Abstract

This paper reports our experience of designing a human-centric repository to support learning object authoring. The users of DesignPatterns.ca are designers who want to explore, communicate and share parametric modeling knowledge and strategies. Aiming to support such a community, we provided succinct functions to meet their evident needs, kept the effort of packaging and publishing to a minimum, and left space for user innovation. We used agile methods and negotiated details with users regularly. The paper records and critically reviews the process we followed and lessons we learned in this repository development.

1. Introduction

This paper introduces and discusses our experience of developing the DesignPatterns.ca, a learning object repository and authoring platform for teaching parametric modeling. We design learning objects for parametric modeling based on the software Bentley’s Generative Components (GC) [1]. Our main client group is SmartGeometry, whose workshops around the world have successfully established a large community of GC users. We want our learning objects to not only teach new users parametric modeling skills but also assist existing users in developing new strategies. Furthermore, we want to encourage more users to author new learning objects and share their ideas during the process. In a busy working environment, even if people are willing to share their experience and outcomes, they are reluctant to invest precious working time to do so. We needed to develop easy and efficient channels to encourage authoring and sharing. Due to of the complexity of the software and variety of user roles, at the outset we imagined a complicated repository system to support the project. Surprisingly, we ended up with using a simple system to solve the problem. Starting from a basic FTP webspace, we developed the repository simultaneously during the learning object design process and obtained a very simple system that is both efficient and effective.

2. Background of this project

Our learning objects in the repository demonstrate and discuss common strategies to solve various parametric modeling problems using the metaphor of software patterns. We use patterns to encapsulate these learning objects and to provide a rhetorical structure common to all patterns.

2.1. Parametric Modeling

Parametric modeling software has introduced into practice, computational mechanisms and interfaces for representing variation in design. These work best when variation is continuous and distinctly different alternatives are not part of the model. Using parametric modeling, it is possible to develop models that support discrete variation, but it is very difficult to understand the range of possibilities entailed by such models. Parametric modeling interfaces thus provide partial support for expressing variation and, because they are increasingly used in practice, a means by which new variation techniques can be tested in actual use. Bentley’s Generative Components (GC) [1], developed by Robert Aish, formerly Chief Scientist at Bentley Systems, provides an environment in which geometry (lines, arcs, circles, solids, and surfaces) can be related, transformed, generated and manipulated within a user-defined framework. The end results are complex and sculpted geometry that can be quickly generated and manipulated in real-time, allowing design exploration and variation. Due to its active development and large beta-testing community, we chose to use GC as the software to teach parametric modeling.

2.2. Design Patterns as Learning Objects

Experts use their experience of solving problems in the past to build on and create new solutions in new situations. Such experience is part of what makes them experts. Some of these reusable solutions can be expressed in what are known as design patterns. Originating with Christopher Alexander in 1977[2],
the term design pattern expresses design work at a tactical level, above simple editing and below overall conception. In the process of parametric design, patterns can be observed as a general repeatable solution to a recurring modeling problem. Architects may use the same pattern in different circumstances and may also derive new patterns as they work. This concept originated in urban architecture but has been adapted quite successfully to software engineering [6] and extended to other disciplines such as interaction design [16], web usability [7], education science [4] and communication [14].

Patterns provide a language for communication among collaborators. Rather than having to explain a complex idea from scratch, the group of people can just mention a pattern by name and everyone will know, at least roughly, what is meant. In this sense patterns act as an excellent vehicle for communication. A learning object is defined as any entity, digital or non-digital, that may be used for learning, education or training [9]. It is very essential for learning objects to be communicative between instructors and students (or experts and novices). The sophisticated structure of pattern provides a useful template to construct such a learning object.

Our parametric modeling learning objects are constructed as an adaptation of Jenifer Tidwell’s UI pattern [16] expression model. There are mainly six components in a learning object: diagram, intent (analogue to Tidwell’s “what”), use when, why, how and samples.

- The diagram is a graphic representation of the pattern.
- Intent states a one-sentence description of the goal behind the pattern.
- Use When describes a scenario consisting of a problem and a context.
- Why states the reasons to use this pattern.
- How explains the details of how to adopt the pattern to solve the given problem.
- Samples illustrate how the patterns can be used in several different contexts.

2.3. XML Template for Authoring

After authoring independent learning objects, it is critical to preserve and represent the semantic relations among documents from different learning objects in a repository. Structured documents associate explicit semantics with content, but authoring rigorously structured documents is a very difficult task [13]. Tim Berners-Lee argues that “the concept of machine-understandable documents does not imply some magical artificial intelligence which allows machines to comprehend human mumblings. It only indicates a machine’s ability to solve a well-defined problem by performing well-defined operations on existing well-defined data. Instead of asking machines to understand people’s language, it involves asking people to make the extra effort” [5]. The semantic web offers a collection of models and languages such as XML, XML Schema, RDF, OWL, SPARQL to handle various aspects of document semantics under the form of “well-defined data”. Each model or language plays a different role, and all are well organized in the so-called semantic web layer cake [8]. But the “extra effort” that people have to make to produce these data is the key issue, which probably explains why the semantic web is not deploying as fast as the traditional web. Most document authors do not spend additional time to formally encode in a computer language (some aspects of) the information and thoughts they have already expressed in natural language or through drawings and pictures.

We focus on the first levels of the semantic web, namely XML and its schema languages, trying to help authors to safely produce “well-defined data” while they are writing documents. XML’s goal is to explicitly represent the logical structure of documents and data. With this structure, programs can securely and unambiguously identify the various parts of a document or a data base, and they get some hints about the type of information located in each part. Structure provides context for the content.

3. Our experience with LO Repositories

Education covers a broad range and can exist in different forms. During the past three years, we were involved in several projects developing learning objects and their repositories. In the project of “Learning Objects for Design”, the repository is as simple as a collection of semantically linked “objects” [11]. “GeometryWare” demonstrates the LO repository could be an independent learning platform which including not only presentation and demonstration but also implementation and submission [10]. Aiming to teach spatial computing uniquely, GeometryWare structure includes a complete learning process. AVIRE (A VIsual REte) is a generic repository for visual material related to cultural disciplines. AVIRE is designed to be an online space where different users (such as curators, exhibitors, critics and viewers) play together to create a large social entity. Since its users
could upload new resources, organize exhibitions, and annotate resources and exhibitions in the system. A\textbullet{}ViRE demonstrates great capability and obvious complexity [12]. A\textbullet{}ViRE was constructed inside the Tikiwiki portal-building system. It inherits many powerful functions, but suffers great difficulty in customization and further development. We also participate in administrating an enterprise-level repository – the BCCampus Shareable Online Learning Resources (SOLR) (available at http://solr.bccampus.ca). The repository is built based on the Equella system of the Learning Edge International. Such systems work in a relatively large scale, for example, at institution level, being able to handle hundreds of courses with different teaching methods and content arrangements, as well accommodate different standards such as SCORM and IEEE LOM and different online education tools (e.g. WebCT and MOODLE). These repositories are so complex that normally they require a dedicated support team and administrators to support authors and end users.

When we considered the design of this parametric modeling LO repository, we wanted it to have a simple structure as “Learning Object for Design” but direct authoring and interaction as in “Geometry Ware”. Users do need powerful functions, but we want fulfill their needs and still retain system efficiency. Since there would not be a dedicated system support team, the user roles in the repository should be clear, straightforward and result in correctly structured objects.

4. Consider the Needs of Target Users

The repository aims to teach and communicate common strategies used in the parametric modeling process. Target users could be undergraduate and graduate students, architects, civil engineers, constructors, industrial designers or any 3D designers who are interested in parametric modeling and want to use the new tool in their design practice. Architecture education has the tradition of teaching in studios and instructing individually or on a small group basis. We needed to work within this tradition. Thus our learning objects are not structured as course materials. Rather they are devices to structure dialogue around good experience and useful “tricks” experts have and novices need. Also in the studio tradition, it is very common for new users to post their new findings to communicate with each other or experts. This repository should make it easy for them to put their work online as new learning objects.

On the other hand, we want the repository to be simple to use. Most of our users are computer experts only in their narrow field of computer-aided design. Being simple does not mean the system has as few functions as possible (sometimes having less functions does not necessarily achieve smooth navigation experience). Rather, it means that the tasks are clear and easily accomplished. In general, it is impossible to understand the tasks that are actually needed at the outset of system design. This occurs because the act of using a system changes the tasks that the system must support. Therefore, the tasks supported and the resulting system must converge over time if the system is to be successful.

We began with the user roles we developed for the A\textbullet{}ViRE project and reformulated the use cases for each role in the patterns context. There are four user roles in our system: curator, author, critic and visitor. Our development team took the role of curator. We discussed the flow of the content, arranged the sequence of learning objects and edited the links across different objects. For the other three roles, we have developed use cases by observing, meeting and discussing with users during the development period.

4.1. Use Case1: an author

“I learned the theories of spatial computing in my graduate school. During the past three years, I have used Generative Components frequently in my own design projects. We have our discussion forum and need more participants in the community to share ideas. Through authoring some pattern learning objects, I want to show new users the significant potentials of parametric modeling and share my experience with examples. Having a busy life, I can’t afford a lot of time on organizing and formatting the outcomes. I want to get concentrated on the text writing and the example creation. If I made any mistakes even after publication, I want to be able to fix it quickly and directly.”

By observing the authors, we found three main parts in the learning object authoring process: preparing materials, organizing the content and publishing the learning object. To prepare for a pattern, the author needs to write introduction text, draw an expressive diagram, build several parametric examples in Generative Components and capture a sequence of screenshots. After most of these materials are ready, they will start to organize all these materials into the outcome: a document (mostly a webpage or MSWord document) composed by text, diagram, images, links to source files, and sometimes movies or
animations, and then publish it. The main intellectual effort of authoring is in preparation, a task mostly beyond automated assistance. We focused on supporting he acts of composing and publishing. Our tools for this included both scripts within Generative Components and website features, for example, a simple “drag + drop” uploading function. A key lesson here is that the system implementation is likely to cross the boundary between online tools and the off-line process of creating learning objects.

4.2. Use Case2: a critic

“I am familiar with the theories and applications of parametric modeling. I am quite interested in the area and have attended the SmartGeometry workshop before. Although I do not have a lot of implementation experience to share, I think my background knowledge can provide some useful advice to those learning objects. Firstly, I want to be able to download all the source files of a learning object so that I can have a close look at the script code. But I hate to use right clicks and download items one by one. I also want to ask questions, leave comments or suggest better solutions on the ones I am interested in. In the future when I get enough knowledge about parametric modeling, I wish I can also participate and contribute.”

Critics are the group of “serious readers”. They are usually interested in the source code and want to leave comments. To make their job simple, we should allow them to download a learning object as a package. They should be able to jump across different learning objects, leave annotations and become an author or even a curator.

4.3. Use Case3: a visitor

“My architecture classmate Jack told me that this repository was interesting. Before that, I have never heard of adopting parametric modeling in the design projects, but I definitely want to see new and fun stuff. I do not have the time to install a new application, learn its basic functions and run a file to see how it works. So, if I can understand the thing directly while I am just reading a webpage, it would be brilliant. If it is really as exciting as Jack describes, I would go to tell my other friends immediately.”

For the “visiting” users, we want to make our system as easy and friendly as an interactive reference book. They should be able to easily understand each learning object by reading its description, looking at the 3D model and considering the suggested solution. In the webpage, we should find a way to simulate the changes taking place in the application so that they do not need to try it by themselves. The interface design should be clean and the text should be very easy to read. The system should also be able to welcome a large increase in the number of users in the future. The size of multimedia documents should be relatively small so that users can download and view them without much delay.

Although there are only four kinds of users in this repository, we have to understand and meet their different needs.

5. Foundation and Development Method

5.1. Start from a FTP based System

Our system originates from a simple FTP based repository. In a SmartGeometry workshop held in February 2006, participants (mostly architects or civil engineers) needed a centralized place to share their ongoing work and present their outcomes at the end of the workshop. We had a powerful multimedia repository A•Vi•RE ready. However, the workshop was so intensive that there was absolutely no time for these participants to organize their work into nice looking PowerPoint files or web pages. Few users were willing to use the standard html form’s file upload function to upload many files one by one. They were also reluctant to use tools (e.g. WinZip) to compress files (some participants are not familiar with computer tools apart from CAD applications). Their most familiar and preferred way is use drag + drop to copy files directly from one window to another. Thus, we decided to create a FTP space in a web server and use server side technology to generate webpages (available at http://smartgeometry.designscience.ca).

In this space, each participant can create his/her own folder, use drag + drop to upload all their files from a local folder to the ftp site through their browser (such as IE 6) directly. Meanwhile, s/he could write a text file in any text editor (such as Notepad or TextPad) and name it as README to descript his/her work. A server page (written in ASP.net) automatically generates a webpage based on the uploaded files. In this webpage, text in README is placed on the top, uploaded images are aligned in a sequence and source files are listed as external links at the end. Although the system is extremely simple (the author can not even control the layout of rendered webpage), it turned out to be quite successful, supporting more than 60 participants to upload, share and present their work in a 4-day workshop. After the workshop, we improved the README file into structured text. We introduced
some Wiki rules to provide limited authoring flexibility. People can use simple HTML tags (if they know any) and several simple rules to compose the webpage. This simple FTP/Web repository served hundreds of participants in the following GC workshops for a year.

During the last year, the FTP space became improper for the workshop groups because the population of GC users increased rapidly. It became difficult to manage many FTP usernames and passwords directly. FTP itself also has quite a few security issues. New versions of web browsers no longer support the drag + drop function. Users have to open a FTP client application to upload files. To save our user’s time and enhance system security, we integrated a third party drag + drop Java Applet into the system. Currently this system is still serving the SmartGeometry workshops. Participants use it to collect and communicate their works. This experience encouraged us to build a repository system with better functions based on the same design principles – simple, quick and useful.

5.2. Agile Software Development Method

Agile development methods attempt to achieve faster and nimbler software development processes. They were firstly published by a group of software practitioners and consultants in 2001 [3]. The fundamental principles of agile methods include:

- Close and regular cooperation between users and developer.
- User and developer work in the same lab to meet face-to-face as the primary communication.
- All group members are motivated and are trusted.
- Continuous attention to technical excellence and good design.
- Simplicity is always the highest priority.
- As the project proceeds, the user may continuously change their requirements. Any late changes are welcomed.
- The working software with changes is delivered frequently.

There are many kinds of existing agile methods such as extreme programming, Scrum, crystal family of methodologies, feature driven development and the rational unified process [3]. For us, the critical aspects of agile methods are simplicity and speed, which match our design goal of the new learning object repository. We selected the loose form of Dynamic System Development Method (DSDM). The fundamental idea behind DSDM is that, instead of firstly fixing the amount of functionality in a product, it is preferred to fix time and resources and adjust the functionality accordingly [15]. Our limited development resources essentially pegged time at a constant, so we had to reach a compromise between what we wanted and what we could achieve at each stage.

6. Design of Design Patterns Repository

![Figure 1: Screenshot of Learning Object “Place Holder” in DesignPatterns.ca](image)

There were six members in the project. The supervisor directed both the content and system development teams. Four researchers were responsible for authoring learning objects and one researcher focused on coding the repository. One researcher in the content authoring team also actively participated in the system design. So this researcher and the supervisor served as the bridge and interpreter to communicate between the users and developer. All the researchers have a weekly meeting to discuss their work and issues arising. The first 15 minutes is usually dedicated to discussing the repository. Users may talk about new needs, report bugs and suggest better ideas on operation and interface. Basically, we developed the system through adding or fixing functions according to users’ requests during the entire period of content authoring.

6.1. For Authors: a XML Template

In order to help authors to correctly produce “well-defined data” while they are writing learning objects, the development team suggested using XML as the template for authoring at the beginning of the project. Content authors drafted the XML template’s first version. Metadata and tags are defined based on the learning object (design patterns) syntax. The XML template is as follows:

```xml
<?xml version="1.0"?>
```
XSLT is used to display this XML file into readable webpage. Currently we provide one standard XSLT file for all learning objects. In the future, it is possible to customize the XSLT for certain learning objects to display differently.

There is not much syntax for writing a learning object, but we found that our authors still do not want to remember any grammar, syntax or hierarchy of file organization. While authoring a new learning object, they never started from scratch. They usually duplicated an existing learning object into a new folder to use it as a template. However, the files and text in the old learning object may not have everything ready for the new learning object (e.g. missing the image tags). We also continuously developed new syntax based on authors’ requests. Remembering all the syntax and rule is still an extra effort.

Therefore, we created a dummy learning object as a template containing the most recent package definition. The package contains the main control xml files, file hierarchy of images, animations and source files. In the control XML file, every possible tag (with sample text), rules and external file links are included. The author does not need to remember anything. S/he just has to download the package, delete unnecessary parts and replace files with their real content. To author a new learning object, s/he always can use the most updated package.

6.2. For Authors: Manual Coding Instead of WYSIWYG GUI

The core of learning object is the text file containing explanatory and external file links. It must be displayed as a webpage. To create such a document, a GUI interface with WYSIWYG would be welcomed by some novice users. But for experienced users, command line or scripts enable them to work in a more flexible and efficient way. Most users have their own favorite text editor, so we do not need to provide a web based GUI-interface or editor. After negotiating with content authors, we decided to use a simple GUI to create, upload and delete patterns in the system. The pattern itself will be organized and written in user’s own text editor. Files, images and texts will be created and packaged in their local editor. After all the content has been prepared, authors could upload and publish the content as a whole to the server through a drag + drop webpage. Minor changes can be made in the simple online editing window. While it is possible to edit directly online, our authors far preferred the faster offline process.

6.3. For Authors: Naming Convention and File Hierarchy

At the beginning of the project, we set several simple rules for the syntax of XML file and folder hierarchy. After a few weeks, the content authoring team found it essential to make the files organization explicit and meaningful. Then we formally wrote down naming convention rules and posted in the public space. These rules were also improved and edited during the following weeks.

To collect feedback and comments from external users, the repository was open to the public from the very beginning. With more and more colleagues and friends being invited in, the project team realized that they need a private place for themselves to store temporary files and internal documents. The system was thus separated to two parts. One is open for the public and the other is password protected. The naming convention document, the XML template and other development information was moved into are protected space. Only registered content authors were able to view them.

6.4. For Visitors: Animation to Present 3D Model Online

The requirement of presenting 3D models vividly online came up later after authors had created several samples in the learning objects. In this project, models were created parametrically – their 3D appearance changes when their parameters are adjusted. To help people understand the problem in a 2D screen, the model should at least be shown from different perspectives.
An “ideal” solution is allowing users to manipulate the model directly in a 3D environment online. However, it is hard to achieve in an inexpensive way. Moreover, being interactive is not the original purpose of a learning object (understanding the underlying parametric controls is more important).

Figure 2: Several frames of images to compose one animation

Although learners can download the source file and run it on their local machine, it will still interrupt the flow of reading. One researcher in the group suggested using animation to demonstrate such dynamic 3D models. In a group meeting, we discussed the suggestion and decided to use Macromedia Flash as the media to play animation in web browser. As people are not as familiar with creating animation in Flash as using MS Word, we came up with a solution to have the Flash file generate an animation by automatically picking up a series of images from a folder (Figure 2). Authors firstly capture the screenshots manually as frames of the animation. To make this step even simpler, another team member developed an auto-screen-capture module. Users just need to insert the module into their source file. After turning on the module, updates of the screen caused by the parametric model are captured and saved into a serial of images. To give better control of the animation, we asked the author to write a simple XML file (also can be modified in the template) to define the appearance of the animation (display sequence and time interval). The Flash (.swf) file automatically reads the XML file and images and displays them in the webpage as an animation.

The animation control XML file looks like:

```xml
<gallery timer="1.5" order="sequential" fadetime="0.5"
looping="yes" xpos="0" ypos="0" scale="0.5">
<img src="PlHo_Hedgehog_00000.jpg"/>
<img src="PlHo_Hedgehog_00001.jpg"/>
...
<img src="PlHo_Hedgehog_00020.jpg"/>
</gallery>
```

Later on, authors reported that they wanted multiple animations for one sample in a learning object. Therefore, in the naming convention part, we created a new rule for naming the Flash and XML file – the Flash file would only read the XML file with exactly the same name and pickup images listed in that XML file. Thus, the authors had more freedom to create animations efficiently and directly.

### 6.5. For Critics and Other Users

At the beginning, we mainly considered the needs of authors and visitors. With more colleagues visiting the system, we found that we should start to facilitate the critics. The first critiques came internally from the project team. Content authors read others’ work and left suggestions. Although we had face-to-face meetings, we still need to write done these suggestions when they occurred to us. Some external visitors also wanted to contribute new ideas or leave comments to these learning objects. We created a registration module and a discussion board. External users can register and discuss focus on their interested learning objects (or any part of the repository). In each of the learning objects, it is also possible to leave annotations directly.

### 6.6. Remove and Simplify Functions

Inherited from the FTP based repository, some features of this system were not welcomed by users. For example, in the FTP repository, files in the server are organized in the folder tree structure, and the navigation interface reflects the tree structure. Users are able to “go back to parent folder” or “back to root”. The development team kept the folder navigation tree in the interface at the beginning. But in a learning object repository, visitors do not need to know how files are organized from the server side and this actually confuses users. The development team received the complaint several times from group meetings. So we removed this folder navigation interface from this system. Although one developer still thinks it was a very useful feature, no end user felt inconvenienced by its removal. Having more functions does not necessarily make a system work
better. To reduce the users’ cognitive load, the system should provide the users exactly and only what they need.

7. Discussion

Although the current system was built to host one certain type of learning objects, we want to extend this project to handle more in the future. Currently we use a simple XML metadata to organize our learning objects. We can extend our XML definition to handle a wider range of specifications, standards and developments. Actually the next request from the research group is to fit our current XML metadata into IEEE Learning Objects Metadata (LOM). Due to the flexibility and extensibility of XML standard, this migration should be fairly easy with minimum impact to this research group and learning object authoring.

Considering that XML may still be too technical for some users, we kept some features from the ancestor (the FTP based repository). The user can create a folder in the system (same as making a new pattern) and drag + drop files into the folder. The system will pick these files up and display them as much as possible. Thus we made this repository more generic.

Such a system is only a tool for users who are busy with their own professional work. Working (especially sharing their own work) with the system is only a tiny part of their professional life. Their willingness to spend time with the tool is minimal. They do not have time to explore and learn feature of the system. This is also a lesson we learnt from managing the BCCampus SOLR repository. Instructors in British Columbia public institutions are encouraged to share their learning resources (such as webCT courses) with other instructors. If a contributor meets any problem during his/her first visit, he/she is unlikely to take a second try. Simplicity and directness is a key part of encouraging contributions from a wide audience.

8. Conclusion

Currently, DesignPattern.ca is neither a sophisticated learning object repository nor a complete content authoring tool. It combines both basic functions and aims to provide simple and sufficient support to its users/authors. Because of the timeline of the project, we did not have enough time to build the system before the content development stared. We chose the DSDM agile method and communicated with our users regularly and frequently during the entire process. The paper summarizes our experience of designing this learning object repository and authoring platform. Trying to create a simple and elegant system for all the users, we observed users, analyzed use cases and outlined essential needs of target users.

During the development, literature review of parametric modeling application, structures of design patterns and XML templates shaped our design. Until the moment of writing this paper, we continued to improve the system and some authors are writing new learning objects. Some users from Generative Components community have been invited to visit our repository. We continue to receive positive feedback and advices. An interesting fact of this project is that we learned much more from our users than we could contribute to them. This paper is dedicated to them.

9. References


