OpenEdu Patterns: A Tool to Support Pedagogical Patterns Management

MARIA LYDIA FIORAVANTI, University of São Paulo
GABRIEL SANTOS NICOLAU, University of São Paulo
ARACELE FASSBINDER, Federal Institute of Education, Science and Technology of South of Minas Gerais
ELLEN FRANCINE BARBOSA, University of São Paulo

The effective communication of a knowledge is often a struggle for teachers and instructors. In general, patterns provide a method for capturing and communicating knowledge in several domains, such as pedagogy. Pedagogical patterns try to capture expert knowledge of the practice of teaching and learning. The intent is to capture the essence of the practice in a compact form that can be easily communicated among those interested in the knowledge. Although patterns are useful for storing and recording tacit knowledge, pedagogical patterns and tools that support teaching and learning practice, particularly in Portuguese, are still little explored by the scientific community of Computing applied to Education. Considering the aforementioned scenario, this paper describes *OpenEdu Patterns*, a web tool that allows the creation, management and sharing of pedagogical patterns, both for reuse and for modifications or improvements. The tool can be used by educators, tutors, instructional designers, and so forth, regardless of the context, area or discipline of action.

Categories and Subject Descriptors: K.3.2 [**Computers and Education**]: Computer and Information Science Education—*Computer science education*

General Terms: Pedagogical Knowledge Sharing, Pattern Repository

Additional Key Words and Phrases: Pedagogical Pattern, Educational Pattern, Supporting Tool

ACM Reference Format:

Fioravanti, M. L. et al. 2019. OpenEdu Patterns: A Tool to Support Pedagogical Patterns Management HILLSIDE Proc. of Conf. on Pattern Lang. of Prog. 26 (October 2019), 12 pages.

1. INTRODUCTION

According to [Pressman and Maxim 2014], patterns constitute a mechanism for capturing domain experience and knowledge to allow it to be reapplied when a new problem is encountered. In some cases, domain knowledge is applied to a new problem in the same application domain. In other cases, domain knowledge captured by a pattern can be applied by analogy to a completely different application domain.

[Pressman and Maxim 2014] also stated that a pattern can save from “reinventing the wheel”, or worse, inventing a “new wheel” that is slightly out of round, too small for its intended use, and too narrow for the ground it will roll over.

In general terms, a pattern describes a solution to a problem in a recurring manner. However, the early history of software patterns begins not with a computer scientist but a building architect, Christopher Alexander, who recognized that a recurring set of problems were encountered whenever a building was designed in the late 70’s. He characterized these recurring problems and their solutions as patterns and wrote two books [Alexander et al. 1977; Alexander 1979] to exemplify and describe his method for documenting patterns.

[Alexander et al. 1977] stated that:

> "Each pattern describes a problem that occurs over and over again in our environment and then describes the core of the solution to that problem in such a way that you can use the solution a million times over without ever doing it the same way twice".

Later, [Alexander 1979] affirms that a pattern can be characterized as "a three-part rule which expresses a relation between a certain context, a problem, and a solution".

Considering the scenario of software patterns, [Beck and Cunningham 1987] used the Alexander's concepts in a pattern language to design windows in Smalltalk. After this pioneering work, other initiatives involving software patterns have emerged. [Coad 1992] described seven different analysis patterns. Then, [Coplien 1992] published a book defining several "idioms", which are specific programming patterns to the C ++ language. In 1993, [Gamma et al. 1993] introduced the first of their 23 design patterns, which would be published in 1995 [Gamma et al. 1995].

The book of [Gamma et al. 1995], later called "GoF book", began the dissemination of software patterns to the scientific community. The aforementioned works are the pioneers in patterns, followed by many others in the following years.

When performing the analysis of many patterns, we notice that a pattern solves a problem, but its application can generate other problems, that can be solved through the use of other patterns. In short, no pattern is an isolated entity. Each pattern can exist in the world, only to the extent that is supported by other patterns [Alexander et al. 1977]: the larger patterns in which it is embedded, the patterns of the same size that surround it, and the smaller patterns which are embedded in it.

In a software development perspective, a *pattern language* is a structured collection of patterns that rely on each other to transform requirements and constraints into an architecture [Coplien 1998]. A pattern language is a way of subdividing a general problem and its complex solution into a number of related problems and their respective solutions. Each pattern language solves a specific problem in the common context shared by the language. It is important to note that each pattern can be used separately or with a number of patterns of the language. This means that a pattern alone is considered useful even if the language is not being used in its fullness.

[Alexander et al. 1977] also stated:

> "A collection of patterns forms a vocabulary for understanding and communicating ideas. Such a collection may be skillfully woven together into a cohesive "whole" that reveals the inherent structures and relationships of its constituent parts toward fulfilling a shared objective."

In a different, but related perspective, the effective communication of a knowledge is often a struggle for teachers and instructors. According to [Bergin et al. 2012], they may try several teaching strategies, but this trial and error process can be time-consuming and fraught with error. Usually, advice is often sought from other expert instructors, but these individuals are not always readily available. This creates the need to find other ways to facilitate the sharing of teaching techniques between expert and novice teachers.

Thus, patterns have received much attention in the educational and learning design domains. Despite the use of different terms to define patterns in this context, such as pedagogical patterns [Laurillard 2013], learning patterns [Iba and Sakamoto 2011] as well as instructional design patterns [Goodyear 2005], in our work we refer to them as pedagogical patterns.

In this sense, [Bergin 2002] defined:

> "Pedagogical patterns try to capture expert knowledge of the practice of teaching and learning. The intent is to capture the essence of the practice in a compact form that can be easily communicated to those who need the knowledge. Presenting this information in a coherent and accessible form can mean the difference between every new instructor needing to relearn what is known by senior faculty and easy transference of knowledge of teaching within the community".

In turn, [Goodyear 2005] defined them as a clear articulation of a design problem and a design solution, and offering a rationale which bridges between pedagogical philosophy, research based evidence, and experiential knowledge of design. Furthermore, they can be related to each other and thus offer a toolkit of interrelated design solutions that can be applied to novel problems.

In this context, although patterns are useful for storing and recording tacit knowledge, pedagogical patterns and tools that support teaching and learning practice, particularly in Portuguese, are still little explored by the scientific community of Computing applied to Education.

Considering the aforementioned scenario, this paper describes *OpenEdu Patterns*[1], a web tool that allows the creation, management and sharing of patterns and pedagogical pattern languages as well as the search for educational practices described as patterns. Teachers, tutors, and instructional designers can use the tool to formalize and store knowledge related to experience in teaching practices. They can also use it to find practices shared by other colleagues, which can be applied in their respective contexts to enhance teaching and learning.

The remainder of this paper is organized as follows. In Section 2, we present related work. In Section 3, the tool *OpenEdu Patterns* is presented. Finally, our conclusions and perspectives for future work are drawn in Section 5.

2. RELATED WORK

To the best of our knowledge, few tools have been released to support the creation, management and sharing of patterns, in general. Some initiatives can be found in particular domains, such as software architecture, user interface and education.

In the software architecture domain, for instance, *Open Pattern Repository*[2] was an initiative whose goal was to create a publicly available and freely usable online repository for patterns with a focus on architectural and design software patterns.

In the context of user interface, we can mention *UI-Patterns*[3], which is an attempt to over time create a library of solutions to common design problems, but also rationalizing about how, when, and why such solutions should be used.

In the educational perspective, Derntl [Derntl 2004] proposed the *Person-Centered e-Learning (PCeL) Patterns Repository*, which was basically a pedagogically enriched approach to blended learning. Also, we can mention the work of Botturi and Belfer [Botturi and Belfer 2003], who used *E(2)ML*, an Educational Environment Modeling Language, aiming to implement a pattern system.

In a context more related to this work, we can highlight *Pedagogical Pattern Collector*[4], a project developed by English researchers that allows educators to explore, create or adapt sets of pedagogical patterns (lesson plans, learning strategies, experiments, among others).

We can also mention *E-LEN*[5], an E-learning Design Patterns Repository. The E-LEN project involved a network of European institutions with expertise in e-learning. The project aimed at the development and dissemination of design knowledge tailored to the needs of people who were professionally involved in e-learning, such as the staff of e-learning centres.

More recently, an initiative entitled *OPTion (Open Pattern Tool for Higher Education Research and Practice)*[6] emerged aiming at aims at the systematic documentation, discussion and sustainable dissemination of practices in higher education.

[Köppe et al. 2016] recently discussed the reason why several pattern repositories, such as *Portland Pattern Repository*, *PATONGO*, *PatternPedia*, or the *Open Pattern Repository*, which emerged in the last years seemed

[1] `http://caed.icmc.usp.br/openedupatterns/`
[2] `http://www.cs.rug.nl/search/ArchPatn/OpenPatternRepository`
[3] `http://ui-patterns.com/`
[4] `http://www.ld-grid.org/resources/tools/pedagogical-pattern-collector`
[5] `http://www2.tisip.no/E-LEN/`
[6] `https://www.patternpool.de/`

to disappear. The authors discussed the reason why this may have happened and they mentioned [Köppe et al. 2016]: (i) an experienced lack or inappropriateness of functionality required for supporting all activities in the pattern lifecycle and the shepherding process; and (ii) the lack of a community of stakeholders that actively evolves the repository and keeps it alive through adding or updating content.

Despite their relevance, the initiatives available still lacked some functionalities. We can mention that not all the tools were open to the community of users to create and reuse the existing patterns; the collaboration among users is limited, and the one we consider the most important which is availability of the tools in multiple languages, particularly Portuguese. *OpenEdu Patterns* is a step forward in this direction aiming to bridge such gaps.

3. OVERVIEW OF *OPENEDU PATTERNS*

3.1 Prototype

The work of Fioravanti [Fioravanti and Barbosa 2017; Fioravanti 2017] investigated how pedagogical patterns and pattern languages could assist in the requirements elicitation for mobile learning apps, resulting in a pattern language entitled MLearning-PL. Similarly, Fassbinder [Fassbinder et al. 2017; Fassbinder 2018] investigated the use of patterns to support the design of MOOC courses. In this context, there was a need to make the resulting pattern languages available so that they could be easily used by interested parties. However, we did not find a repository or tool encompassing the features mentioned in section 2 and that has not been discontinued. Another relevant factor to be highlighted is the language, because in general the tools are in English and it was important that we had an option that was also in Portuguese, since some users prefer to use tools in their mother tongue.

Aiming to bridge this gap, we proposed *OpenEdu Patterns*. In general terms, *OpenEdu Patterns* is a web platform that allows users to register, visualize and customize patterns of teaching practices and experiences. It is a pattern repository specifically designed for the educational area, aiming to catalog patterns and allowing educators to share their practices, experiences and ideas regarding certain content, to be reused by third parties. Patterns can be categorized using tags, searched by keywords or authors. Patterns may or may not be related, and one pattern may be a prerequisite or complement to another, for example.

OpenEdu Patterns was created using the Minimum Viable Product (MVP) [Duc and Abrahamsson 2016] concept in order to release small versions of the software already validated in a short period of time for use by the target audience [Silva et al. 2017]. The target audience of the tool are educators, tutors, instructional designers, researchers and so forth, regardless of the context, area or discipline of action. After several validations on its prototype version, we considered the improvement points to start developing the tool.

3.2 Design

While in a traditional process each iteration is not necessarily focused on adding a significant new set of functionalities, an agile software project seeks the ability to deploy a new version of the software at the end of each iteration, a stage at which the team in charge re-evaluates the project priorities.

In this sense, we used an adapted version of framework Scrum [Schwaber 1997; Schwaber and Beedle 2002] to guide the development of *OpenEdu Patterns*. Scrum is based on the theory of empirical process control, that is, it employs an iterative and incremental approach to optimize predictability and control risks.

One of the key Scrum artifacts we used was the Product Backlog, which is a prioritized list of everything needed to build or increment the product. As this list may include new features and bug fixes, we have added all of the items gathered from *OpenEdu Patterns* prototype, and the feedback of the conducted validations.

Table I shows an excerpt of our Product Backlog with some examples of features gathered from *OpenEdu Patterns* prototype, and the feedback of the conducted validations.

The stage of prototyping and validating with end users was of extreme importance for this phase of development. The feedback received allowed us to better evaluate which features and enhancements would be included in the Product Backlog.

Table I. Product Backlog Excerpt

ID	Feature	Story Description
1	Login	As a user, I would like to login to the platform so that I can access my patterns and languages, as well as rate and comment on other patterns and languages.
2	User registration	As a user, I would like to create an account to share my pedagogical patterns using my email and an 8 to 16-character password.
3	Pattern Management	As a user, I wish I could create new patterns and put them on the platform so my colleagues could see. It would be great to be able to edit or delete patterns.
4	Language Management	As a user, I want to be able to create new pattern languages to put them in the tool, as well as edit or delete them. It would also be interesting to be able to list all the pattern languages created by me.
5	Search pattern/language	As a user, I want to be able to search for patterns and pattern languages related to certain subjects or keywords.
...	...	...

3.3 The *OpenEdu Patterns* Platform

OpenEdu Patterns was developed using an MVC architectural pattern [Buschmann et al. 2013]. As shown in Figure 1, the *Model* consists of tables implemented in MySQL to store *User*, *Pattern* and *Language* information as well as the relationship among tables. The *View* was implemented using Embedded JavaScript to generate HTML markup, along with the Bootstrap framework for GUI with JavaScript scripts to handle page interactions. Lastly, the *Controller*, implemented in NodeJS, handles the interaction between the *Model* and the *View*. It gets the information sent by the user and processes it using the MySQL tables, sending the results to the *View*.

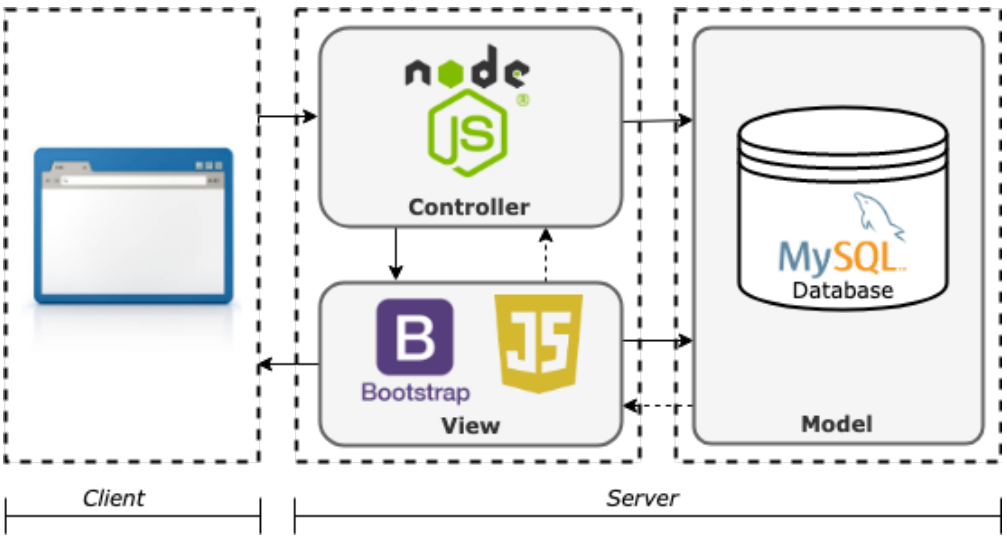

Fig. 1. *OpenEdu Patterns* architecture

In its current version, *OpenEdu Patterns* has the following main features:

User Registration:. the user can create an account on *OpenEdu Patterns* using an e-mail.

Login:. Using the e-mail and password defined in the registration process, the user is able to log on *OpenEdu Patterns*.

Languages Management:. This feature allows a user to create a new pattern language, edit or delete his/her existing ones.

Patterns Management:. Similarly to the previous feature, a user is allowed to create a new pattern, edit or delete his/her existing ones.

Search by Pattern/Language:. Using this feature, a user is able to search patterns and pattern languages that are stored in *OpenEdu Patterns*.

It is important to mention that the user must be logged in to create, edit and delete patterns or pattern languages, but all patterns and pattern languages set as public are available to be searched and reused.

Figure 2, for instance, shows *OpenEdu Patterns* home page, where the user can either log in the system or create an account. In the right top of the screen, the search feature is available.

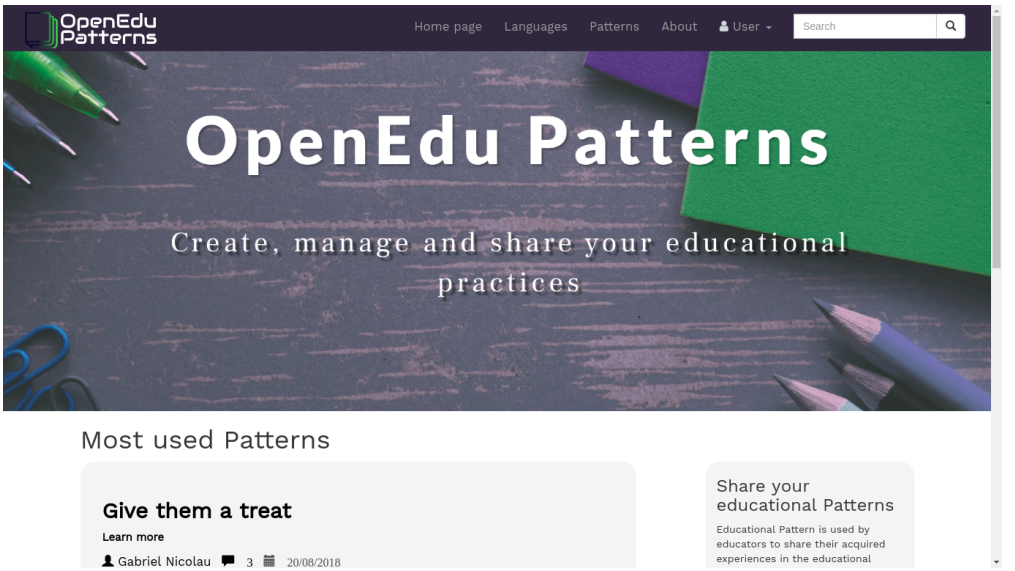

Fig. 2. *OpenEdu Patterns*– Home page

With the use of *OpenEdu Patterns*, it is possible to contribute to the storage and retrieval of patterns and pedagogical pattern languages. It is also possible to disseminate the knowledge of pattern languages in the educational area and encourage their use by educators of different levels and modality of teaching; and contribute to the improvement of teaching methods through the availability of pedagogical practices.

To better illustrate how *OpenEdu Patterns* works, its usage flow will be shown next. Suppose an educator wants to store a pattern language *ABC*, containing three patterns: *A*, *B*, and *C*. It is worth mentioning that to insert patterns in the context of a pattern language and relate them, they must have already been created in the tool.

Therefore, the following steps (Figure 3, Figure 4 and Figure 5) should be performed.

Step 1. Access *OpenEdu Patterns* and click Login

Step 2. Fill in the details to log in

Step 3. Enter the *Patterns* section and click the *Create New Pattern* button

Step 4. Choose a template that has *Related Patterns* and click *Choose Template*

Step 5. Fill in the pattern data, such as *Pattern A*. Scroll to the bottom of the page and click *Create Pattern*

Step 6. Repeat steps 2 through 5 to create *Pattern B* and *Pattern C*

Step 7. Once the required patterns are created, enter the *Languages* section and click *Create New Language*

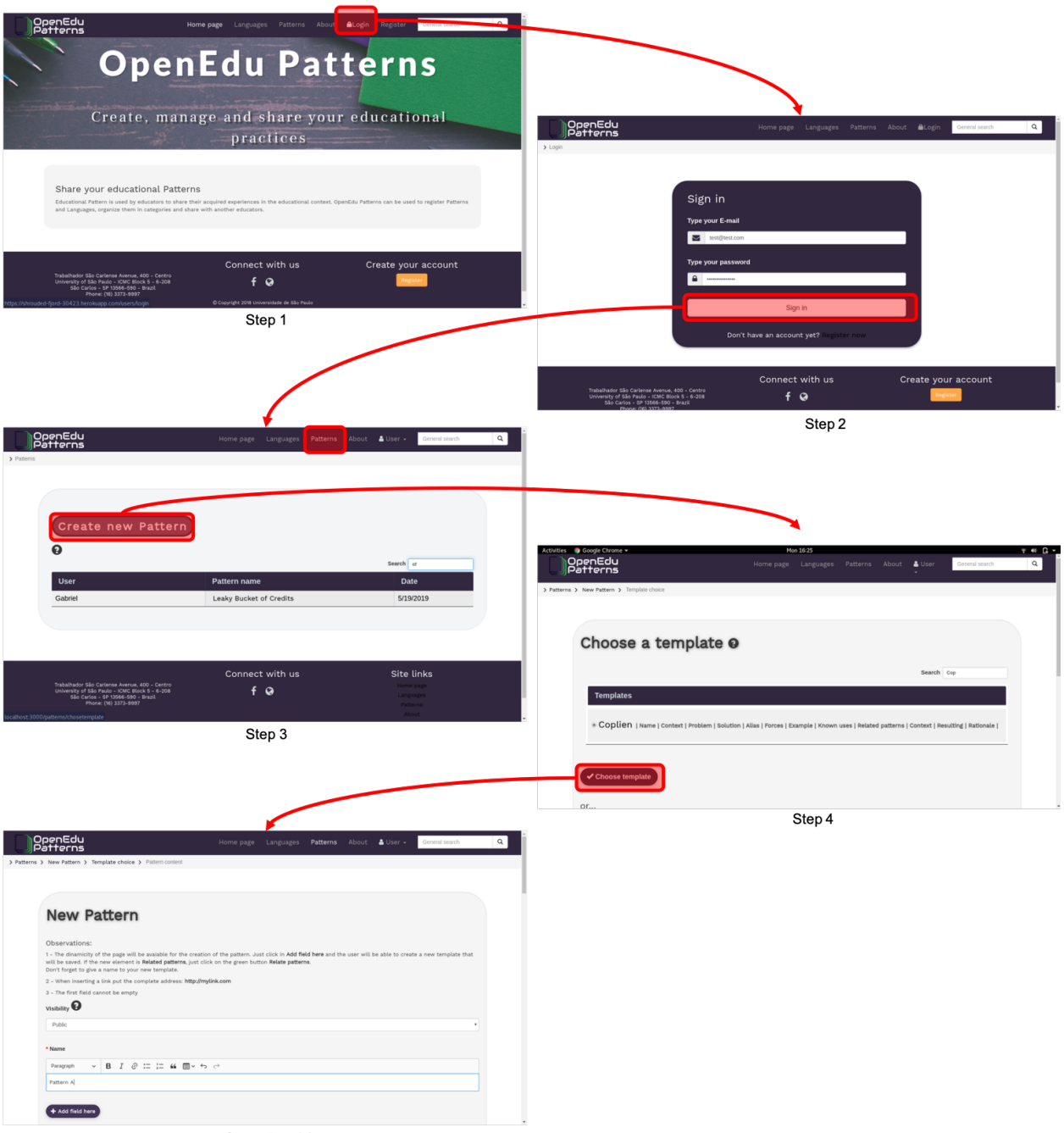

Fig. 3. Steps 1 through 6

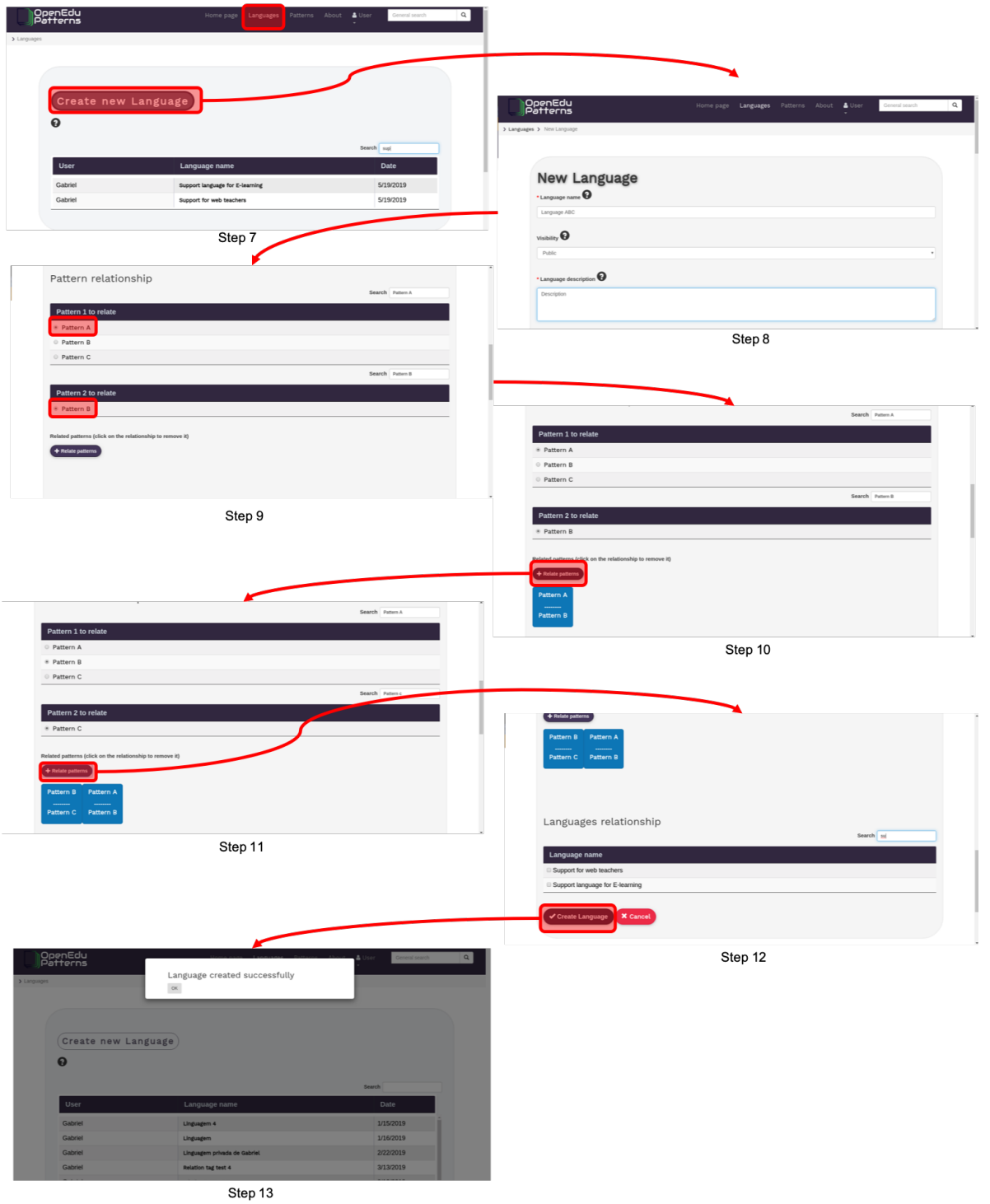

Fig. 4. Steps 7 through 13

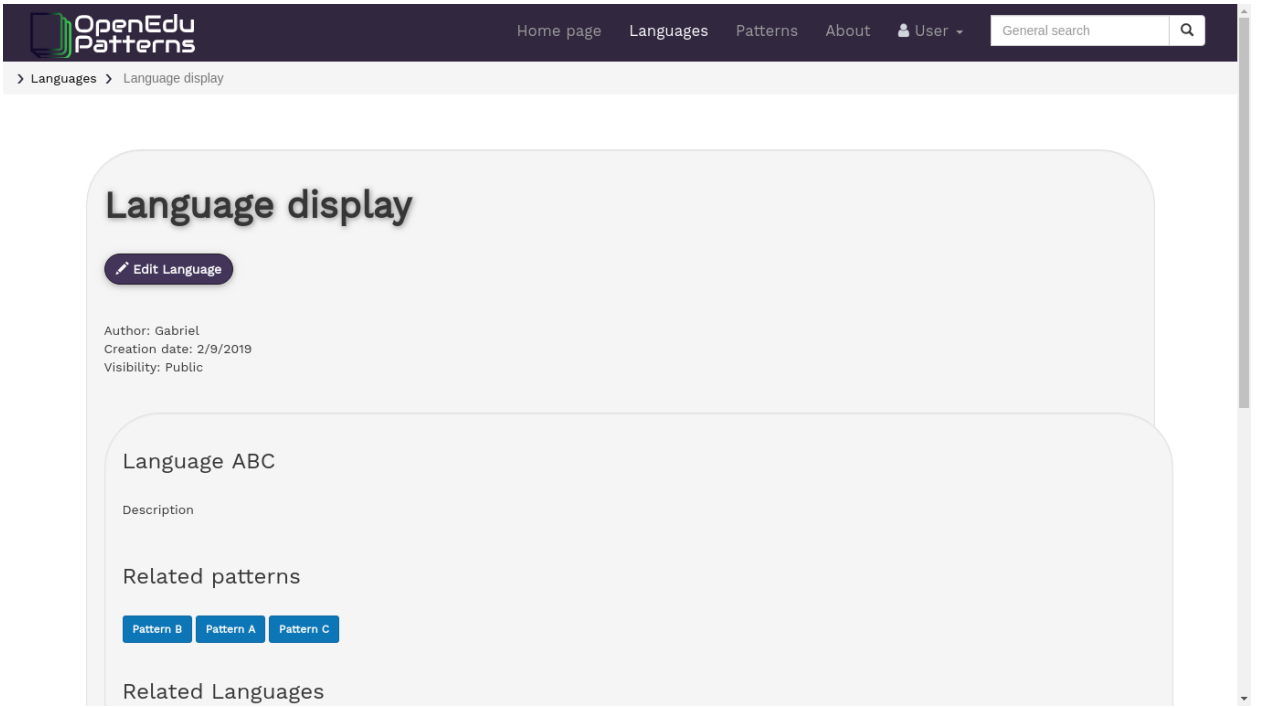

Fig. 5. Step 14

Step 8. Fill in the pattern language name and its description.

Step 9. To relate two patterns within one pattern language, there will be a section entitled *Pattern Relationship*. It is divided into 2 parts: *Pattern 1 to relate* and *Pattern 2 to relate*. To relate, for example, *Pattern A* to *Pattern B*, the user must select *Pattern A* at the top and *Pattern B* at the bottom

Step 10. After selecting the two required patterns, simply click *Relate Patterns*. A block with the relationship should appear below the button. To remove any relationship between patterns, just click on the blue block of the relationship.

Step 11. Repeat Steps 9 and 19 to establish the relationship between *Pattern B* and *Pattern C*

Step 12. Scroll the page to its end and click *Create Language*

Step 13. The user will be redirected to the *Languages* section. To view the created pattern language, simply select it from the displayed table.

Step 14. Visualization of the created pattern language

4. EVALUATION

Usability is most often defined as the ease of use and acceptability of a system for a particular class of users carrying out specific tasks in a specific environment. Ease of use affects the users' performance and their satisfaction, while acceptability affects whether the product is used [Bevan 1995]. To ensure *OpenEdu Patterns* has these essential usability characteristics, we chose to carry out the evaluation using inspection methods.

Heuristic evaluation (HE) is the most common informal method. It involves having usability specialists judge whether each dialogue or other interactive element follows established usability principles [Nielsen 1994b].

The original and adopted approach is for each individual evaluator to inspect the interface alone. Only after all the evaluations have been completed are the evaluators allowed to communicate and aggregate their findings. This restriction is important in order to ensure independent and unbiased evaluations. During a single evaluation session, the evaluator goes through the interface several times, inspects the various interactive elements, and compares them with a list of recognized usability principles (in this case, Nielsen's Usability Heuristics [Nielsen 1994a]).

Our heuristic evaluation was performed by three usability specialists who followed pre-defined instructions and filled in a table with the following information:

—ID: Sequential numbering that identifies the problem pointed out by the expert.

—Heuristic: Represents the numbering of each of Nielsen's heuristics.

(1) Visibility of system status
(2) Match between system and the real world
(3) User control and freedom
(4) Consistency and standards
(5) Error prevention
(6) Recognition rather than recall
(7) Flexibility and efficiency of use
(8) Aesthetic and minimalist design
(9) Help users recognize, diagnose, and recover from errors
(10) Help and documentation

—Description of the problem: Description presented by the expert for the problem found.

—Task: Represents the tasks previously presented.

—Screen: Name that best represents the system screen where the problem was identified.

—Degree of severity:

—0 = I don't agree that this is a usability problem at all
—1 = Cosmetic problem only: need not be fixed unless extra time is available on project
—2 = Minor usability problem: fixing this should be given low priority
—3 = Major usability problem: important to fix, so should be given high priority
—4 = Usability catastrophe: imperative to fix this before product can be released

After they completed the individual evaluations and then their findings were aggregated, 82 issues were identified.

We grouped the average and maximum severities by heuristic (Figure 6) to analyze the strengths and weaknesses identified. As we can see, heuristic #10 regarding *help and documentation* was not a raised concern. On the other hand, heuristic #5 (regarding *error prevention*) was violated with maximum severity and five other heuristics (#2, #3, #4, #6 and #9) had major usability problems.

Analyzing the comments of the evaluators, we agreed that improvements should be made in order to evolve *OpenEdu Patterns* and most of them are already done. After conducting more evaluations other improvements will be made.

5. CONCLUSIONS AND FUTURE WORK

This paper described a web tool entitled *OpenEdu Patterns*, which is in its final stage of development. The target audience is educators and other stakeholders interested in the management and sharing of pedagogical patterns.

The expected contribution is twofold: (i) provide means for storing and disseminating both the patterns already existing in the literature and also new pedagogical patterns and pattern languages; (ii) stimulate the creation of

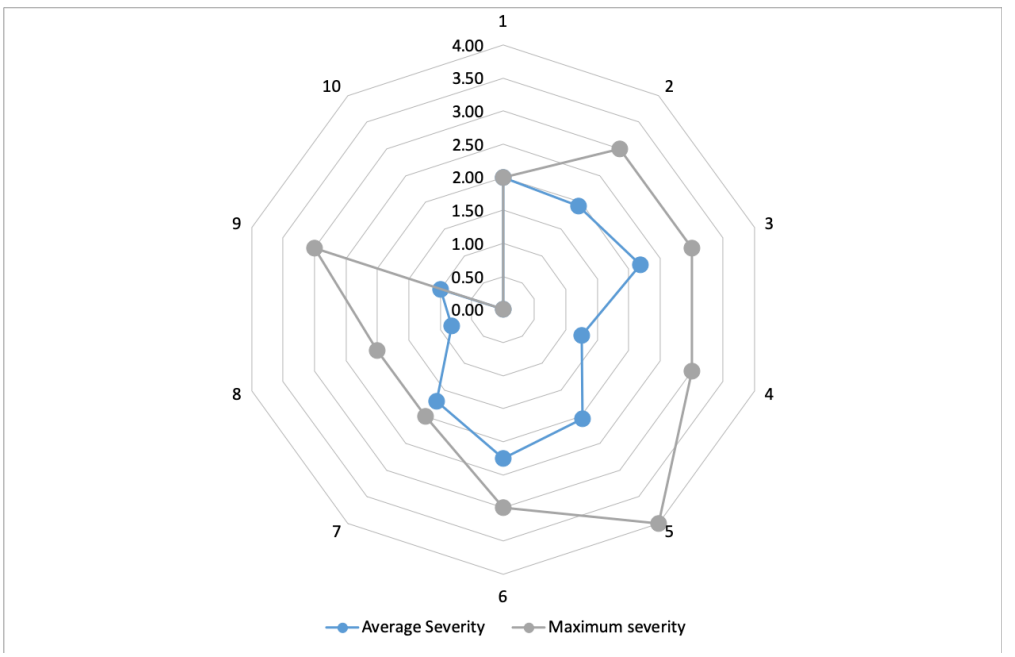

Fig. 6. Nielsen's Heuristics vs. Severities

innovative teaching practices and their subsequent formalization through writing patterns and their relationships and groupings.

As future work, we aim at conducting different types of evaluations to raise improvement points that should be addressed. Also, we intend to release a mobile application of *OpenEdu Patterns*.

Acknowledgments

The authors would like to thank the Brazilian funding agencies - FAPESP, CAPES and CNPq. Our greatest thank and gratitude goes to our shepherd, Yannis Dimitriadis, for his valuable suggestions and comments to improve this paper.

REFERENCES

Christopher Alexander. 1979. *The Timeless Way of Building*. Oxford University Press, New York.

Christopher Alexander, Sara Ishikawa, and Murray Silverstein. 1977. *A Pattern Language: Towns, Buildings, Construction*. Oxford University Press, New York.

Kent Beck and Ward Cunningham. 1987. Using Pattern Languages for Object Oriented Programs. In *Conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA)*. `http://c2.com/doc/oopsla87.html`

Joseph Bergin. 2002. Some Pedagogical Patterns. (May 2002). `http://csis.pace.edu/~bergin/patterns/fewpedpats.html`

Joseph Bergin, Jutta Eckstein, Markus Volter, Marianna Sipos, Eugene Wallingford, Klaus Marquardt, Jane Chandler, Helen Sharp, and Mary Lynn Manns. 2012. *Pedagogical Patterns: Advice For Educators*. Joseph Bergin Software Tools.

Nigel Bevan. 1995. Measuring usability as quality of use. *Software Quality Journal* 4, 2 (1995), 115–130.

Luca Botturi and Karen Belfer. 2003. Pedagogical Patterns for Online Learning. In *Proceedings of E-Learn: World Conference on E-Learning in Corporate, Government, Healthcare, and Higher Education 2003*, Allison Rossett (Ed.). Association for the Advancement of Computing in Education (AACE), Phoenix, Arizona, USA, 881–884. `https://www.learntechlib.org/p/15083`

Frank Buschmann, Regine Meunier, Hans Rohnert, Peter Sommerlad, and Michael Stal. 2013. *Pattern-Oriented Software Architecture, A System of Patterns*. Vol. 1. John Wiley & Sons.

Peter Coad. 1992. Object-oriented Patterns. *Commun. ACM* 35, 9 (Sept. 1992), 152–159. DOI:http://dx.doi.org/10.1145/130994.131006

James O. Coplien. 1992. *Advanced C++ Programming Styles and Idioms*. Addison-Wesley.

James O. Coplien. 1998. The Patterns Handbooks. Cambridge University Press, New York, NY, USA, Chapter Software Design Patterns: Common Questions and Answers, 311–319.

Michael Derntl. 2004. The Person-Centered e-Learning pattern repository: Design for reuse and extensibility. In *EdMedia+ Innovate Learning*. Association for the Advancement of Computing in Education (AACE), 3856–3861.

Anh Nguyen Duc and Pekka Abrahamsson. 2016. Minimum viable product or multiple facet product? The Role of MVP in software startups. In *International Conference on Agile Software Development*. Springer, 118–130.

Aracele G. O. Fassbinder. 2018. *A Contribution to the Process of Designing for Learning in Massive Open Online Courses*. Ph.D. Dissertation. Institute of Mathematics and Computer Science – ICMC/USP, São Carlos, SP.

Aracele G. O. Fassbinder, Ellen Francine Barbosa, and George Magoulas. 2017. Towards and Educational Design Pattern Language for Massive Open Online Courses (MOOCs). In *Proceedings of 24th Conference on Pattern Languages of Programs (PLoP 2017)*. Vancouver, BC, Canada.

Maria Lydia Fioravanti. 2017. *MLearning-PL: a pedagogical pattern language for mobile learning applications*. Master's thesis. Institute of Mathematics and Computer Science – ICMC/USP, São Carlos, SP.

Maria Lydia Fioravanti and Ellen Francine Barbosa. 2017. A Pedagogical Pattern Language for Mobile Learning Applications. In *Proceedings of 24th Conference on Pattern Languages of Programs (PLoP 2017)*. Vancouver, BC, Canada.

Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. 1995. *Design Patterns: Elements of Reusable Object-oriented Software*. Addison-Wesley, Boston, MA, USA.

Erich Gamma, Richard Helm, Ralph E. Johnson, and John M. Vlissides. 1993. Design Patterns: Abstraction and Reuse of Object-Oriented Design. In *Proceedings of the 7th European Conference on Object-Oriented Programming (ECOOP '93)*. Springer-Verlag, London, UK, UK, 406–431.

Peter Goodyear. 2005. Educational design and networked learning: Patterns, pattern languages and design practice. *Australasian journal of educational technology* 21, 1 (2005).

Takashi Iba and Mami Sakamoto. 2011. Learning patterns III: a pattern language for creative learning. In *Proceedings of the 18th Conference on Pattern Languages of Programs*. ACM, 29.

Christian Köppe, Paul Salvador Inventado, Peter Scupelli, and Uwe Van Heesch. 2016. Towards Extending Online Pattern Repositories: Supporting the Design Pattern Lifecycle. In *Proceedings of the 23rd Conference on Pattern Languages of Programs (PLoP '16)*. The Hillside Group, USA, Article 15, 26 pages. http://dl.acm.org/citation.cfm?id=3158161.3158180

Diana Laurillard. 2013. *Teaching as a design science: Building pedagogical patterns for learning and technology*. Routledge.

Jakob Nielsen. 1994a. *Usability engineering*. Elsevier.

Jakob Nielsen. 1994b. Usability inspection methods. In *Conference companion on Human factors in computing systems*. ACM, 413–414.

Roger S. Pressman and Bruce R. Maxim. 2014. *Software Engineering: A Practitioner's Approach*. McGraw-Hill Education.

Ken Schwaber. 1997. Scrum development process. In *Business object design and implementation*. Springer, 117–134.

Ken Schwaber and Mike Beedle. 2002. *Agile software development with Scrum*. Vol. 1. Prentice Hall Upper Saddle River.

Josias Marques Silva, Ellen Francine Barbosa, Maria Lydia Fioravanti, and Aracele G. O. Fassbinder. 2017. Uma Ferramenta de Apoio ao Gerenciamento de Padrões para Propósitos Pedagógicos (in Portuguese). In *Anais dos Workshops do VI Congresso Brasileiro de Informática na Educação (WCBIE 2017)*. Recife, Pernambuco, Brasil.

Received May 2019; revised September 2019; accepted Month YYYY