A Design Pattern Management Tool for Educational Purposes

Khalid S. Y. Al-Tahat, Ayat N. Al-ahmad,
Noor B. Kallajo, Wala' B. Al-Jayyousi

*Software Engineering Department, Information Technology Faculty,
Hashemite University, JORDAN*
ktahat@yahoo.com

Abstract

The benefits of using design patterns have been widely acknowledged by experienced software professionals. The application of design patterns to object-oriented software development has already brought positive results in practice. Reported benefits include significant increases in quality and productivity. However, using design patterns is not a trivial process especially for students and novice designers/programmers. This paper introduces a tool for teaching design patterns to students and novice designers/programmers in order to teach those recognizing appropriate patterns and applying them properly. It provides an interactive environment that is designed as if the users requested the information in multiple stages. Each stage adds additional features that were not included in the previous stage. The help supporting each stage guarantees that the user can understand the current stage and help him moving to the next stage. The tool provides a real life existence for each pattern selected that illustrates the application of that pattern in software development.

1. Introduction

The benefits of using design patterns have been widely acknowledged by experienced software professionals. Experienced designers/programmers reuse structures and designs that have worked well in the past without thinking [1]. However, it is not intuitive for novice programmers to recognizing appropriate patterns and applying them in designing robust systems [2]. Design patterns are not code; they must be implemented each time they are applied. Designers supply application-specific names for the key "participants"--classes and objects--in the pattern. Then they implement class declarations and definitions as the pattern prescribes. If this were all that was needed to implement a pattern, it would not be a big chore. But often there are many trade-offs in a pattern to consider, and different trade-offs often work synergistically, resulting in a proliferation of variant implementations--too many to support through conventional code reuse techniques. Developers are therefore likely to duplicate their efforts and those of other developers each time they apply a pattern.

Object-oriented design and modeling patterns is complex [3]. Whereas, until know design patterns are introduced as an essay for the designers to read, think about, and ultimately adapt what is learnt to his or her needs. In fact a designer may encounter situations where none of the sample implementations, offered in the design patterns document, are suitable to the designer's particular situation. Thus the development of a tool that aids in design patterns understanding and automations is badly needed. There are however no such tools. An important reason for this lack of computer support may be that design patterns are written in a rather informal way [3]. Computer support of the design process however, could well lead to further improvements on object oriented design.

In this work, we are developing a tool for design pattern management for educational and training purposes with a main objective of simplifying the learning curve for design patterns. This kind of tool does not replace the deliberations in choosing which pattern to use at any given time. It does not replace the need for designers to understand how and why they need to use a particular pattern. Nor do the tools replace the need for designer to understand balance of forces and tradeoff involved in choosing different solution variations. To be able to benefit from the tool, students must have the basic understanding of the software process and appreciate the importance of careful design. In addition, students must have some experience in object-oriented programming. We assume that students at this level have not established a consistent approach in performing object-oriented analysis and design. All patterns supported by the tool are taken from Gamma [5].

The composition of the paper is as follows. The second section introduces our tool focusing on its environment interactivity and the stages of exploring and learning design patterns. Concluding remarks are given in section three. The last section, section four, lists our references.

2. The Interactive Environment

As we mentioned earlier, our tool provides an interactive environment for the user. It came with a user friendly interface that was built with the guidance of the

famous "eight golden rules" [6] for designing user interfaces. This includes providing an easy and intuitive way to edit the design, displaying level of detail is adjustable, and in order to stimulate exploratory design, it is possible to undo multiple actions.
This interactive environment was designed as if the users requested the information in multiple stages. In this approach, the problem starts with simple requirements set and grow rapidly as the user realizes the potential of the system. This rapid growth of the requirements is one of the original motives in using design patterns to create robust systems [4]. In this section we shall explore the different stages of using our tool.

2.1. Stage 1: Selecting a Track

In the main screen, figure 1, the user can go in one of three different tracks, namely, 'Design Pattern Catalogue', 'Help', and 'New Project with Requirements'.

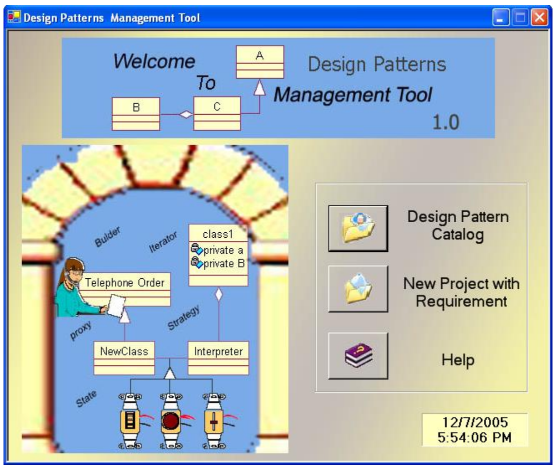


Figure 1. The main screen.

The 'Design Pattern Catalogue' choice is designed for expert design pattern user, figure 2. It allows them to select a particular design pattern and then drops its structure (as UML class diagram) onto a UML workspace without passing through the system wizards. The user may then modify the resulted design pattern class diagram and change in it according to the specifications of his problem. This is the traditional way of supporting design pattern in most of the tools that support design pattern.
The 'Help' in the main screen, explains everything about design patterns, the UML editor provided by the tool, and the components and functionalities of the tool. Providing a help on our tool is an essential because the design pattern topic is new, unfamiliar, and difficult to understand. Students and novice programmers/designers find it difficult to understand them. What are they? How do they work? And what are there benefits? At this stage the help is covering three main issues (see figure 3). The first issue is design patterns background where definitions, history, elements, examples of design pattern are given. The second issue covers the need and importance of having a design pattern management tool. Where the third issue covers what does the tool do and how to go around with it.

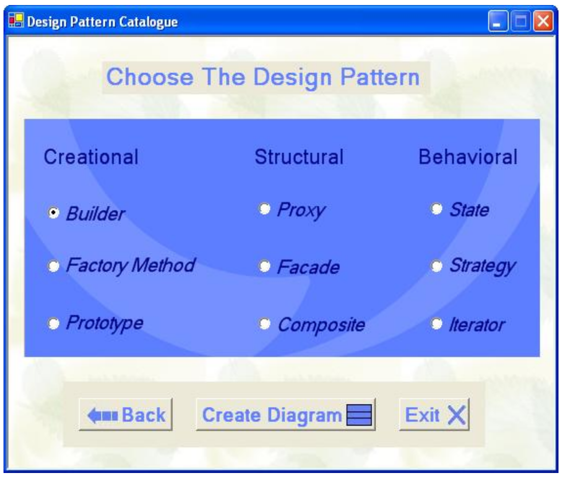
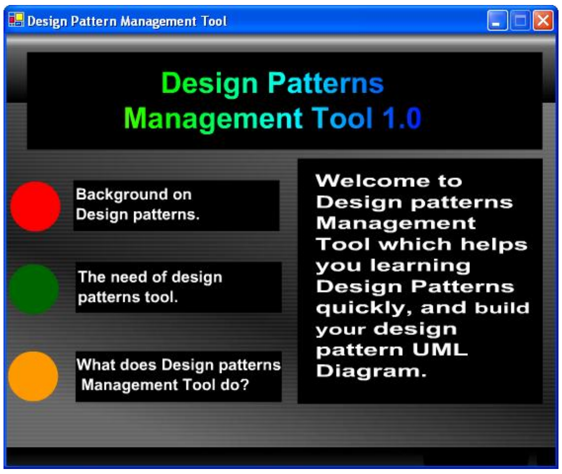


Figure 2. Design pattern catalogue.


Design Pattern Management Tool
Design Patterns Management Tool 1.0
Background on Design patterns.
The need of design patterns tool.
What does Design patterns Management Tool do?

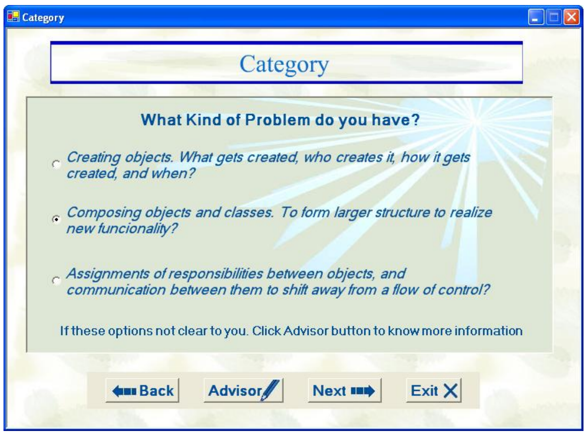
Welcome to Design patterns Management Tool which helps you learning Design Patterns quickly, and build your design pattern UML Diagram.

Figure 3. The main help screen.

Users start using the interactive environment provided by the tool, when selecting 'New Project with Requirements'. This choice will support the novice users to create a new project through a wizard system. In other words, the user will start his/her journey with the tool looking for the best solution (i.e. design patter) for his/her problem. This methodology provides level of automation which allow pattern to be customized to closely fit problem and context of its proposed use. As we mentioned earlier, this usage comes in few stages, each of which adds additional features that were not included in the previous stage. At any time during the his/her journey the user can call for a help which is provided in many forms (as we will see later), go to the previous "station", or "go home" by quitting. The rest of the section explores the stages of following this particular track.

2.2. Stage 2: Selecting the Design Pattern Category

In this stage the user is expected to select the main category for the problem he/she is facing. Figure 4 shows three main categories of problems supported by our tool: creating objects, composing objects and classes, and assigning responsibilities between objects. In case of having problem with understanding the categories presented and how to match his problem to one of them, the user may consult the ‘Advisor’ which would provides an intensive help to guide him/her through this stage. If the user continues to have problems at this stage, he has the chance to go back to the help provided by the previous screen and learn more about design patterns.



Category

What Kind of Problem do you have?

Creating objects. What gets created, who creates it, how it gets created, and when?

[✓ Composing objects and classes. To form larger structure to realize new funcionality?]

Assignments of responsibilities between objects, and communication between them to shift away from a flow of control?

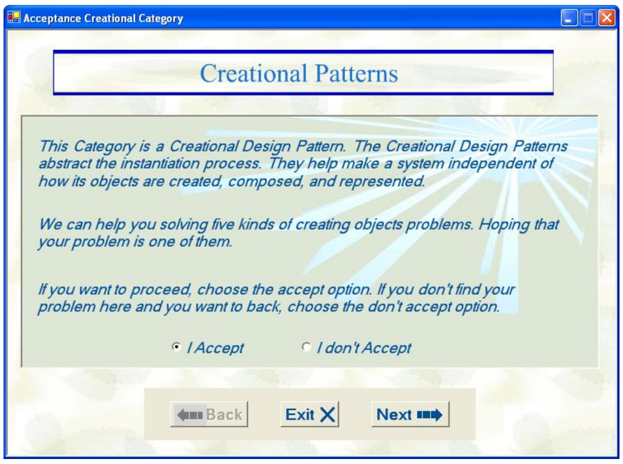
If these options not clear to you. Click Advisor button to know more information

Figure 4. Design patterns categories screen.

Otherwise, selecting ‘Next’ would take him to the second screen of this stage (see figure 5) where more details about the design pattern categories will be given. Here, the user needs to confirm that his/her problem falls in with the selected category. This would ensure that the user has selected the right category of design patterns, solutions, for his/her problem. Naturally, if the user’s problem does not fit to any of the categories supported by the tool, the user may go back to the previous screen or quit by clicking ‘Exit’. Quitting choice is made possible in every stage to support users’ internal locus of control.

2.3. Stage 3: Selecting a Particular Design Pattern

After ensuring that the user is on the right track, we come to the point where the user may select a particular pattern for the problem in hand. To help the user selecting the right pattern, the intent of each pattern in for each and every category is described by a meaningful line of text, see figure 6.



Creational Patterns

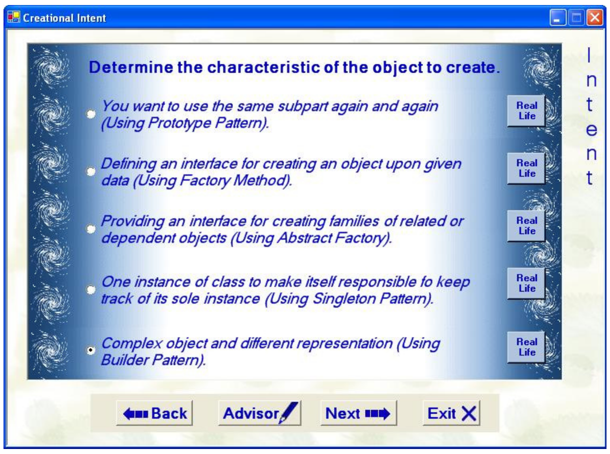
This Category is a Creational Design Pattern. The Creational Design Patterns abstract the instantiation process. They help make a system independent of how its objects are created, composed, and represented.

We can help you solving five kinds of creating objects problems. Hoping that your problem is one of them.

If you want to proceed, choose the accept option. If you don't find your problem here and you want to back, choose the don't accept option.

[✓ I Accept] I don't Accept

Figure 5. Confirmation screen for the design pattern category.



Determine the characteristic of the object to create.

You want to use the same subpart again and again (Using Prototype Pattern).

Defining an interface for creating an object upon given data (Using Factory Method).

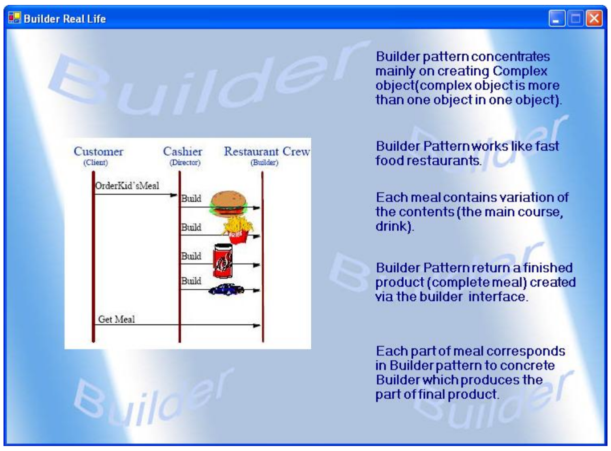
Providing an interface for creating families of related or dependent objects (Using Abstract Factory).

One instance of class to make itself responsible fo keep track of its sole instance (Using Singleton Pattern).

[✓ Complex object and different representation (Using Builder Pattern).]

Figure 6. Selecting design pattern screen.

Every word in this description is carefully selected to reflect, in depth, the intent of the pattern and all possible uses and solutions that the pattern may provide. Along with the intent of the pattern, the name of the pattern is also given. From this point on, the user would now what his candidate pattern is. As the intutivity is one important design goal of the tool, a real life existence of every design pattern were made available in the help to ease the understanding of the design pattern and relate it to some real life problem, see figure 7. The given example of using the design pattern in solving a real life problem is describes in points form and supported with some diagrams. This makes the example more attractive and its understanding easier. Comparing his problem with real life problem solved by the pattern. The user will, possibly, be able to have early judge of his selection. As for all other screen, the ‘Advisor’ is always available to give the help whenever it is needed. The user can go on with tool, go back to the previous screen, or even quit.


Builder Real Life

Builder pattern concentrates mainly on creating Complex object(complex object is more than one object in one object).

Builder Pattern works like fast food restaurants.

Each meal contains variation of the contents (the main course, drink).

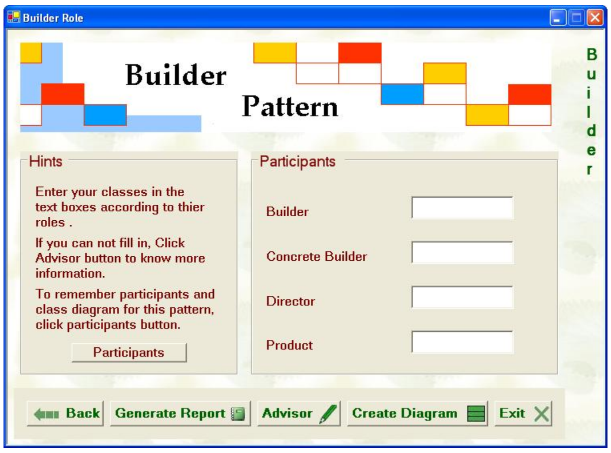
Builder Pattern return a finished product (complete meal) created via the builder interface.

Each part of meal corresponds in Builder pattern to concrete Builder which produces the part of final product.

Figure 7. Real life existence of the Builder design pattern

2.4. Stage 4: Specifying the Participating Classes

Now, the "right" design pattern has been selected. Here, we need to specify the participant classes of the pattern. A 'Role' screen was provided for specifying the participating classes, see figure 8. It is normally hard for the user, even the experts, to remember the class participating in the design pattern and their roles. 'Participants' button in this screen provides detail explanations of the pattern's participating classes and roles of each one of them. It also shows a UML class diagram for the design pattern which memorizes the user with the collaborations among the participation classes. If the user still faces any problem an advice can be requested from the 'Advisor'. After specifying the participating class, the user may get his pattern's class diagram dropped on a UML workspace by clicking 'Create Diagram'.


Builder Role

Builder Pattern

Hints

Enter your classes in the text boxes according to thier roles .

If you can not fill in, Click Advisor button to know more information.

To remember participants and class diagram for this pattern, click participants button.

Participants

Builder	
Concrete Builder	
Director	
Product	

Figure 9. The 'Role' screen for the Builder design pattern.

2.5. Stage 5: Creating the UML Class Diagram

We have built our own UML class diagram editor to support our tool. Seventeen class diagram components were provided to support all possible needs, see figure 9. This editor is supported with 'Drag and Drop' abilities and the use pop-up menus to perform the different operations over the design pattern class diagram. Using this editor, the user can drop a particular design patterns class and methods onto a UML workspace. It is flexible enough to allow the user modifying the resulted design pattern class diagram and change in it according to the specifications of his/her problem.

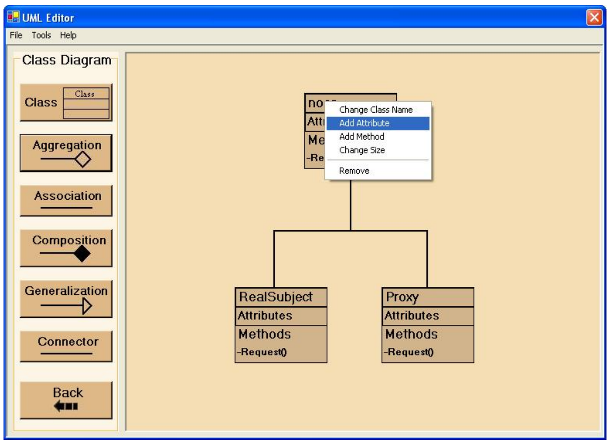


Figure 9. The UML class diagram editor.

3. Conclusions

The tool introduced in this work had aimed at breaking the ice of the learning and using design patterns outside the group of experts. We believe that this goal was achieved and the learning curve for design pattern was simplified through building set of interfaces which present full educational information and help user to take decisions. We have support our tool with significant strategy; design pattern with requirement which present design pattern's information progressively using a wizard system. According to [3] our tool comes in the second level of pattern automation, namely, the parameterized templates level.

We have selected UML [7] as the primary notation of our work, as it has well defined semantics, and has been accepted as a standard notation in the industry and academia. The implementation is done using Visual C#.NET language because is an event-driven, fully object-oriented programming language.

4. References

[1] B., K., Coplien, et al. ”Industrial Experience with Design Patterns”, *Proceedings of the 18th International Conference in Software Engineering*, March 1996, pp. 103-114.

[2] M., Clancy, and M., Linn, “Patterns and Pedagogy”, *Proceedings of the 30th ACM Technical Symposium on Computer Science Education*, March 1999, pp. 37-42.

[3] A., Bulka, “Design Patterns Automation”, *Proceedings of the 3rd Asia-Pacific Conference on Pattern Languages and Pprograms*, 2002, Vol. 13.

[4] M., Cline, “The Pros and Cons of Adopting and Applying Design Patterns in the Real World”, *Comunication of the ACM*, Vol. 99, No. 10, October 1996, pp. 47-49.

[5] Gamma et. al, “*Design Patterns: Elements of Reusable Object-Oriented Software*, Reading MA: Addison-Wesley, New York, 1995.

[6] Shneiderman, B., *Designing the User Interface*, Reading MA: Addison-Wesley, 3rd edition, 2005.

[7] Object Management Group 1998, *OMG Unified Modeling Language Specifications*, Framingham, MA 1998.