TOWARDS A COMMUNITY FOR INFORMATION SYSTEM DESIGN VALIDATION

S. Dupuy-Chessa, D. Rieu and N. Mandran
LIG Laboratory, CNRS, University of Grenoble, 681 rue de la Passerelle, BP 72, 38402 St Martin d'Hères, France

Keywords: Design, Validation, Evaluation, Patterns, Community.

Abstract: Information systems become ubiquitous. This opens a large spectrum of possibilities for end-users, but the design complexity is increasing. So domain specific languages are proposed sometimes supported by appropriate processes. These proposals are interesting but they are under-validated. Even if validation is a difficult task, which requires specific knowledge, we argue that validation should be systematic. But many problems remain to be considered to achieve this goal: 1) computer scientists are often not trained to evaluation; 2) the domain of information systems design validation and evaluation is still under construction. To cope with the first problem, we propose to capitalize evaluation practices into patterns so that they can be reusable for non-specialists of validation practices. For the second issue, we propose an environment where evaluation specialists and engineering methods specialists can work together to define their common and reusable patterns.

1 INTRODUCTION

Nowadays, technological progresses such as microprocessors and sensors miniaturization, and communication technologies explosion, allow end-users to access information everywhere at any time and in a personalized manner. In other words, information becomes instantaneous, universal and ubiquitous. This opens a large spectrum of possibilities for the end-users: they can see their bus timetable on their mobile phone; they can see contextualized information on special devices while visiting a museum; etc. For instance, simply considering the new human-computer interaction possibilities triggers business evolution (Godet-Bar, 2008). Therefore design complexity is increased by adding parameters like devices, location, user's characteristics... As mentioned by L. Palen (Palen, 2002), the level to design such systems has been moved up: many different people (designers, stakeholders, sponsors end-users) are involved in the design; many new domain specific languages are proposed to represent the ubiquitous aspects; personalised processes must be defined.

However if many proposals for languages or for processes exist, they are focused too often on the conceptual contributions while their validation, even empirical, is not addressed. For instance, in the software engineering domain, validation is absent of approximately 30%-50% of the papers that require validation (Tichy, 1997) (Zelkowitz, 1997). Considering that this issue is increased by the expansion of new languages and processes for ubiquitous information systems, we think that it is time to cope with the problem of validation in information systems design methods.

In other domains of computer science such as human-computer interaction or empirical software engineering, evaluations are required for any contribution. In information systems, some works have defined conceptual frameworks for defining the quality of languages (Lindland, 1994) (Krogstie, 1998) (Moody, 2003). If these proposals are interesting to understand the characteristics of a language or to provide a framework for evaluation, they do not help non-specialists with practical considerations. Only few works give practical guidelines: for instance, (Aranda, 2007) (Patig, 2008) define generic experimental protocols for models understandability. But evaluation remains still very much an «art» than a «science» (Nelson, 2005).

To address this issue, we think that evaluation specialists and engineering methods specialists must collaborate in order to share and consolidate their practices. They must create a community for evaluation where they can exchange their knowledge. This knowledge must be also reusable even for non-specialists in evaluation so that validation can become systematic. So we propose to formalise evaluation knowledge with design patterns, which are well-known by information system specialists. We are creating an environment for the diffusion of these patterns, but also for their elaboration in a collaborative way.

In the next section, we present our vision of patterns for validation. The third section details the concept of collaborative design patterns and a tool that we are currently developing for their creation. Finally, we conclude this paper with some perspectives.

2 PATTERNS FOR VALIDATION

2.1 Methods Validation Practices

An information systems design method consists of modelling languages, a development process model and tools to manage models or to support process. The quality of all these aspects has been studied in some extends in the literature (Dupuy-Chessa, 2009).

Many different ways have been proposed to evaluate models (Lindland, 1994) (Lange, 2005) (Si-Said, 2007) or languages (Krogstie, 2003) (Aranda, 2007). Only a few of them have addressed the problem of process validation. Anyway we can note the existence of different approaches, which can be combined in order to offer a global vision of evaluation practices.

The main approach to evaluate a language is to realize empirical evaluations with user experiments. These evaluations are complex because they must not assess the quality of particular instances of the languages (e.g. a particular class diagram), but the quality of the language in general (the class diagram). (Siau, 2001) uses an approach based on the human-information processing model Goals Operators Methods and Selection Rules (GOMS) to evaluate UML diagrams. The authors measure the execution time to realize some UML diagrams so as to determine their complexity.

Another approach to measure a language complexity is to study its meta-model. A complex meta-model should lead to a greater expressive power and thus to smaller models (Mohagheghi, 2007). (Rossi, 1996) explains that there exists an intrinsic dependency between the meta-model and the learnability of a language: a modelling language is composed of a set of diagrams for which some metrics are calculated. The conceptual complexity of a diagram is a sum vector of the above diagram metrics while the complexity of the whole language is a sum vector of the complexity of its diagrams. In such a view, the relations between diagrams are not considered. However the approach has permitted to compare several object-oriented languages and to conclude that object-oriented languages become more complex with time.

Based on their generic quality framework, which defines the various views of language quality, Krogstie et al. have also evaluated UML in its 1.4 version. They concluded that UML is difficult to comprehend because there are many fundamentally different concepts, which are not always formally defined.

Finally there is a reverse inference approach (Moody, 2005) where researchers work backwards from the quality characteristics of the final system to the characteristics of the model. We use this approach to evaluate a new component model, called Symphony Objects model, because we hypothesised the existence of a causal relationship between the characteristics of our conceptual model (the Symphony Objects model) and the characteristics of the code. Then we try to evaluate the Symphony Objects model through the quality of several of its implementations (Ceret, 2010).

All these proposals are valuable for language validation and could be reused and combined to ameliorate information systems design validation. But they often need to be generalized and described as practical guidelines.

For processes, their quality is generally measured by evaluating the process model. For instance, (Mendling, 2007) studies the characteristics that make a process model understandable. Many other works have been realized in the domain of business process models. But it is also possible to realize empirical experiments to validate a process without considering its model. In (Hug, 2010), we describe a qualitative evaluation of a method for information systems engineering processes.

2.2 Formalization with Patterns

Many approaches exist to validate a new language or a new process. But there are rarely presented in a reusable way. To be reusable, they must present practical guidelines such as those proposed in (Aranda, 2007) and (Patig, 2008). These works define generic experimental protocols for models understandability. We want to promote such proposals by developing these guidelines in a well-known approach: design patterns.

A pattern proposes a solution to a recurring problem occurring in a given context. Patterns are used to represent both knowledge and know-how that are linked together through methodological guidance.

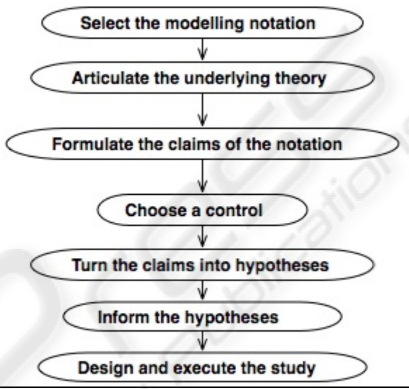
In our context, patterns can be used to provide techniques and tools for capitalizing knowledge and know-how of the validation domain. A pattern allows to identify a problem to resolve (for example, how to evaluate the understandability of a notation), proposes a generic and correct solution to this problem (for example, the generic protocol proposed by (Aranda, 2007)) and finally offers indications to adapt this solution to a particular context (for instance, how to apply the protocol to UML notation). In table 1, the solution of the pattern, i.e. the protocol, is a know-how represented by an activity diagram. The first step is to select the notation. Then the assumptions about the notation must be identified ("Articulate the underlying theory" activity). The third activity is to formulate the claims of the notation regarding comprehension. A control must be chosen in order to have a baseline for comparison. Then the claims are turned into testable hypotheses. Other research area can also bring some insights about the hypotheses ("inform the hypotheses"). Finally the study itself is designed and executed.

In table 1, the pattern describes a know-how; so it is a process pattern. A **process pattern** details the steps to follow to reach a goal (for example, how to evaluate understandability, how to formulate the claims of the notation etc) whereas a **product pattern** permits expressing a goal to reach (for example, "What are the characteristics of languages quality?").

Finally, problems have still to be solved regarding pattern organization in order to ensure effective reuse. Patterns must organize hierarchically and functionally problems and the manner to resolve them. They form hence an engineering guide called patterns catalogues. For the understandability evaluation problem, all these activities of the pattern solution (Table 1) can also be represented by patterns that would be linked in order to show their dependency. So we will obtain a catalogue of 8 patterns for "understandability evaluation": one for the global protocol and one for each of its composing activity.

With patterns, all the different approaches for evaluation can be described and presented in an uniform way so that it is easier to find the adequate solution of a given problem. We hope that this can help in making evaluation more systematic.

Table 1: Pattern Example.

Pattern name	Understandability evaluation
Problem	how to evaluate the understandability of a notation?
Motiva-tion	The effectiveness of models depends on the communication quality of their languages. This quality aspect relies partly on understandability, which can be validated with this solution.
Solution	 Select the modelling notation Articulate the underlying theory Formulate the claims of the notation Choose a control Turn the claims into hypotheses Inform the hypotheses Design and execute the study

3 TOWARDS AN EVALUATION COMMUNITY

3.1 Definition of Collaborative Patterns

Usually, patterns express consensual solutions of an expertise domain. Then patterns are formulated by domain experts and are used by a whole community. Recently, the pattern concept has become a way of sharing and concealing community knowledge (ODP, 2010): the "best" solution can be at any time reconsidered and then, modified. The pattern does not belong to a person, but to a set of experts who work together in order to consolidate the knowledge of their domain. Patterns are becoming collaborative.

We propose to manage collaborative patterns by adapting the Lécaille's work on the evolution of a type of objects: the *digigraphics* during design according to three action modalities (Lecaille, 2003):

- *Draft* is an object on which we apply the modalities of creation and validation of hypothesis or solutions to a problem. They are defined by the design actor herself or conjointly with other actors who use graphics represented *in traceable objects, printings or in screen views.*

- *Exhibit* is an object on which we apply a persuasion modality in accordance with what is represented either for convincing about the existence of a problem or for showing a solution and allowing a common construction and the exchange of the point of view.
- *Enabled trace* is a traceable or digigraphic object that one applies a modality of circulation without constraints. The creator accepts to diffuse it to the others, after her consent or her agreement with a collective prescription which she takes part to.

In our work, we consider *Draft*, *Exhibit* and *Enabled trace* as the three possible states of a collaborative pattern (Fig. 1).

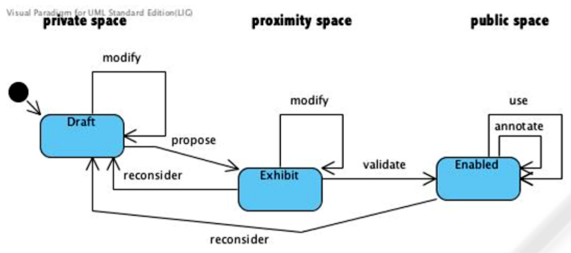


Figure 1: Collaborative pattern lifecycle.

Dr*afts* are kept in the *personal* (*private*) *workspace* of a community member. Then the creator needs to confront her ideas with other members' point of view; she proposes the pattern to her *proximity workspace* based on her personal network and her loyal relationships. In the proximity workspace, the creator can expose herself to critics and judgment of others. When the collaborative pattern is considered as enabled, it is validated to be transmitted outside the personal network in the *public space*. In the public space, a pattern cannot be modified but it can be used (i.e. selected and adapted to a given problem). It can also be annotated (i.e. commented with marks or evolution suggestions) so that the pattern creator can decide to reconsider it if necessary.

The operations on patterns cannot be realized by anyone. We distinguish 3 roles for a pattern:

- Pattern owners are the pattern creator and any other person that the creator has accepted as owner. They can do all operations on their patterns. In our case, owners will be experts in evaluation.
- Pattern collaborators are people in the proximity workspace of the pattern owners. They can view and modify the pattern when it is in the "Exhibit" state. For the domain of information systems design validation, collaborators can be other specialists in evaluation, but also specialists in information systems design. Thus the evolution of a pattern is a cooperative activity where the experts of two different domains must work together in order to create a valid solution.
- Readers can view, use and annotate the pattern when it is in the public space. Readers are non-specialists of the pattern domain who are simply looking for information. They can be information system specialists looking for a way of validating their proposal or evaluation specialists who search for new evaluation solutions.

Collaborative patterns seem to be a solution to construct, gather and share the evaluation knowledge for information systems design. However to be easily used, they need to be available with a common tool support.

3.2 Tool Support

In our vision, the evaluation community will share their knowledge in a web site. This web site will present the community (their actors, their goals…). It will also support discussion in a wiki or in a forum in order to allow the community members to contribute to the community life. The global community will be managed by a moderator who will be responsible for accepting new members, validating patterns,…(Fig. 2).

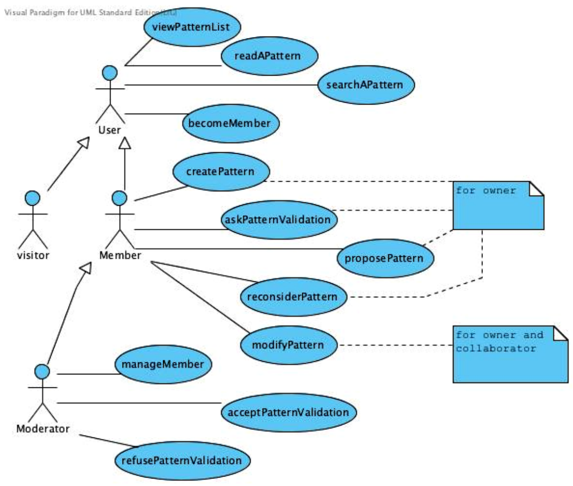


Figure 2: Main use cases.

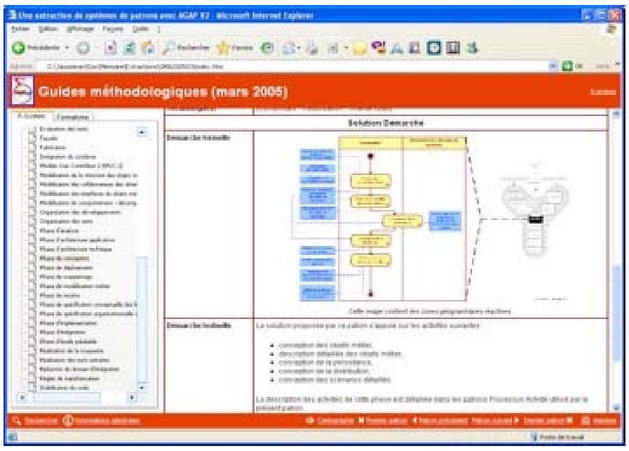


Figure 3: Website generated by AGAP.

There are also two other actors inside a community:

- Visitors who are not yet members of the community. They can only view and search for patterns.
- Members who have adhered to the community. They can use and annotate patterns. They can also create patterns, becoming then "pattern owner". And if they are "pattern collaborator" of a pattern, they can modify this specific pattern.

The evaluation community web site will also contain collaborative patterns. Each pattern will be described in a web page on which it will be possible to realize the operations identified in the previous section: creation, modification, use, reconsideration, validation, but also sharing with the proximity network or the community. For each pattern, we will propose a forum to keep discussion opened and to permit annotations.

As the management of the evaluation community must be similar to the one of any other community, we are developing a tool for managing collaborative patterns in general. Our tool, C-OPEN (COllaborative Pattern Environment), will permit the creation of a new community (for instance the evaluation one) and their management.

This tool will be based on the principles of the AGAP tool (Conte, 2002) that was designed to support pattern catalogues. AGAP allows designers to enter information for patterns (structure, context, motivation, solution, links between patterns etc) and then to generate an autonomous website corresponding to a patterns catalogue.

Fig.3 shows a screenshot of the main page of a patterns catalogue, which represents the activities of the Symphony development method. The main part of the interface is dedicated to the display of the pattern chosen by the user. In the pattern, the solution, represented by a model (here an activity diagram), can be selected in order to be reused. Finally all the patterns are browsable with the list presented on the left. They can also be found by a search function.

C-OPEN will add the management of the collaborative aspects that we have described previously to AGAP. It will be implemented using the Alfresco content management solution in order to facilitate its maintenance.

4 CONCLUSIONS AND FURTHER WORK

This paper argues for a more systematic validation of the research proposals of the information systems design domain. In order to achieve this goal, we would like to create an evaluation community where evaluation specialists could share their knowledge via collaborative patterns. Patterns are viewed not only as a way of describing knowledge collaboratively, but also as a mean to share this knowledge with non-specialists. A specific web site can then present the knowledge into a user-friendly way.

Currently we are working with evaluation specialists so as to identify and describe some of their knowledge. This will give us our first patterns for information systems design evaluation.

Moreover, we will be soon able to propose a tool support for the community so that it will be easy share knowledge about evaluation. The first community supported by the tool will be the evaluation one with the patterns that we are currently identifying. It will give us the starting point of the evaluation community for information systems design. We hope that the existence of this community will encourage information systems specialists to validate their proposals.

REFERENCES

Aranda J., Ernst N., Horkoff J., Easterbrook S., 2007, A Framework for Empirical Evaluation of Model Comprehensibility", In *Int. Workshop on Modeling in Software Engineering (MISE'07),* IEEE.

Ceret E., Dupuy-Chessa S., Godet-Bar G., 2010, Using Software Metrics in the Evaluation of a Conceptual Component, *In Proc. Of the 4th Int. Conf. On research Challenge in Information Science RCIS'2010,* IEEE, France.

Conte A., Giraudin J. P., Hassine I., Rieu D., 2002,Un environnement et un formalisme pour la définition, la gestion et l'application de patrons, revue Ingénierie des Systèmes d'Information (ISI), volume 6 N°2, Hermès. [in french]

Dupuy-Chessa S., 2009, Quality in Ubiquitous Information System Design, *In Proc of the 3rd Int. Conf. On research Challenge in Information Science RCIS'2009,* IEEE, Maroc,

Hug C., Mandran N., Front A., Rieu D., 2010, Qualitative Evaluation of a Method for Information Systems Engineering Processes, *In Proc. Of the 4th Int. Conf. On research Challenge in Information Science RCIS'2010,* IEEE, France.

Godet-Bar G., Dupuy-Chessa S., Rieu D., 2008, When interaction choices trigger business evolution, *In Proc of 20th International Conference on Advanced Information Systems Engineering (CAiSE'08),* LNCS 5074, Springer, Montpellier, France, pp 144-147.

Krogstie J., 1998, Integrating the Understanding of Quality in Requirements Specification and Conceptual Modeling. Software Engineering Notes, ACM SIGSOFT, 23(1), pages 86-91.

Krogstie J., 2003, Evaluating UML Using a Generic Quality Framework, chapter UML and the Unified Process, Idea Group Publishing, pages 1-22.

Lécaille P., 2003, La trace-habilitée, une éthnographie des espaces de conception dans un bureau d'études mécanique: l'échange et l'équipement des objets grapho-numériques entre outils et acteurs de la conception, PhD dissertation in INPGrenoble, novembre 2003. [in french]

Lange C., Chaudron M., 2005, Managing Model Quality in UML-based Software Development", In *Proc. Of the 13th Int. Workshop on Software Technology and Engineering Practice (STEP'05),* pages 7-16.

Lindland O. I., Sindre G., Solvberg, A., 1994, Understanding quality in conceptual modeling. 1EEE Software, pages 42-49.

Mendling J., Nemann G., Van Der Aalst W., 2007, On correlation between process Model Metrics and Errors, *In Proc. 26th Int Conference on Conceptual Model (ER'2007),*New Zealand.

Mohagheghi P., Aagedal J., 2007, Evaluating Quality in Model-Driven Engineering, In *Int. Workshop on Modeling in Software Engineering (MISE'2007),* IEEE.

Moody, D. L., 2003, The Method Evaluation Model: a Theorical Model for Validating Information Systems Design Methods. In *11th European Conference on Information Systems ECIS 2003.*

Moody D., 2005, Theoretical and practical issues in evaluating the quality of conceptual models: current state and future directions, Data & Knowledge Engineering, Vol 55, pages 243-276.

Nelson J., Poels G., Genero M., Piattini M., 2005, Quality in Conceptual Modeling: five examples of the state of the art", Data & Knowledge Engineering, vol 55, pages 237-242.

Ontology Design Patterns web site, 2010, http:// ontologydesignpatterns.org/wiki/Main_Page (last consultation March 2010).

Palen L., 2002. *Beyond the Handset: Designing for Wireless Communications usability*. ACM Transactions on Computer-Human Interaction, 9(2), pages 125-151.

Patig S., 2008, A practical guide to testing the understandability of notations, Conferences in Research and Practice in Information Technology Series; Vol. 325, *In Proc. of the fifth on Asia-Pacific conference on conceptual modelling* - Volume 7, pages 49-58

Siau K., Tian Y., 2001, The Complexity of Unified Modeling Language: A GOMS Analysis, *In Proc. Of the 22th Int. Conference on Information Systems,* pages 443-447.

Rossi M., Brinkkemper S., 1996, Complexity Metrics for System Development Methods and Techniques, Information Systems, Vol. 21, num. 2, pages 209-227.

Si-Said Cherfi S., Akoka J., Comyn-Wattiau I., 2007, Perceived vs. Measured Quality of Conceptual Schemas: An Experimental Comparison . In Tutorials, posters, panels and industrial contributions at the 26th International Conference on Conc, Grundy, John and Hartmann.

Tichy W., Lukowicz P., Prechelt L., Heinz E., 1997, Experimental evaluation in computer science: a quantitative study. Journal of Systems Software, 9, pages 9-18.

Zelkowitz M., Wallace D., 1997, Experimental validation in software engineering, Information and Software Technology, 39, pages 735-743.