OOPSLA 2000 Workshop: Mining Pedagogical Patterns

Submission by Markus Voelter
voelter@acm.org

ABSTRACT

This paper is a submission to the OOPSLA 2000 workshop "Mining Pedagogical Patterns". It briefly describes some of the problems of the current state of the PPP and tries to give some proposals as to how to improve the situation. In particular, it describes an alternative form for pedagogical patterns, it tries to formalize a schema for categorization of the patterns and the relationships among them, and it defines an XML based notation for the patterns.

Introduction

For the reader of the PPP, the PPP has a very non-uniform appearance. This is because of the following reasons:

- The Pedagogical Pattern Project currently consists of a collection of so-called proto-patterns. These patterns have different levels of maturity. Some have been workshoped on a PLoP conference, some haven't.
- The patterns have different scope. Some are of a more general nature, some are very specific. There is no way for the reader to distinguish the different kinds of patterns.
- The patterns are not related to each other, i.e. they do not form a pattern language. In addition, it is very hard to find a pattern for a specific problem.

Proposed Solutions

One way to resolve those problems is the one that is addressed during this workshop: The proto-patterns are read by the workshop participants, with the goal to find commonalities and basic principles in the patterns, a kind of refactoring process. However, in the long run, this is not enough. The following additional steps should be considered:

- It is necessary to create a homogeneous look-and-feel for the patterns. The selected pattern form should be very shallow to allow the use of the form for as many patterns as possible.
- The patterns should be categorized regarding several different aspects to facilitate search and retrieval functions.
- The patterns should be related to each other to create a kind of pattern language that can dynamically grow when new patterns are added.
- The patterns should be stored in a way that makes automatic processing possible.

The following sections propose some solutions for these aspects.

Aspects of the Solution

Homogeneous Look and Feel – Pattern Form

To make the pattern form usable for as many authors as possible, the form needs to be very non-restrictive. The current form consists of more than ten sections. For many pedagogical patterns, this is too strict. In contrast, a couple of other forms have been used by pedagogical pattern authors. For example:

- Name – Problem – Forces – Solution – Dicussion in *Learning to Teach and Learning to Learn* by *Jutta Eckstein*, or
- Name - Problem/Issue – Audience/Context – Forces – Solution – Dicussion – Resources – Related Patterns – Example Instances – Contraindications – References in *Fourteen Pedagogical Patterns* by *Joe Bergin*
- Name – Problem - Solution outline, consequences, drawbacks - Examples and additional implementation information in *SEMINARS* by *Astrid Fricke* and *Markus Voelter*

As can be seen, these forms are different from the one used in the PPP. To find a common form, the following guidelines should be considered:

- The form should provide a rough skeleton and should not force the pattern author to introduce artificial sections that do not come naturally.
- The general form should allow more specific forms through the introduction of subsections.
- The form should be suitable for the different kinds of patterns as discussed below.

Therefore, the following pattern form is proposed as a basis for further discussion (sections in parentheses are optional subheadings):

Name - **Author** (Version, Release date, etc.) - **Context** (Audience, Indications, Contraindications) - **Problem** (Forces) - **Solution** (Discussion, Resulting Context, Consequences, Drawbacks, Necessary Resources) – **Examples**

This form gives the author sufficient freedom and can still be detailed as far as necessary.

Pattern Categorization

To allow the reader a more sophisticated search and retrieval capability, the patterns need to be categorized. It is especially important to allow the categorization in different dimensions, as different users might have different categorization requirements.

As a first proposal, the patterns could be categorized along the following dimensions:

Time When, in the context of a seminar or a lecture, can the pattern be employed. Possible values could be:

- during seminar preparation
- just before the seminar starts
- when the seminar runs
- after the seminar

Aspect Which aspect of the overall problem space does the pattern address? Possible values could be:

- Social: The pattern tries to enhance the social relationships in the group
- Environment: The pattern tries to enhance the physical environment in which the training/seminar/etc. is run.
- Pedagogical: The pattern deals with the pedagogical aspect, i.e. how to teach something.
- Technical: The pattern gives hints on what to teach.
- Organizational: How to organize the schedule/etc. of a pattern

Scope Defines how broadly the pattern can be applied, some possible values could be:

- general: The pattern can be used in every setting
- academic: The patter can be used in an academic setting only.
- domain-specific: The pattern applies to a specific domain only.

Domain Defines the domain to which the pattern applies.

Of course, there are interrelation between these categories, e.g. if a pattern only applies to a specific domain (scope), then this domain must be specified in the domain category.

For some categories, multiple values are possible.

Pattern relationships

To form a systematic pattern catalog – or even a pattern language – there must be a way to define relations among patterns. These relations should be defined in a standard way, together with a descriptive comment, to allow automatic processing. Among others, such relations could be the following:

Generalization / Specialization	One pattern can generalize/specialize another pattern. This can be reflected by relaxing/constricting the pattern categorization. For example, if the scope of a certain pattern only applies to a specific domain, then a generalized pattern could relax this scope to be general.
Requires as predecessor/has as successor	A pattern can require another pattern to define the context for the pattern, or the pattern can define the context for another one.
Can be supported by	Another pattern could help in implementing the current pattern
Can be implemented by	Other patterns can be used to implement the current pattern
Alternative	A pattern can be a an alternative to another pattern.

Pattern Storage and retrieval – Technical implementation

STRUCTURE

The following paragraphs show a pattern from the *SEMINARS* pattern language. The pattern is printed in its current (MS Word97) format, together with relations and categorizations annotated:

Introduction Session *

To make PERSONAL COMMUNICATION possible during the rest of the seminar, the participants need to learn something about each other at the beginning. Usually, the participants are a little bit shy and you should start this process. In addition, you might want to learn something about the participants, to ADAPT TO THE PARTICIPANTS BACKGROUND or to LET THEM DECIDE.

***

Therefore, take the time at the beginning of the seminar to let everybody introduce him-/herself to the others. The participants should be given a chance to state their expectations towards the seminar and tell the others about their professional background, their company, et cetera. This session should be held in an informal context, which can be achieved by using a suitable TABLE ARRANGEMENT. It is also possible to use GAMES at the beginning of such a session. To make a start, you should begin the session by introducing yourself. Be sure to introduce yourself, not the seminar.

There are different ways how this introduction session can be held. The most common form is that everybody introduces himself to the others, including his name, employer, his field of activity, et cetera. In general, the contents depend heavily on the clientele (imaging the difference between programmers and pedagogists). It is a good idea to let the participants decide what they want to include in their introduction. An alternative form is to let one person interview another person and introduce this other person to the group. The introduction session usually ends with everybody attaching a NAMEPLATE to himself.

Categorization
Time: During a Seminar
Aspect: Social, Organizational
Scope: *

Relations
Precondition for: Personal Comm: Personal communication can only take place if there is room for it.
Precondition for: Adapt to Participants Background:
Precondition for: Let them Decide:
Can be implemented by: Games:
Can be supported by: Table Arrangement:
Can be supported by: Nameplate:

By providing a suitable hypertext representation for this pattern, together with a powerful search engine, a kind of dynamic, extensible pattern language can be implemented.

NOTATION IN XML

The requirements outlined above lend themselves to be implemented using XML. A common DTD for the patterns can be defined. By using XSL, different renderings of the patterns can be defined. A transformation to HTML is possible, there are also tools to create PDF for printing. The following is the above pattern as an XML document:

```
<pattern id="IntroductionSection">
  <title>Introduction section</title>
  <administrative>
    <author email="voelter@acm.org">Markus Voelter</author>
    <author email="astrid.fricke@gmx.de">Astrid Fricke</author>
```

```
    <version>1.2</version>
    <date>2000-08-16</date>
  </administrative>
  <categorization>
    <category name="time" value="during a seminar"/>
    <category name="aspect" value="social"/>
    <category name="aspect" value="organizational"/>
    <category name="scope" value="*"/>
  </categorization>
  <relations>
    <relation type="precondition for" destination="PersonalCommunication">
       Personal communication can only take place if there is room for it.
    </relation>
    <relation type="precondition for"
destination="AdaptToParticipantsBackgound">
    </relation>
    <relation type="precondition for" destination="LetThemDecide">
    </relation>
    <relation type="Can be implemented by" destination="Games">
    </relation>
    <relation type="Can be supported by" destination="Table Arrangement">
    </relation>
    <relation type="Can be supported by" destination="Nameplate">
    </relation>
  </relations>
  <content>
    <context>
    </context>
    <problem>
       To make PERSONAL COMMUNICATION possible during the rest of the seminar,
       the participants need to learn something about each other at the
       beginning. Usually, the participants are a little bit shy and you
       should start this process. In addition, you might want to learn
       something about the participants, to ADAPT TO THE PARTICIPANTS
       BACKGROUND or to LET THEM DECIDE.
    </problem>
    <solution>
      Therefore, take the time at the beginning of the seminar to let everybody
      introduce him-/herself to the others. The participants should be given a
      chance to state their expectations towards the seminar and tell the
      others about their professional background, their company, et cetera.
      This session should be held in an informal context, which can be achieved
      by using a suitable TABLE ARRANGEMENT. It is also possible to use GAMES
      at the beginning of such a session. To make a start, you should begin the
      session by introducing yourself. Be sure to introduce yourself, not the
      seminar.
    </solution>
    <examples>
      There are different ways how this introduction session can be held.
      The most common form is that everybody introduces himself to the others,
      including his name, employer, his field of activity, et cetera. In
      general, the contents depend heavily on the clientele (imaging the
      difference between programmers and pedagogists). It is a good idea
      to let the participants decide what they want to include in their
      introduction. An alternative form is to let one person interview
      another person and introduce this other person to the group. The
      introduction session usually ends with everybody attaching a
      NAMEPLATE to himself.
    </example>
  </content>
```

```
</pattern>
```

A possible XML DTD could look like the following.

```
<!ELEMENT pattern (title, administrative, categorization, relations, content)>
<!ATTLIST pattern
  id CDATA #REQUIRED
>
<!ELEMENT title (#PCDATA)>
<!ELEMENT administrative (author+, version?, date?)>
<!ELEMENT author (#PCDATA)>
<!ATTLIST author
  email CDATA #REQUIRED
>
<!ELEMENT version (#PCDATA)>
<!ELEMENT date (#PCDATA)>
<!ELEMENT categorization (category*)>
<!ATTLIST category
  name CDATA #REQUIRED
  value CDATA #REQUIRED
>
<!ELEMENT relations (relation*)>
<!ELEMENT relation (#PCDATA)>
<!ATTLIST relation
  type CDATA #REQUIRED
  destination CDATA #REQUIRED
>
<!ELEMENT content (context, problem, solution, examples)>
<!ELEMENT context (audience?, indications?, contraindications?)>
<!ELEMENT problem (forces?)>
<!ELEMENT solution (discussion?, resultingContext?, consequences?, drawbacks?,
necessaryResources?)>
<!ELEMENT examples (#PCDATA)>
```

SEARCH AND RETRIEVAL

To make a pedagogical pattern repository really useful, there must be different ways on how to search and browse the patterns.

- The simplest form is full text searching, i.e. there is no guidance for the user about which values can be searched for.
- Another possibility to implement the search algorithm is to allow the user to specify (one or more) values for some of the pattern categories, e.g. the user could search for IT-specific patterns, that address the pedagogical aspect during a seminar.
- When the user has found something, he should be able to relax/constrict some of the categories, and he should be able to follow the pattern's relations to discover other, related patterns.

Doing this, a "dynamic pattern language" can be created, which can easily cope with relations among patterns, with domain-specific patterns as well as general patterns, etc.