For achieving this goal, we use the results of a focus group on online pattern repositories, held at the PLoP'15 conference. We identify the potential stakeholders of a pattern repository and their needs. Based on these needs, we analyze existing literature on pattern repositories as well as existing repositories themselves to extract aspects of functionality that are relevant and required for fulfilling the stakeholder's needs.

As a next step we will explore the community aspects necessary for establishing a pattern repository with an active community, both generically and by the example of design patterns for pedagogy, which we plan to present in future work.

The rest of this paper is organized as follows: In the next section, we give a summary of the results of the PLoP'15 focus group on “Developing and Open, Collaborative Design Pattern Repository". After that, we identify the stakeholders of a pattern repository and their needs. Then we present relevant literature and existing repositories in Section 4, which is accompanied by a more extensive and structured summary in the appendix. Based on these two sections, we present

[3]`http://www.hillside.net/conferences`

a set of requirements which are relevant for addressing the stakeholder's needs in Section 5. The last section provides a conclusion and an outlook on future work.

2. PLOP'15 FOCUS GROUP RESULTS

In PLoP 2015, the focus group entitled “Developing an Open, Collaborative Design Pattern Repository" was conducted to get insights from design pattern authors regarding their thoughts on the hindrances to building pattern repositories and possible ways to address them. These ideas are summarized below.

2.1 Locating patterns

Thousands of patterns have already been developed for multiple domains. It becomes harder for pattern authors and users to find patterns that are relevant to them. When users go into a repository and find it difficult to locate what they need, they may be less interested to come back and use that repository.

It is common for pattern repositories to provide text-search facilities and support hyperlinks between related patterns to make them easier to navigate. However, these pattern-location methods require a certain level of understanding about the domain of interest, some knowledge of patterns that already exist, and familiarity of pattern formats and the kind of information they contain. It may be useful to provide pattern-location methods that search over pattern languages or pattern categories to filter the search space and make it easier to locate relevant and related design patterns. Faceted search might be a good option here to explore.

2.2 Pattern Fragmentation

Related to searching for patterns is the fragmentation of design pattern resources. Design patterns may be found in books, publications, and websites aside from pattern repositories. These patterns need to be encoded into repositories to take advantage of pattern repository facilities, but it takes a lot of effort to locate, encode, sort, and organize these design patterns.

2.3 Pattern Repository Maintenance

Many pattern repositories were developed as part of research projects. Maintenance stops when a student managing the repository graduates, a funded project that developed the repository is completed, or a sponsoring group loses interest in the repository.

A central repository may help maintain interest in design patterns that belong to inactive repositories by assimilating them into the repository so they can be searched and linked to other design patterns.

2.4 Pattern Presentation

Pattern authors have their own preferences on how to express patterns. Their preferences may be based on the domain they are working on (e.g., programming codes are incorporated in a software pattern), their intended audience (e.g., short descriptions make it easier for teachers to use patterns), their experience with patterns (e.g., novice pattern authors may find it easier to read and write patterns organized into sections), and so forth.

It is difficult to identify a general pattern format to accommodate the needs of pattern authors. It may be a good idea to take a bottom-up approach by taking a template that works for one group of authors and expanding it to accommodate pattern formats from more groups. A simpler approach may be to support multiple pattern formats, but keep a core set of elements shared across all pattern formats (e.g., context, problem, forces, solution).

2.5 Copyright and Ownership

There may be legal issues with uploading design patterns to repositories when they have been published in proceedings, journals, books, or other copyrighted medium. There may also be issues with uploading design patterns developed through funded projects due to funding agency requirements or limitations. For example, the data needs to be hosted in a server within the country that funded it.

A possible solution to avoid copyright or ownership issues may be to allow uploading pattern abstracts or pattern descriptions instead of the actual patterns and providing links to the original source. It may also be a good idea to prefer publishers that allow pattern authors to retain copyright over their patterns. For example, most PLoP conferences are in agreement with ACM to give authors the option to keep their copyright.

2.6 Pattern Evolution

Some design patterns may benefit from continued refinement. For example, new findings may require the addition of forces, the modification of the solution, or the inclusion of more known uses of the pattern. It will be useful to keep track of information regarding the changes made to the pattern, the justification for the changes, and the contributing authors.

2.7 Pattern Quality

The usefulness of a pattern repository is certainly related to the quality of patterns it contains. While some agree to only accept patterns of a certain quality, others argue that enforcing a strict evaluation process will discourage authors to submit patterns. Selecting the acceptable level of quality or assigning moderators tasked to assess the quality of submitted patterns are complicated issues. Furthermore, there are different aspects of the pattern that can be evaluated. For example, will quality refer to the readability of the design pattern or the effectiveness of the design pattern when applied in actual cases.

There were several suggestions to address this problem. First, a quality measure may be based on the number of people who find the pattern useful or the number of positive reviews on the pattern. Reviewers may be ranked to consider the weight of their review. This is a similar measure used by systems that involve customer reviews (e.g., Amazon shopping, TripAdvisor, Yelp). Usage metrics may also be used to measure quality (e.g., frequency of actual pattern use). The use of such measures requires much care, however, to avoid assigning high scores to design patterns that are frequently used because of their simplicity despite having relatively low impact; and assigning low scores to patterns that are less-frequently used because of their complexity despite having relatively high impact.

2.8 Community-related Issues

There were other issues discussed during the focus group that required the community's involvement to resolve. For example, most pattern authors choose to submit their patterns to PLoP conferences because it does not only allow them to share their patterns and get feedback, but it also gets their papers published, which they can use towards their promotion or to satisfy funding requirements. However, there seem to be few incentives for pattern authors to submit their patterns to pattern repositories. Similarly, patterns may need to be updated after they are published (e.g., additional known uses, link to new related patterns, improvements in presentation), but it is difficult to encourage authors to do so. Thus, there is a need to find creative ways to encourage pattern authors to upload and update their patterns.

Another issue is the case when pattern authors think of ways to refine their own patterns or other authors' patterns, but run into legal issues because of copyright and ownership concerns. Standards may need to be developed to ensure continued development and improvement of design patterns.

Finally, allowing multiple authors to refine and extend existing design patterns leads to issues regarding attribution. Methodologies and standards may need to be defined to ensure authors are properly attributed for their work and to encourage collaboration among various stakeholders.

These challenges are important to promote the use of the repository, but are outside the scope of this paper. Functionalities discussed in the paper might help enable the community to resolve these issues, but these will be addressed specifically in future work.

3. PATTERN REPOSITORY STAKEHOLDERS AND THEIR NEEDS

For identification of the repository stakeholders, we used as starting point the work of Derntl and Botturi on essential use cases for pattern systems [Derntl and Botturi 2006]. Even though their work is focused on the domain of pedagogical patterns, the functionality they describe is very generic and therefore likely applicable to other pattern domains too. We did not find conflicts or discrepancies during the analysis of relevant literature and existing repositories. Derntl and Botturi identified four actors in a pattern system:

—**Author** - writes the patterns
—**Maintainer** - maintains the pattern system (authors are also maintainers)
—**Analyst** - uses the patterns to analyze design artefacts
—**User** - all other actors plus people who apply the patterns

The **User** actor defined by Derntl is quite generic. We think there may be more specific roles to address the needs of a pattern system such as:

—**Reviewer** - a user who reviews patterns (and other repository content) and provides feedback, additional examples, or suggestions for improvement; shepherds are specific instances of such reviewers
—**Domain Expert** - a user with deep knowledge of a certain domain and can identify elaborate scenarios that may exist in that domain (e.g., combinations of patterns); the domain expert may not necessarily be a pattern author
—**Domain Novice** - a user new to design patterns who may need help to navigate the large amount of information on design patterns; novice lecturers in the domain of education are examples of novice users who may need help to find information related to their teaching tasks

Derntl and Botturi described a set of essential use cases for a pattern system, namely *Write Pattern*, *Revise Pattern*, *Remove Pattern*, *Maintain Patterns* (as abstract use case for the previous three ones), *Browse Patterns*, *Identify Pattern*, *Apply Patterns*, and *Analyze an Artifact*. While the last three are certainly relevant for a pattern system, they are not directly part of the functionality of a pattern repository, but are executed outside of it. Still, the repository needs to take them into account by providing the necessary information (*Apply Patterns* and *Analyze an Artifact*) or by offering a way of connecting the results of them with the repository (*Identify Pattern*).

From our analysis of existing repositories and relevant literature, we used the following simplified features to cover the core functionality described by Derntl and Botturi as well as additional features we believe are essential to a pattern system:

—**Repository Content** - the information included in the repository and the way this is structured
—**Pattern Writing** - all aspects of adding/updating patterns and related information in the repository
—**Browsing/Searching** - all aspects of finding relevant patterns and the presentation of them
—**Other Requirements** - all other aspects that do not fall into one of the other categories, e.g. import/export functionality, reviewing aspects, evaluation/validation, or layout/visualisation aspects
—**Comments** - relevant information which is not directly related to requirements

4. PRESENTATION OF RELEVANT LITERATURE AND EXISTING REPOSITORIES

In this section, we introduce related work we used to identify the requirements of a pattern repository. A summary of the relevant properties can be found in the appendix.

Cunningham and Mehaffy show that a wiki offers a good base for a design pattern repository, as wikis and pattern languages share fundamental structural characteristics such as being open-ended sets of information (consisting of unitary subsets), being topical essays with a characteristic structure (easy to create, share and edit by many people), in principle evolutionary, falsifiable and refinable, and aim to create useful ontological models of a portion of the world [Cunningham and Mehaffy 2013]. In fact, wikis were initially developed as a tool for supporting the development of pattern languages (mainly for software)[4].

[4]Ward Cunningham interview with Michael Mehaffy, Youtube video from "Lightning Interview Series" 2009: `https://www.youtube.com/watch?v=Fyc1JGXP-hc`, accessed July 17, 2016

Fehling et al. introduced *PatternPedia*: “a collaborative tool chain to document existing solutions and manage patterns abstracted from them" [Fehling et al. 2015]. In their work, they present an extensible pattern metamodel (including typed references such as *InContextOf*, *ConsiderAfter*, or *Alternative*) where concrete repository implementations can be based on.

In 2010, Birukou did a structured survey of existing approaches for pattern search and selection [Birukou 2010]. For the survey, he identified potential problems of pattern repositories and documented a set of features and properties of pattern repositories and catalogs (available in 2010), which potentially help to solve these problems. The results show that the Open Pattern Repository (discussed below) contains most of the valuable features and properties. However, the described features and properties can also be seen as valuable part of requirements for a pattern repository and are therefore included in our work.

Both Mundie et al. [Mundie et al. 2012] and Köppe [Köppe 2013] suggest to use a faceted classification and the facet map approach for better support of browsing and searching for patterns in an online repository. Facet maps provide a mechanism to filter resources (such as patterns) based on selections of combinations of resource-specific attributes. These attributes are grouped in facets according to classifiable characteristics. This approach requires a multi-dimensional categorization such as the one proposed for security patterns [VanHilst et al. 2009]. The more generic Facet-Map based search itself is described in [Smith et al. 2006]

Reiners et al. focused on the collaborative formulation process of evolutionary patterns and described high-level requirements that support such process [Reiners et al. 2013]. The results were applied in the development of the BRIDGE[5] pattern library, where additionally also a workflow for evolutionary pattern formulation was introduced. Requirements which do directly address functionality are summarized in the appendix and reflected in this work.

Inventado and Scupelli [Inventado and Scupelli 2015b] investigated the design of an open, collaborative repository whose goal is to foster collaboration between design pattern authors and stakeholders throughout the design pattern lifecycle. The design of the repository is tied to a data-driven design pattern production (3D2P) methodology, which utilizes data collected from existing systems to implement a five-step iterative process of (a) pattern prospecting, (b) pattern mining, (c) pattern writing, (d) pattern application, and (e) pattern evaluation and refinement [Inventado and Scupelli 2015a]. Stakeholders are envisioned to collaborate with each other and design pattern authors in each of these processes. A prototype of their open, collaborative repository[6] is currently available to interested parties.

Pavlic et al. [Pavlič et al. 2009] describe an ontology-based repository system that makes use of semantic web technologies in order to also support question-answer expert systems and full-text search.

Schümmer and Haake introduce PATONGO (Patterns and Tools for Non-Profit Organizations), a pattern-based approach for helping voluntaries to identify and share good practices [Schümmer and Haake 2010]. They focus on organizational learning and communication between repository users (with emphasis on learning inside of an organization). The described process supports various levels of abstractions, starting with challenges and ideas, then good practice descriptions, pattern descriptions and overview articles with introductory overviews of the patterns of a specific domain and also successful applications of combinations of patterns (we will call these scenarios in this work).

Finlay et al. present the Pattern Language Network (PLaNet) for Web 2.0 in Learning [Finlay et al. 2009]. This project included a participatory pattern workshop methodology for mining patterns from domain experts (with the focus on learning through web 2.0 technologies) and the development of a collaborative software platform to facilitate community based pattern creation and use. The source code of the final version of the platform is available for download[7].

Birukou and Weiss describe a service for selecting patterns [Birukou and Weiss 2009]. This service makes use of the content of existing pattern repositories and adds functionality that is not included in them, such as recommendations based on various criteria.

[5]`http://bridge-pattern-library.fit.fraunhofer.de/pattern-library/`

[6]`http://www.learningenvironmentslab.org/openpatternrepository/`

[7]`https://code.google.com/archive/p/patternlanguagenetwork/`

Fincher specifies the Pattern Language Markup Language (PLML), which is the result of a CHI2003 workshop [Fincher 2003]. PLML describes an XML-format that is supposed to serve as common pattern format for HCI patterns (and is likely also appropriate for other domains according to the author). It contains a set of essential elements and sub-elements (see related work table in Appendix) and aims at providing a format that can be used for interchanging their pattern descriptions and as basis for pattern collections.

Deng et al. discuss “the main requirements for a tool to be used by researchers and user interface designers that can manage a repository of possibly disparate pattern collections" [Deng et al. 2005]. They build on existing specifications for UI pattern tools and define a feature framework, which addresses identified problems when building pattern management tools. The features comprise authoring, manipulating forces, browsing, searching, modification (versioning and customizing), relating patterns, manipulating collections, and input/output (see the table in Appendix A for a more detailed description of the features).

Welicki et al. introduce the Entity Meta-Specification Language (EML) for pattern specification [Welicki et al. 2006]. EML can be used for describing patterns from different languages and in different formats using the same semantic and syntactic elements. A pattern is hereby defined as an entity with properties, and the properties are the specific sections of the pattern. They show how EML can be used for specifying a pattern catalog and introduce a web-based catalog visualization tool that allows searching, linking and using the patterns in a catalog.

Weiss and Birukou propose using wiki as repository platform, as this offers some technical and social benefits due to the open, lightweight, and participative nature of wikis [Weiss and Birukou 2007].

Van Heesch and colleagues developed the *Open Pattern Repository*[8], which supports most activities of the pattern lifecycle. The source code and documentation of the repository is publicly available and the repository is used by different institutions for internal pattern documentation. A publicly deployed version is currently not available.

The Integrated Learning Design Environment[9] (ILDE) was the result of the European project Metis[10] on Lifelong Learning. It offers a place for capturing Learning Design Solutions (LdS) with varying levels of abstraction and scope such as design patterns, course maps, scenarios, user stories etc.. It is community-focused and offers various ways of interaction such as reviewing, commenting, adding your own LdS, integrating exisitng LdS into scenarios etc.. At the time of writing (september 2016), ILDE does not contain much mature content.

The Placepatterns[11] repository contains architectural patterns similar and additional to the ones described in *A Pattern Language* [Alexander et al. 1977]. It uses the same Alexandrian format and contains the patterns as published in a book (no linking or followable references). Browsing is possible via tags and scale and places can be added which have the patterns.

The Liberating Voices Pattern Language focuses on positive social change and is part of the Public Sphere Project[12]. The patterns are published in a book [Schuler and Douglas 2008] and also (slightly altered) online in an online repository[13]. At this moment (september 2016) the patterns are only published as is, but the website states that in the future they will also work on allowing comments on patterns (including questions, suggestions, examples of pattern usages, and relevant references), adding functionality so that people and groups can develop their own pattern languages (allowing re-using existing patterns and addition of new patterns), and a general software effort to support all of the patterns (similar to our work).

The BRIDGE pattern library[14] was developed to better handle knowledge acquisition and transfer between different (research) project partners and work packages. The repository supports an extensive workflow called *Evolutionary Pattern Formulation*, with varying degrees of maturity of the patterns based on reviews, feedback and comments of

[8]`https://github.com/wizzn/openpatternrepository`
[9]`http://ilde.upf.edu/`
[10]`http://www.metis-project.org`
[11]`http://placepatterns.org`
[12]`http://publicsphereproject.org/`
[13]`http://publicsphereproject.org/patterns/LV`
[14]`http://bridge-pattern-library.fit.fraunhofer.de/pattern-library/`

other users and experts. The functionality also supports community aspects with e.g. a *Hall of Fame* for the most active contributors or the presentation of a list with patterns that need to be rated or need evidence.

Martijn van Welie hosts an online pattern library with patterns in interaction design[15]. The patterns are presented as is, but visitors can comment on them. No information is available about latest changes or if the library is actively maintained and updated.

5. COMMON REQUIREMENTS FOR THE DESIGN PATTERN PROCESS

The requirements presented in this section are based on the reviews of existing repositories and related literature. Per publication we summarized all relevant features, grouped by different aspects of the pattern life cycle. Furthermore, existing pattern repositories were analyzed regarding the features they offer and the findings documented. All results were included in the summary table (in Appendix A). Based on these results, we identified requirements that subsume all features of existing repository approaches. These requirements can then be used for ensuring that a pattern repository—either existing ones such as the Open Pattern Repository, PatternPedia, or PATONGO, but also new repositories—supports the whole design pattern lifecycle.

The presented requirements are grouped by different aspects of the pattern life cycle, consistent with the summary table.

5.1 Pattern Writing Requirements

Patterns can be written by individual authors, but it's more likely that pattern writing is a collaborative effort. This can be either through a group of authors and/or the combination of author/s and shepherd/s. In that way the repository can also be used for shepherding, for example, as part of the xPLoP process.

Collaborative authoring can be realized by allowing a defined group of authors to work on the patterns or by making one author responsible for adapting the pattern based on the suggested adaptations by the other authors.

The repository should support various collaboration activities within the community, which may include communication, collaborative editing, and versioning (see separate section below).

5.1.1 *Authorship.* Patterns can have one or more registered authors. The author/s are the only users that are allowed to change the content of the patterns. Other users can make suggestions for improvements. Subsequently, the authors can accept suggestions by improving the pattern based on them, getting more clarification from the suggesting user if needed, or rejecting them (with the reasons communicated to the user who made the suggestion).

5.1.2 *Pattern Status.* In order to support the pattern writing process, each pattern is always in one specific status. Different repositories made use of different statuses (such as seed idea/alpha/beta in PLaNet or "pattern under consideration" / "pattern candidate" / "approved pattern" in the BRIDGE Pattern Library). The concrete statuses (and likely the corresponding workflow) still need to be discussed in the community. It might also be possible that different sets of statuses and workflows will be defined for different domains or kinds of patterns. We therefore propose that statuses can be flexibly defined.

Aside from pattern status, the maturity of published patterns may also be defined (similar to the star system Alexander used) based on different sources such as user ratings or expert judgments. See the section on ratings and validation for more details.

5.1.3 *Shepherding.* A repository that covers the complete pattern life cycle could also be used for supporting the shepherding process as established in the pattern community. Authors can add pattern drafts to the repository and open that for the shepherd to make suggestions for improvement, hereby improving the pattern/s iteratively. Functionality is required that allows exporting the patterns to e.g. Word or Latex in a desired format so that they can be included in the conference submissions (see section on export/import functionality). A limitation is that usually submissions to

[15] `http://www.welie.com/patterns/`

conferences do not only contain the patterns themselves, but also some introduction, background information, running examples etc. One way of handling this limitation could be to make use of the proposed overview articles inside of the repository, which include also the described patterns and can serve as paper outlines. However, this likely requires some more effort and still needs to be discussed by the community.

An advantage would be that by using the repository also for shepherding, the patterns would already evolve *inside of the repository* and the final versions are therefore automatically included in it (besides also being published e.g. in the proceedings of a xPLoP or other conference). This automatic inclusion removes the earlier described hurdle of having to do an extra effort to add the pattern to a repository *after* having it finished for some proceedings.

5.2 Pattern Application Requirements

Authors may be interested to see who used their patterns, how many times it has been used, where their patterns have been used, when it was used, how it was applied, and so forth. The information can easily be stored into the repository, but a bigger challenge is encouraging pattern authors or stakeholders to update the system. This is outside the scope of this paper, but we plan to address this in future work.

5.3 Pattern Evaluation Requirements

Evaluating patterns involves many complex issues, which may need to be resolved before a system can be developed to support it. Some issues that we think might need to be addressed would include:

—pattern-evaluator requirements and selection process (e.g., pattern-writing experience, knowledge of the domain, community participation)
—appropriate pattern evaluation process (e.g., peer-review, expert-review, forums)
—appropriate pattern evaluation metrics (e.g., pattern clarity, effectiveness in actual applications, usage frequency)
—requirements for evaluation contexts (e.g., pattern usage in diverse domains, pattern usage by a large population, pattern usage over an extended period of time)

The issue of an appropriate evaluation process requires the functionality to provide feedback and comments on patterns, independent of who is doing the review or commenting. Such general functionality is included in our requirements.

The issue of evaluation metrics is addressed by adding a rating functionality that allows various rating possibilities. It is furthermore possible to collect user data such as if a pattern has been applied or the number of provided example applications.

The other issues are outside the scope of this paper, but we plan to address them in future work.

5.4 Pattern Evolution Requirements

All registered users can suggest improvements or add new examples to an existing pattern. For example, a teacher who applied a pedagogical pattern in class and found the pattern effective can share his/her experience with the author, which the author can decide to include in a pattern's list of known uses. It may also be possible that a teacher is unable to replicate the benefits of applying the pattern, which can be clarified with the author to identify if the issue was with the pattern's application or if the issue was with the design pattern that needs to be addressed by the author.

Issues and suggestions need to be verified and, if valuable, addressed by the author/s. If the author/s do/es not agree with the suggestions or has questions about them, they should be discussed. All suggestions need to be either incorporated or declined with good reason. All made suggestions for a pattern show up in the activity list of the author of this pattern (or if the author is part of the assigned author group).

5.5 Repository Content

5.5.1 *Multiple Pattern Structures and Pattern Views.* The format of a pattern can be defined as the set of semantically coherent and named sections of a pattern. Pattern authors often select pattern formats according to their personal preferences, but there are cases when certain pattern formats are more appropriate for a domain. For example, domains that need elaborate descriptions may benefit from pattern formats that split content into multiple sections. Pattern readers may also have their own preferences, which could be different from that of the pattern author. For example, a novice pattern author or someone interested in applying a design pattern may have less experience with design patterns and may find it difficult to understand the Alexandrian form, but find it easy to understand a sectional format.

In many repositories (appr. 50% of the ones we analyzed), there is one pattern format defined that contains both the set of sections of a pattern *and* their visual representation. We believe that this is not sufficient for a pattern repository that aims at including patterns of different domains and different authors. Such repository needs to support multiple pattern formats. Furthermore, consistent with [Welicki et al. 2006], we propose to separate the information and the representation of a pattern in order to allow multiple representations of the same pattern, all based on the same information.

We there therefore propose the two concepts of *Pattern Structure*, which contains information on how the pattern content is divided into separate sections (perhaps of specific types such as text, images, multimedia data, or combinations of these), and *Pattern View*, which defines how (and which of) the sections are visually represented. Sections can also be divided into sub-sections, e.g. for specific parts of a more generic section (like “Core" and “Details" for a “‘Solution" section, “example" and “rationale" in an “evidence" section as in PLML, or for lists of specific forces or consequences).

An example of a pattern structure for a pedagogical pattern could be: Name, Summary, Context, Problem, Forces, Solution, Solution Details, Benefits, Liabilities, and Examples. Figure 1 shows how a pattern that is described using that structure could be represented in different ways (Alexandrian-inspired, with heading and as pattlet).

The list below highlights some benefits of supporting multiple pattern structures and pattern views:

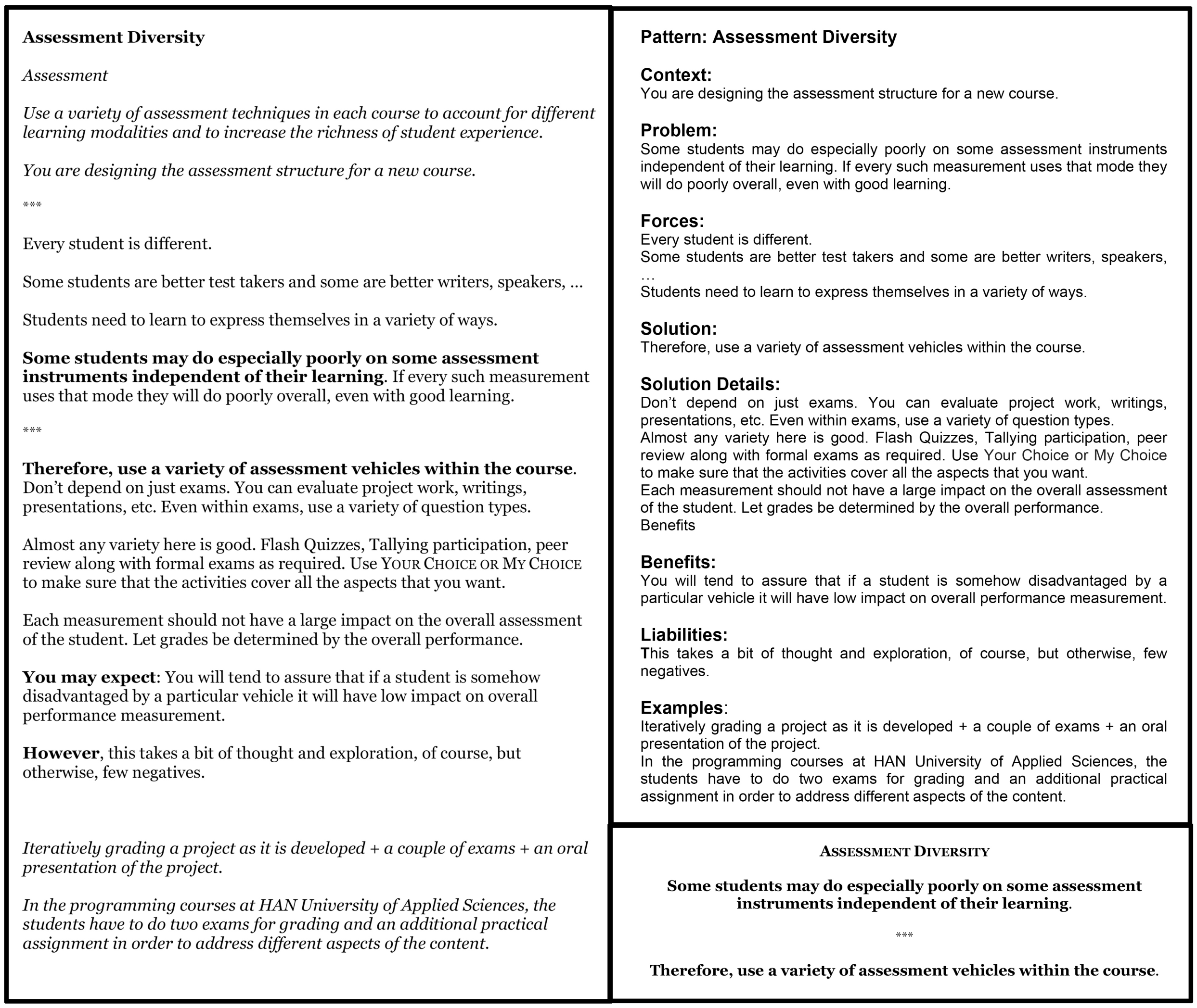
—pattern authors are not limited to a particular pattern format (structure and view), which may encourage them to submit their patterns into the repository
—several pattern variations written using different structures can be associated with the pattern author’s *source pattern*
—pattern readers may choose to view the pattern using the view that is easiest for them to understand or utilize (e.g., learning scientists might want to see literature references in pedagogical patterns, while practitioners might want to see a summarized rationale; novices may want to see examples first before other sections etc.)
—pattern views for specific use-cases may be created (e.g., pattern formats that are easily viewed on small screens – mobile phones vs. tablets vs. desktop computers; pattern formats that are printer-friendly; pattern formats that can be printed into pattern cards)
—clearly-defined pattern structures may facilitate translation between structures and even the possibility of automatic pattern translation

The definition of a pattern format needs to include a name for the format, the obligatory sections, and, if appropriate, the content format per section. The actual content of each of these sections should be freely definable and can comprise text, pictures, or other digitally storable information. It is important to note that the definition of the pattern sections should be driven by the semantics of each pattern part and not just the visual layout.

There are three pattern sections that are obligatory for all formats: *name*, *summary*, and *original source*. Having these sections ensures that all patterns have this common base and all patterns can be included, independent of the copyrights of the initial publication (as a self-written summary including a reference to the original publication will always be allowed). Additionally, all patterns need to be connected to at least one author.

A pattern author who submits a pattern to the repository can upload a pattern using any preferred format with the obligatory sections. The source pattern can be freely modified by the author. Other pattern authors or pattern readers who wish to view the pattern in other formats can submit pattern variations following other pattern formats. People accessing the repository may then see the source pattern and variations to that pattern. They can easily switch between variations to choose their preferred format.

5.5.2 *Pattern View.* A pattern view defines how a pattern will be visually represented. It hereby makes use of the semantic information contained in the pattern structure, e.g. that a pattern section contains the problem statement and that this section should be represented using a bold font. For each pattern structure, multiple views can be defined



Assessment Diversity

Assessment

Use a variety of assessment techniques in each course to account for different learning modalities and to increase the richness of student experience.

You are designing the assessment structure for a new course.

***

Every student is different.

Some students are better test takers and some are better writers, speakers, ...

Students need to learn to express themselves in a variety of ways.

Some students may do especially poorly on some assessment instruments independent of their learning. If every such measurement uses that mode they will do poorly overall, even with good learning.

***

Therefore, use a variety of assessment vehicles within the course. Don't depend on just exams. You can evaluate project work, writings, presentations, etc. Even within exams, use a variety of question types.

Almost any variety here is good. Flash Quizzes, Tallying participation, peer review along with formal exams as required. Use YOUR CHOICE OR MY CHOICE to make sure that the activities cover all the aspects that you want.

Each measurement should not have a large impact on the overall assessment of the student. Let grades be determined by the overall performance.

You may expect: You will tend to assure that if a student is somehow disadvantaged by a particular vehicle it will have low impact on overall performance measurement.

However, this takes a bit of thought and exploration, of course, but otherwise, few negatives.

Iteratively grading a project as it is developed + a couple of exams + an oral presentation of the project.

In the programming courses at HAN University of Applied Sciences, the students have to do two exams for grading and an additional practical assignment in order to address different aspects of the content.

Pattern: Assessment Diversity

Context:
You are designing the assessment structure for a new course.

Problem:
Some students may do especially poorly on some assessment instruments independent of their learning. If every such measurement uses that mode they will do poorly overall, even with good learning.

Forces:
Every student is different.
Some students are better test takers and some are better writers, speakers, …
Students need to learn to express themselves in a variety of ways.

Solution:
Therefore, use a variety of assessment vehicles within the course.

Solution Details:
Don't depend on just exams. You can evaluate project work, writings, presentations, etc. Even within exams, use a variety of question types.
Almost any variety here is good. Flash Quizzes, Tallying participation, peer review along with formal exams as required. Use Your Choice or My Choice to make sure that the activities cover all the aspects that you want.
Each measurement should not have a large impact on the overall assessment of the student. Let grades be determined by the overall performance.
Benefits

Benefits:
You will tend to assure that if a student is somehow disadvantaged by a particular vehicle it will have low impact on overall performance measurement.

Liabilities:
This takes a bit of thought and exploration, of course, but otherwise, few negatives.

Examples:
Iteratively grading a project as it is developed + a couple of exams + an oral presentation of the project.
In the programming courses at HAN University of Applied Sciences, the students have to do two exams for grading and an additional practical assignment in order to address different aspects of the content.

ASSESSMENT DIVERSITY

Some students may do especially poorly on some assessment instruments independent of their learning.

***

Therefore, use a variety of assessment vehicles within the course.

Fig. 1: Three different views of the same pattern structure (adapted from [Bergin et al. 2015]).

where one view will serve as default (can be changed by user). An example for three different views of the same pattern (structure) is shown in Figure 1.

The definition of a specific pattern view comprises following parts:

—To which pattern structure the view belongs,

—if it is the default view (can be changed by the user),

—the selection of sections, their order and optionally separators between the sections, and

—the layout per section, incl. information on how to treat specific section elements.

5.5.3 *Pattern Relations.* The power of patterns is even stronger when applied in combinations, likely as part of pattern languages. However, these relations contain more information than simple hyper-linking does. Providing more

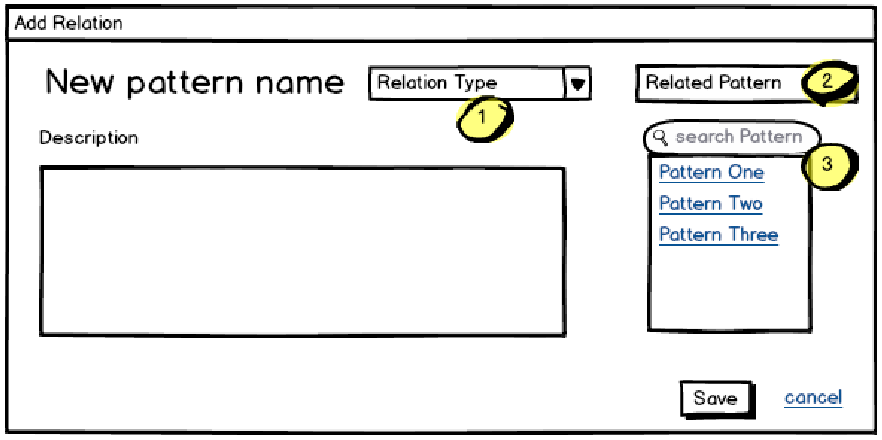


Fig. 2: Example UI sketch for adding a relation to another pattern: (1) the relation type can be selected, (2) the related pattern, (3) a search support for finding the correct related pattern. Furthermore, additional information about the relation can be provided.

information about pattern interrelationships can help pattern authors and stakeholders find relevant patterns. Properties of these relations are the relation type, the direction, and, if necessary, additional information.

The relations are included as part of the pattern section content (see YOUR CHOICE OR MY CHOICE in the examples in Figure 1) and can be added with extra functionality as shown in Figure 2.

Adding relations to patterns is part of pattern writing and therefore has to be done by the author/s. However, as other users can suggest improvements in general, they also can suggest adding missing relations to other patterns. It is then the responsibility of the author to incorporate these suggestions into the pattern content if appropriate.

5.5.4 *Pattern Categorization.* Nearly all pattern collections provide some categorization of the patterns. These categorizations hereby reflect different aspects of the patterns, such as quality goals, moment of implementation, context-related aspects, or properties of the elements involved in the solution. In some cases the categorization is very flexible and unorganized, using a simple tagging mechanism. Other cases provide a more organized categorization system, where mostly a couple of (sometimes distinct) categories are grouped together under a higher level topic, sometimes defining a deeper hierarchy too. The number of categories (or dimensions) varies from few (such as two categories in the GoF book [Gamma et al. 1994]) to many (such as six for security patterns [VanHilst et al. 2009] or more than six in the PLaNet repository [Finlay et al. 2009]).

In order to reflect these forms of categorization, we propose a hierarchical organization into key categories (classifiable characteristics) and specific categories (attributes of such characteristic). A specific category hereby can also become a lower-level key category in the hierarchy, e.g. the specific category “Lecturing" as part of the key category “Educational Methods" also can be a key category on a lower level with specific categories such as “Lecture Design", “Lecture Execution", or “Lecture Evaluation".

Finding suitable categories is not easy and requires involvement of experts and/or a community of practice. In many cases, some of these key and specific categories are implicit to the original publication of the patterns. For example, the patterns in a paper on lecture design are implicitly mapped to the key category *Lecturing*, which is part of the key category *Educational Methods*. The Fearless Change patterns [Manns and Rising 2005] do contain categories on the moment of implementation, but no explicit categorization as “organizational" patterns or patterns for “change management".

Having such an extensible categorization system based on key categories and specific categories that can dynamically reflect various aspects of the patterns offers a good base for various search and filter mechanisms, such as FacetMaps [Smith et al. 2006].

5.5.5 *Pattern Versioning.* Versioning is important in most collaborative projects because it keeps track of specific changes in the document, the contributor who made the change, and provides facilities to revert the changes. Pattern authors whose patterns are linked to a changed pattern need to know about such changes to make sure their patterns remain coherent and updated. It is also important to provide proper attribution for authors who contributed to the pattern's development. Finally, it is convenient to provide a facility for reverting changes to allow the review and reuse of previous versions.

5.6 Requirements related to using the patterns: Browsing/Searching

There should be different ways of browsing and searching the repository.

5.6.1 *Faceted Search.* As described in the related work section, multiple authors suggest the use of faceted search in pattern repositories[Mundie et al. 2012; Köppe 2013]. Faceted search makes use of facets (a classifiable characteristic) and headings (attributes of this characteristic). Each selection of a specific heading (or category) a pattern belongs to limits the list of relevant patterns but also the still available facets and headings for the resulting list. This is a powerful mechanism for supporting users searching for patterns based on known characteristics (the categories the patterns are connected with).

5.6.2 *Free-text Search.* Besides using the facet and heading for filtering, one can also make use of a free-text search in the pattern content (maybe limited to specific pattern sections or in combination with facet map-based search).

5.6.3 *Sorting.* The sorting of patterns in the list with search results can be dependent on the user: a regular repository visitor might want to see the newest additions to the repository first while a repository novice might want to see the list with the highest valued or easiest to apply patterns on top of the result list.

5.6.4 *Browse/Search Result.* The result of the browse or search should be presented as a list and include as minimum the name and summary of the patterns and links to the complete pattern description. Additionally, information such as ratings or date of last change can be provided.

5.7 General Access to Repository

Browsing the repository is possible for everyone. Registered users can place comments and examples, rate various aspects of patterns, and make suggestions for improvement.

5.8 Other Requirements

—Software is open source, multiple instances are possible

—there is an import/export functionality, that makes exchanging the repository content (or parts of it) possible between different instances of the repository software, furthermore, via a mapping functionality it also should be possible to import content from other repositories (e.g. defined using the Pattern Language Markup Language[16])

—it should be possible for pattern authors to gain access to patterns that are no longer maintained

5.9 A Conceptual Domain Model for the Repository

In order to get a better overview of the concepts and their relations contained in the previously described requirements, we decided to make use of a conceptual domain model using UML class diagram as notation. Due to the number of included concepts, the model has been split into three parts: Pattern Description (Figure 3), Pattern Writing/Evolution (Figure 4), and Pattern Browsing/Searching (Figure 5).

[16]see `https://www.cs.kent.ac.uk/people/staff/saf/patterns/plml.html` and `https://www.cs.kent.ac.uk/people/staff/saf/patterns/diethelm/plmlx_doc/index.html`

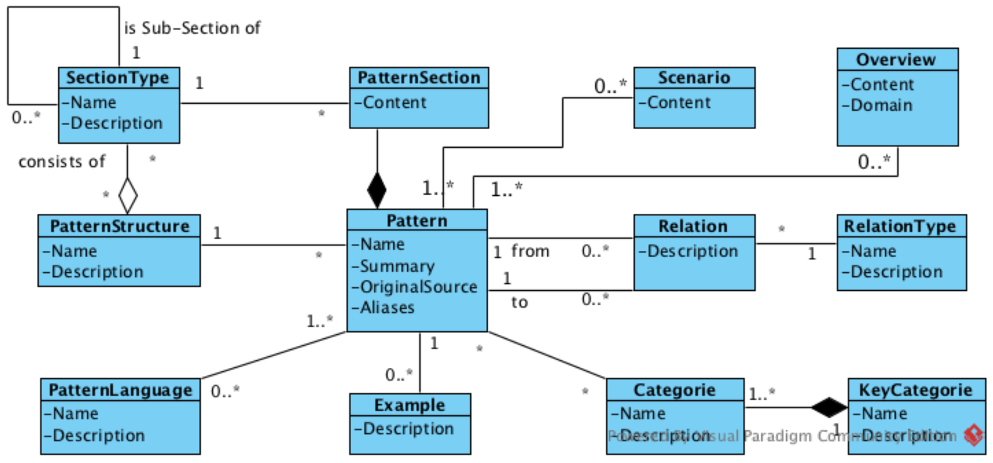


Fig. 3: Domain Model - Pattern Description

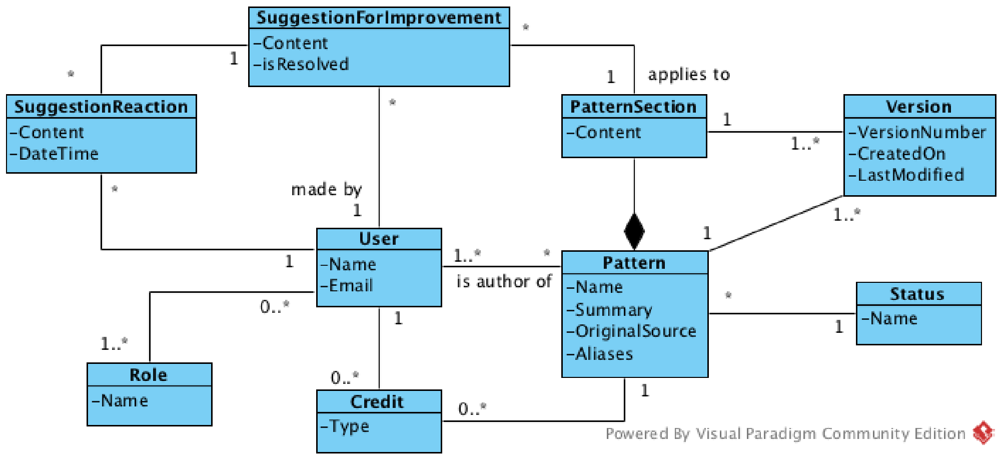


Fig. 4: Domain Model - Pattern Writing/Evolution

The focus of the conceptual domain model lies on the concepts that are part of the domain *pattern repository*, the concepts are therefore independent of the technical realization of the repository. All concepts are also described in more detail below (in alphabetical order).

—**Category** - The specific categories a pattern maps to.

—**Credit** - Indicates which users contributed to a pattern description, e.g. as shepherd, writers' workshop participant, or reviewer.

—**Example** - Examples are an obligatory section for all patterns and can also be used for validation purposes, e.g. a higher number of examples (or known uses) equals a higher validity of the pattern.

—**Key Category** - The groupings of categories used for characterizing various aspects of the patterns (as described earlier).

—**Overview** - An introduction to the most important pattern (and pattern combinations) for a specific domain.

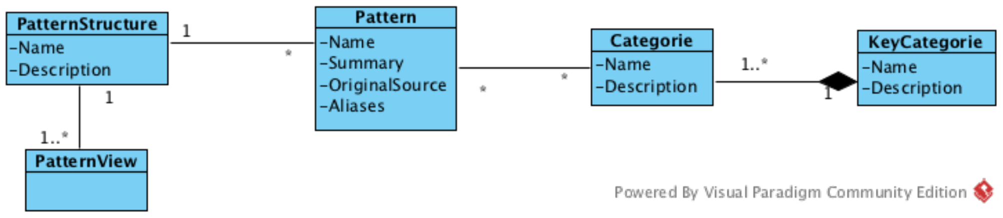


Fig. 5: Domain Model - Pattern Browsing/Searching

—**Pattern** - The core concept for a single pattern. Properties are Name, Summary (also called synopsis in PLML), Original Source (if it has been published elsewhere before), and Aliases. Name and aliases can also be used for identifying if a newly added pattern already exists in the repository.

—**Pattern Language** - A collection of interrelated patterns that are applicable in a specific (sub-)domain.

—**Pattern Section** - The section type content for a concrete pattern.

—**Pattern Structure** - Defines a collection of section types that can function as template for pattern authors.

—**Pattern View** - The definition of how patterns with a specific pattern structure are visually represented. Intentionally left at this high abstraction level, as more fine-grained concepts of pattern view would likely draw already on implementation or technology decisions and are therefore omitted.

—**Rating** - The specific ratings per pattern, version, user, and rating type.

—**Rating Type** - Patterns can be rated using multiple types such as "Easy to Apply", "Maturity" etc. The concrete types still need to be determined.

—**Relation** - Defines the relation between two patterns.

—**Relation Type** - The types relations between patterns can have.

—**Role** - The roles a user can have (determining the accompanying grants) such as author, administrator or “normal" user.

—**Scenario** - A description of a situation where combinations of patterns are used to realize a higher level goal.

—**Section Type** - The specific section of a pattern that forms together with other sections the format of a group of patterns. Examples are *Context*, *Problem*, *Solution*, *Forces*, *Intent*, etc. Section types can be hierarchically ordered so that e.g. sub-sections can be defined such as “Core" and “Details" for a solution section or unique forces in a list-of-forces section.

—**Status** - The status of the pattern with respect to the pattern writing process, such as “Draft", “pattern candidate", or “pattern". The concrete statuses need to be determined and agreed on by the community.

—**Suggestion for Improvement** - Registered users can suggest improvements of patterns. These need to be resolved by the author/s of the patterns.

—**Suggestion Reaction** - The author/s and other registered users can react on suggestions for improvement until these are resolved. These reactions can be used for discussion and/or clarification of the suggestions.

—**Version** - The latest version of a pattern, also related to the updated pattern sections. Includes same attributes as proposed for PLML.

—**User** - A registered user, registration is necessary for becoming an author of patterns and also for providing suggestions for improvement on existing patterns.

6. CONCLUSION

In this paper, we presented the requirements for a online design pattern repository that supports the whole design pattern life cycle and therefore offers a place to:

—write and edit patterns,
—read published patterns,
—collaborate on pattern writing,
—consolidate known patterns,
—share design patterns,
—evaluate design patterns, and
—evolve and refine design patterns.

Open challenges for the design repository community include:

—copyright and ability to share published design patterns,
—design pattern evolution based on the experiences of a community,
—encouraging pattern authors and stakeholders to contribute, and
—given the fragmentation of knowledge in past three decades, connecting distributed patterns and existing repositories.

In future work, we will address the challenge of how to build a community (or multiple communities) that fill the repository with life and make it a valuable place for both people new to a field and experts. We furthermore seek to explore the particular needs of the pedagogical (or educational) pattern community as example domain.

7. ACKNOWLEDGEMENTS

This material is based upon work supported by the National Science Foundation under DRL-1252297. We thank our shepherd Rosana Teresinha Vaccare Braga for her insightful and encouraging feedback.

REFERENCES

ALEXANDER, C., ISHIKAWA, S., AND SILVERSTEIN, M. 1977. *A Pattern Language: Towns, Buildings, Construction.* Oxford University Press.

BECK, K. AND CUNNINGHAM, W. 1987. Using pattern languages for object-oriented programs. Tech. rep., Tektronix Inc.

BERGIN, J., KOHLS, C., KÖPPE, C., MOR, Y., PORTIER, M., SCHÜMMER, T., AND WARBURTON, S. 2015. Assessment-Driven Course Design - Fair Play Patterns. In *Proceedings of the 22nd Pattern Languages of Programs conference, PLoP'15*. Pittsburgh, USA.

BIRUKOU, A. 2010. A survey of existing approaches for pattern search and selection. In *Proceedings of the 15th European Conference on Pattern Languages of Programs - EuroPLoP '10*. ACM Press, Irsee, Germany, 1.

BIRUKOU, A. AND WEISS, M. 2009. Service for selecting patterns. In *Proceedings of the 14th European Conference on Pattern Languages of Programs (EuroPLoP'09)*. 1–12.

CUNNINGHAM, W. AND MEHAFFY, M. W. 2013. Wiki as pattern language. In *Preprints of the 20th Pattern Languages of Programs conference, PLoP'13*. The Hillside Group, Monticello, Illinois, USA, 32.

DENG, J., KEMP, E., AND TODD, E. G. 2005. Managing UI pattern collections. In *Proceedings of the 6th ACM SIGCHI New Zealand chapter's international conference on Computer-human interaction making CHI natural - CHINZ '05*. ACM Press, New York, New York, USA, 31–38.

DERNTL, M. AND BOTTURI, L. 2006. Essential use cases for pedagogical patterns. *Computer Science Education 16,* 2, 137–156.

FEHLING, C., BARZEN, J., FALKENTHAL, M., AND LEYMANN, F. 2015. PatternPedia - Collaborative Pattern Identification and Authoring. In *Proceedings of the International Workshop on Pursuit of Pattern Languages for Societal Change (PURPLSOC)*. epubli GmbH.

FINCHER, S. 2003. Perspectives on HCI patterns: concepts and tools (introducing PLML). *Interfaces 56*, 26–28.

FINLAY, J., GRAY, J., FALCONER, I., HENSMAN, J., MOR, Y., AND STEVEN, W. 2009. Planet: Pattern Language Network for Web 2.0 in Learning.

GAMMA, E., HELM, R., JOHNSON, R., AND VLISSIDES, J. 1994. *Design Patterns: elements of reusable object-oriented software.* Addison-Wesley, Boston, MA.

HARRISON, N. 1999. The Language of Shepherds: A Pattern Language for Shepherding. In *Proceedings of the 6th Annual Conference on the Pattern Languages of Programs (PLoP)*.

HOHPE, G., WIRFS-BROCK, R., YODER, J. W., AND ZIMMERMANN, O. 2013. Twenty Years of Patterns' Impact. *IEEE Software 30,* 6, 88–88.

INVENTADO, P. S. AND SCUPELLI, P. 2015a. Data-driven design pattern production. In *Proceedings of the 20th European Conference on Pattern Languages of Programs - EuroPLoP '15*. EuroPLoP '15. ACM Press, Irsee, Germany, 1–13.

INVENTADO, P. S. AND SCUPELLI, P. 2015b. Towards an open, collaborative repository for online learning system design patterns. *eLearning Papers 42,* June, 1–15.

KÖPPE, C. 2013. Towards a Pattern Language for Lecture Design: An inventory and categorization of existing lecture-relevant patterns. In *Proceedings of the 18th European Conference on Pattern Languages of Programs, EuroPLoP'13*. ACM, Irsee, Germany.

MANNS, M. L. AND RISING, L. 2005. *Fearless Change: Patterns for Introducing New Ideas.* Addison-Wesley.

MUNDIE, D., MOORE, A. P., AND MCINTIRE, D. 2012. Building a Multidimensional Pattern Language for Insider Threats. In *Preprints of the 19th Pattern Languages of Programs conference, PLoP'12*. Tucson, Arizona, USA.

PAVLIČ, L., HERIČKO, M., PODGORELEC, V., AND ROZMAN, I. 2009. Improving Design Pattern Adoption with an Ontology-Based Repository. *Informatica 33,* 2, 181–189.

REINERS, R., FALKENTHAL, M., JUGEL, D., AND ZIMMERMANN, A. 2013. Requirements for a collaborative formulation process of evolutionary patterns. In *Proceedings of the 18th European Conference on Pattern Languages of Program - EuroPLoP '13*. ACM Press, New York, New York, USA, 1–12.

RISING, L. 2000. *The Pattern Almanac.* Addison-Wesley Longman Publishing Co., Inc.

SCHULER AND DOUGLAS. 2008. *Liberating Voices: A Pattern Language for Communication Revolution.* MIT.

SCHÜMMER, T. AND HAAKE, J. M. 2010. PATONGO: Patterns and Tools for Non-Profit OrganizationsâĂŤa pattern-based approach for helping volunteers to identify and share good practice. *New Review of Hypermedia and Multimedia 16,* 1-2, 85–111.

SMITH, G., CZERWINSKI, M., MEYERS, B., ROBBINS, D., ROBERTSON, G., AND TAN, D. S. 2006. FacetMap: A scalable search and browse visualization. *IEEE Transactions on Visualization and Computer Graphics 12,* 5, 797–804.

VANHILST, M., FERNANDEZ, E. B., AND BRAZ, F. 2009. A Multi-dimensional Classification for Users of Security Patterns. *Journal of Research and Practice in Information Technology 41,* 2, 87–97.

WEISS, M. AND BIRUKOU, A. 2007. Building a Pattern Repository: Benefitting from the Open, Lightweight, and Participative Nature of Wikis. In *Workshop on Wikis for Software Engineering at ACM WikiSym, 2007 International Symposium on Wikis (WikiSym), Montre'al, Que'bec, Canada, October 21-23*.

WELICKI, L., LOVELLE, J., AND AGUILAR, L. 2006. Meta-Specification and Cataloging of Software Patterns with Domain Specific Languages and Adaptive Object Models. In *EuroPLoP*. Irsee, Germany.

Appendix

In the following table we present the results of our analysis of the relevant literature and existing pattern repositories.

Source	Repository Content	Pattern Writing	Browsing/Searching	Other Requirements	Comments
Wiki as Pattern Language [Cunningham and Mehaffy 2013]	limited characteristic structure for pattern descriptions, relations by hyperlinks	wiki-based (collaborative, shareable, refinable), crowd-based review	via tags and free-text search	evolutionary (patterns are alive, no "final" state)	suggest federation approach (forking/sharing/merging) and scenario modeling
PatternPedia - Collaborative Pattern Identification and Authoring [Fehling et al. 2015]	extensible pattern metamodel, format definition through metamodel extensions, various formats possible, references between patterns include type, explicitly includes solution documentation (existing pattern implementations)	mainly intended for importing already documented patterns (through a content manager)	full text search and browsing through categories and references, interactive graph of pattern references	import functionality for existing documents (XML)	working on tools for automatically analyzing solutions and extracting patterns
A survey of existing approaches for pattern search and selection [Birukou 2010]	Specify links/relations between patterns within repository or other repositories, pattern description in human-readable and machine-processable format, ability to customize the pattern template	allow users to collaboratively add/edit patterns and relations between them; tag, annotate, comment on, rate patterns; share experience with using patterns	Searching: tags/keywords, full text search, searching using additional data (such as requirements, properties, quality goals); Selection: support through ranked list of results, relevance scores, algorithm/method for selecting patterns; Browsing: display pattern list, view pattern details, navigation via links or relations	Crawling/Indexing for finding links to patterns in other places, Recommendation facilities: suggestions of patterns for problem at hand (or personalized recommendations); specific method supported with tool support, tool is accessible (e.g. via internet), available, and interoperable (possibility to access tool/repository via some API)	Suggestions made: pattern search engine that automatically crawls pattern descriptions on the Internet, support of pattern sequences (ways how patterns are used together in an application), natural integration of approach with the conventional workflow of the user

Continued on next page

Table I – continued from previous page

Source	Repository Content	Pattern Writing	Browsing/Searching	Other Requirements	Comments
PATONGO: Patterns and Tools for Non-Profit OrganizationsâĂŤa pattern-based approach for helping volunteers to identify and share good practice [Schümmer and Haake 2010]	knowledge representation needs to be flexible enough to allow different levels of abstraction: challenges and ideas (title+summary), good practice descriptions, patterns, and encyclopedia overview articles that provide an overview on a specific domain; all of these levels contain pre-defined fields such as title+summary (all), innovative idea+discussion (good practice description) and pattern-related sections (for the pattern descriptions); different text formats represented as structured hypermedia nodes; experience reports of pattern applications; supports various reference types between patterns (uses, is a variant of, specializes, but also address the same keyword, address a related keyword etc.); patterns are positioned in extendible semantic network based on related keywords; additional to content information also usage information is collected	supports working from higher level descriptions (challenges/ideas and good practice descriptions) towards pattern descriptions	search based on keywords in semantic network	focus on patterns as living documents that are improved by peers (evolutionary knowledge process), allowing communication about patterns and experiences with patterns as part of standard interaction; goal is also improvement of patterns over time and extension of patterns with experience reports; also include overview articles and descriptions on successful combinations of patterns (not explicitly named scenarios)	emphasis on organizational learning, interaction between practitioners applying patterns and pattern authors; wiki approaches appropriate, but should be complemented with tools for coordination and communication (within an organization)

Continued on next page

Table I – continued from previous page

Source	Repository Content	Pattern Writing	Browsing/Searching	Other Requirements	Comments
Requirements for a collaborative formulation process of evolutionary patterns [Reiners et al. 2013]	patterns available in any development stage; insertion and adaptation of existing patterns; rule system for definition of pattern maturity; reflecting pattern hierarchies; documentation of anti-patterns; adding semantic meta information to patterns; intorducing key performance indicators for patterns; additional relations that express AND/XOR semantics for library structure	connection between author and ideas/contributions for preserving intellectual properties	Smart visualization methods (for state of pattern, library structure, relations between patterns); Management cockpit	possibility of quick contributions and extensibility; role model for access/contribution organization and maintenance of repository; Involvement of practitioners for determination of pattern validity; decision support using often chosen pattern combinations; transparent process and moderation (reflecting community's activities); process needed for handling actuality of content and outdated patterns	
Improving Design Pattern Adoption with an Ontology-Based Repository [Pavlič et al. 2009]	semantically annotated data using an ontology; hierarchical organization of pattern containers (and patterns); patterns can be in multiple containers; patterns are connected as related, similar (or theSameAs-relation), composed or as hierarchy; includes also real-world examples of pattern usages; patterns can be annotated with additional knowledge	knowledge integration of other sources (via import using RDF)	a question-answer expert system for selecting patterns; indexing of integrated data for full text-search capabilities	Presentation of pattern languages (i.e. interrelated patterns); a questions-answer expert system for selecting patterns; transformations of raw input data (import functionality); service-oriented, so that other services can be built on top	using semantic web technologies

Continued on next page

Table I – continued from previous page

Source	Repository Content	Pattern Writing	Browsing/Searching	Other Requirements	Comments
Planet: Pattern Language Network for Web 2.0 in Learning [Finlay et al. 2009]	templates with predefined sections per status (based on PLML standard): proposal, candidate, and pattern; section with related patterns, different relations possible (extends, is part of, contains, is the same as); includes domain maps (overview) and scenarios	wiki-based; template-based on-line editor; owners of items can grant editorial rights to additional contributors, other users can only comment	tagging framework with key- and sub-categories; search using multiple dimensions (of key-categories); searchable and sortable index per status and for scenarios	explicit evolution of patterns from proposals to candidates to full patterns (with at least 3 application cases); index pages of case studies, patterns, scenarios and domain maps; discussion mechaanism for case studies, patterns, scenarios and domain maps; support of multiple data formats for interoperability (among which PLML, XML, REST, and PDF); (read) access via REST-API	source code of platform is publicly available, unknown if there are running installations
Towards an open, collaborative repository for online learning system design patterns [Inventado and Scupelli 2015b]	hyperlinks between patters, pattern mining sources, application environments; contains links to published literature	wiki-based (collaborative, shareable, refinable)	free-text search; predefined categories	evolutionary	
Service for Selecting Patterns [Birukou and Weiss 2009]			search via user-provided tags	includes service for tracking pattern usage history (for documenting how patterns are used within an organization, collaboration by user linking and perzonalization); recommendation service: patterns for solving specific problems, key patterns in a specific area, pattern sequences for given situation; different roles for users such as admin, repository manager, developer (or user) and writer	

Continued on next page

Table I – continued from previous page

Source	Repository Content	Pattern Writing	Browsing/Searching	Other Requirements	Comments
Perspectives on HCI patterns: concepts and tools (introducing PLML) [Fincher 2003]	Definition of Pattern Language Markup Language (PLML) with these elements: id, name, alias, illustration, problem, context, forces, solution, synopsis, diagram, evidence (examples and rationales), confidence (with star rating), literature, implementation, related patterns (via pattern links, types are is-a, is-contained-by and contains), author, credits, creation-date, last modified, revision-number				Focus is on HCI patterns; the domain model proposed by us allows defining PLML as structure
Building a Pattern Repository: Benefitting from the Open, Lightweight, and Participative Nature of Wikis [Weiss and Birukou 2007]	minimal set proposed that contains: pattlets and their relationships, tags on patterns, organization in pattern collections and pattern languages (via relationships), metadata for pattern descriptions		browsing by collection, pattern name, or tag/s (using a tag cloud).		propose use of wiki as repository platform because of openness, free complex features through plugin architecture, access control, lightweight syntax, and participative architecture
Managing UI Pattern Collections [Deng et al. 2005]	specific and alternative pattern forms; support mulitmedia data; includes implementation code; focus on forces; pattern annotations; versioning; pattern relationships of different types	create new pattern using template; reuse forces; patterns can be modified and customized; import of single patterns and pattern collections	browsing a set of forces; pattern lists; view pattern details; different views; keyword search; full text search; other types of search	implementation code (for code generation); visualizing pattern relationships; collecting sets of patterns for personal use; categorize patterns into named collections; create new collections; XML/PLML support for input/output; alternative output formats	

Continued on next page

Table I – continued from previous page

Source	Repository Content	Pattern Writing	Browsing/Searching	Other Requirements	Comments
Meta-Specification and Cataloging of Software Patterns with Domain Specific Languages and Adaptive Object Models [Welicki et al. 2006]	freely definable pattern format (with a set of properties); source (and event where pattern first was presented); author group; pattern language (with collections of patterns); different pattern types; also relations to (OO-)principles; abstraction level of pattern; targeted role; tags for annotating entities; relationships between patterns (and other entities); summary; also structure and implementation, which are both specific to software design patterns	provides template, based on the sections included in pattern format; support for growing and evolution	multiple views on a pattern (such as complete, summary, CRC, source code, or EML code), depending on interest of user	template definition system that allows combinations of elements of any pattern form; separation of visualization and pattern content; allows discussing about patterns and concepts; ratings	specifies Entity Meta-specification language (EML), where patterns are one kind of enitities
Building a Multidimensional Pattern Language for Insider Threats [Mundie et al. 2012]; Towards a Pattern Language for Lecture Design: An inventory and categorization of existing lecture-relevant patterns [Köppe 2013]			a faceted classification and the facet map approach for better support of browsing and searching for patterns		

Continued on next page

Table I – continued from previous page

Source	Repository Content	Pattern Writing	Browsing/Searching	Other Requirements	Comments
Open Pattern Repository[17]	Specify links/relations between patterns within repository; pattern description in human-readable form; ability to customize the pattern template; focus on design and architectural patterns including a semi-formalization of influence on software quality attributes	Pattern writing online in sections corresponding to a chosen pattern template; support for processing pattern content from PDFs or word files using drag and drop functionality	Searching: tags/keywords, full text search, searching using additional data (such as requirements, properties, quality goals); Selection: support through ranked list of results, relevance scores, algorithm/method for selecting patterns; Browsing: display pattern list, view pattern details, navigation via links or relations	Support for formalizing pattern relationships; building new pattern languages from existing languages	Sourcecode and documentation of pattern repository is publicly available; repository is used by different institutions for internal pattern documentation; a publicly deployed version is currently not available
Integrated Learning Design Environment (ILDE[18])	Different Learning Design Solutions (patterns, scenarios, narratives, course maps etc.) with pre-defined text-templates (thus all are described in RTF-text); url-hyperlinking for references; tagging-mechanism	text-template provided with sections to be filled in, but freely adaptable; additional authors can be added;	browsing per solution type (pattern, scenario etc.); free tags and pre-defined tags in key-categories (descipline and pedagogical approach); free-text search	visibility can be constrained to specific users; specific rankings for completeness and granularity	no mature content present, seems not to be used extensively
Place Patterns[19]	Alexandrian form with only few required sections (title, context, conflict and resolution); scale and tags as categorization; examples of pattern applications (real places); references to other patterns as text only (not linked)	requires registration (not possible anymore)	Browsing via a scale-category and free defined tags; free-text search (in complete documents/pages)		Focus on architecture; seems to be not actively used anymore

Continued on next page

[17] https://github.com/wizzn/openpatternrepository
[18] http://ilde.upf.edu/
[19] http://placepatterns.org

Table I – continued from previous page

Source	Repository Content	Pattern Writing	Browsing/Searching	Other Requirements	Comments
Liberating Voices[20]	fixed pattern format with pre-defined sections (name, number within set, links (to other patterns), problem, context, discussion, solution, verbiage for pattern card, pattern status, tags); references to other patterns via hyperlinks; support of pattern languages (as sets of patterns)	requires registration (access requested)	browsing via tags and links to other patterns or per pattern language (as sorted by numbers in set); free-text search (in complete documents/pages)	translations into other languages; emphasis on knowledge dissemination via pattern cards	focus on positive social change; actively supporting the building of communities
Patterns in Interaction Design[21]	fixed pattern format with pre-defined sections (problem, solution, use when, how, why, more examples, implementation, literature)		index page with all patterns, grouped by categories; free-text search (that does not seem to work well)	comments section per pattern	focus on HCI-patterns
BRIDGE Pattern Library[22]	fixed pattern format with pre-defined sections (context, summary, problem details and forces, solution summary, solution illustration, solution details and consequences, related patterns, origin, author, created on, modified by); different pattern statuses (new submission, under consideration, pattern candidate, approved pattern, revalidation needed)	extended workflow for evolutionary pattern formulation; explicit inclusion of quality assessments and (collaborative) shepherding	simple browsing per hierarchy-level and overview	patterns are continuously monitored and revalidated; discussion section per pattern, also evidence section (supporting and refuting); pattern maturity grows through reviews and collection of evidence; different roles with different responsibilities (visitor, member, author, domain expert, librarian); ratings of patterns (readability, understandability, appropriateness); questions for ratings and evidence of patterns on dashboard; community supported by "Hall of Fame" (for # of submits, comments, votes, evidences)	focused on handling knowledge acquisition and transfer between different (research) project partners and work packages

Table I. : Results of analysis of relevant literature and existing repositories

[20]`http://www.publicsphereproject.org/patterns/`

[21]`http://www.welie.com/patterns/`

[22]`http://bridge-pattern-library.fit.fraunhofer.de/pattern-library/`