Design Pattern - a topic of the new mandatory subject informatics

Markus Schneider
Department of Informatics, Technische Universität München, 80290 München, Germany

markus.schneider@in.tum.de

Key words: Curriculum Changes, Didactics of Informatics, Informatics, Teacher Education

Abstract: Design patterns are commonly used in object-oriented modelling for the solution of frequently appearing design problems. Until recently their use was restricted to the domain of software engineering, However, object-oriented modelling is an important topic of the new mandatory subject informatics, which will be introduced during 2003 in all the four hundred Gymnasiums of Bavaria. In this paper we want to show that design pattern might add a valuable contribution to such a mandatory subject, which has its main focus on teaching the substantial concepts of informatics and on stimulating mental models to give real understanding of informatics.

1. INTRODUCTION

The designer of object-oriented software systems has to deal with recurring design problems such as modelling a recursive data structure, designing the menu of an application or modelling the interaction between the user interface and the core of the application. For such problems there are design patterns that provide a general solution. These solutions are visualized graphically using the Unified Modelling Language (UML) (Booch et al 1997).

The first systematic collection and classification of these design patterns was presented in 1995 and a year later another system of patterns was

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: 10.1007/978-0-387-35663-1_34

T. J. van Weert et al. (eds.), *Informatics and the Digital Society*

published that used a different system of classification (Gamma et al 1995; Buschmann et al 1996). Neither provided a formal definition of the concept "design pattern". However Gamma described the concept informally as:

"The design pattern ... are descriptions of communicating objects and classes that are customised to solve a general design problem in a particular context.";

and Buschmann as:

"A pattern addresses a recurring design problem that arises in specific design situations and presents a solution to it."

Until recently the use of design patterns was restricted to the domain of software engineering. Their use in informatics at school wasn't considered. Currently, there are efforts to introduce the basic concepts of object-oriented modelling in the proposed mandatory subject of informatics at the Bavarian Gymnasium (Hubwieser 2002). To this end, standard business applications are analysed and the structure of these applications is modelled by object-oriented means as early as the 6^{th} grade. The main idea is to familiarise the students with the basic structures of informatics that underlie an application and stimulate the development of cognitive structure to serve as mental models for real understanding. We wish to investigate whether the development of these cognitive structures could be supported by design patterns, since they represent graphically many of the basic concepts discussed in the proposed mandatory subject of informatics.

In the second part of this paper we show that the composite pattern is a frequently appearing design pattern in the projected curriculum of the 6^{th} grade. Furthermore, we illustrate the didactic function of this pattern. Then we point out the extent to which the observer pattern describes substantial aspects of interactive applications and we show the usefulness of utilising this pattern in school. Finally we discuss general criteria to select design patterns. These could be useful at school level. We also present many of the unresolved questions on the subject of design patterns.

2. THE COMPOSITE PATTERN AND ITS APPEARANCE IN THE SUBJECT OF INFORMATICS

In this section we will show that the composite pattern already appears in the curriculum of the 6th grade. However, as this pattern contains elements of object-oriented modelling, which is not discussed at this level, the composite pattern will not become a key topic of 6th grade study. Nevertheless, it is an important element of background information for the teacher and it is a structure well worth discussing in grades 9 - 13. The pattern provides a possible approach to establishing the basic concepts of informatics.

2.1 Curriculum areas related to the composite pattern

Objects in graphics

In the 6th grade, objects in vector graphical images are the first structures to be analysed and modelled.

For the first time the students have to familiarise themselves with the basic concepts of object-oriented modelling. In the discussion of these simple objects the composite pattern occurs. To illustrate this, let us analyse an image typically used in this phase of education (Hubwieser 1998).

The plan of, for example, an apartment is composed of graphics that represent the various rooms (Figure 1). The graphics of each room consist of graphics that represent the various furnishings. Illustrating these relations graphically a tree-like object diagram can be constructed (Figure 2). For reasons of clarity not all attributes have been specified. In a typical class diagram some attributes are declared (Figure 3).

In this example the composite pattern consists of three classes - Graphics, Geometrical Object and Graphical Object. Thereby, it is essential that every graphic is an aggregate of a certain number of graphical objects and, in particular, a certain number of other graphics. Instantiating this pattern, we get the tree-like object diagram shown in Figure 2.

Abstracting from this concrete example the composite pattern takes the form illustrated in Figure 4. It is worth mentioning that the superclass is built by the Component whereas the Compositum, which represents the system, is a lower class.

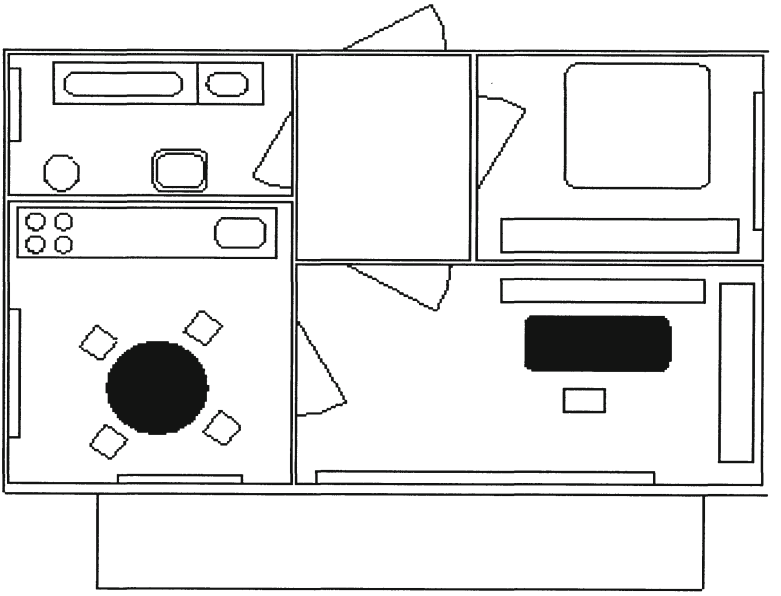

Figure 1. Plan of an apartment

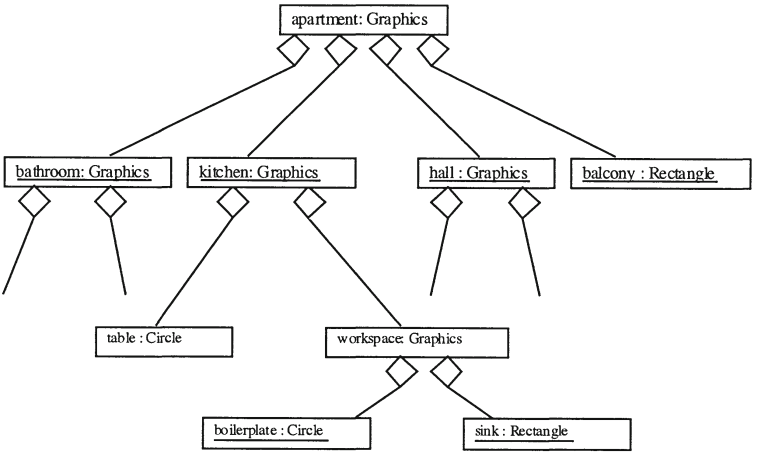

Figure 2. Object diagram of the apartment

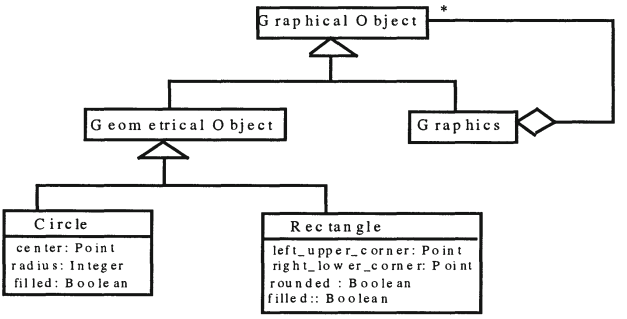

Figure 3. Class diagram of graphics

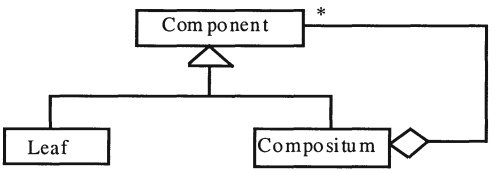

Figure 4. The composite pattern

Thus we have described the concrete scenario, the object diagram, the class diagram and finally the abstract pattern. In the following we will briefly outline the concrete scenario and then present the respective class diagram.

Relations between textual objects

In the 6th grade curriculum the discussion of graphical objects is followed by the object-oriented analysis of textual objects. We also have tree-like object structures for textual analysis. Consider the following scenario. A textual object consists of several sections. These sections are nested or divided into several main sections, e.g. this text is divided into - "Introduction", "The composite pattern" and so on. Then every main section is subdivided and this process of subdivision continues until a single character is reached.

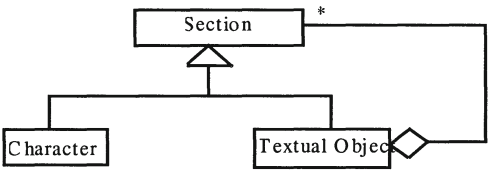

Figure 5. Composite pattern for textual objects

This description suggests the following model - every textual object consists of several sections and every section in turn is either a character or a textual object that aggregates a number of sections (Figure 5).

Hierarchical structures of files and directories

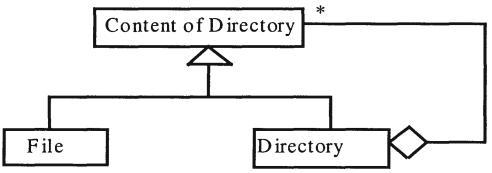

Figure 6. Composite pattern for directory structures

This topic is selected to illustrate the use of the composite pattern. A directory aggregates a certain number of objects. These represent the content of the directory. This content is either a file or a directory and we therefore get the pattern shown in Figure 6.

2.2 Some conclusions

The examples illustrate that the composite pattern is a substantial structure. It occurs at many points in the proposed curriculum for the 6^{th} grade. However, the complexity of this design pattern forbids its direct use as a 6^{th} grade topic as this would require a profound understanding of modelling techniques like inheritance.

Currently, in grades 9 -11, tuition is focused on the improvement of object-oriented modelling techniques established in the 6^{th} grade. In grades 12 and 13 recursive data structures and recursive function calls are discussed

(Hubwieser 2001). We consider that the composite pattern might offer an important contribution to the teaching of such content.

Improvement of the object-oriented modelling skills:

After the introduction of inheritance, the student should acquire an understanding of the interaction between aggregation and inheritance through exploration of practical examples. Obviously familiar examples from the 6th grade could be selected and appropriate class diagrams could be developed in the manner outlined above. This approach offers further advantage as the students would learn (in a very impressive manner) the difference between the instance of a class and the class itself. The tree-like object diagram emerges from the respective composite pattern by instantiation.

Propaedeutics of the topic recursion

Recursion is probably one of the most difficult topics to teach in informatics. Currently recursion is a topic in the 11-13 grades. One possible strategy might be to start with the composite pattern, then discuss recursive data structures and then introduce the recursive function call. This strategy reverses the common approach (Broy 1998) and was proposed by Hubwieser (2000). It has the advantage of starting from known structures - namely the tree-like object diagrams.

3. THE OBSERVER PATTERN

The composite pattern, illustrated in the previous section, is a design pattern that describes, for example, the structure of a text document. The observer pattern, in contrast, is a design pattern for modelling the interaction between certain objects, e.g. the interaction between the document, the user interface and other objects like the clipboard. Essentially, the observer pattern provides two services. On the one hand it decouples the user interface from the methods of the document and on the other hand it allows the design of user interfaces that provide (depending on the state of the document) the functionality that is currently applicable.

Prior to an illustration of the observer pattern the following points should be mentioned. The observer pattern is a pattern that describes substantial structures of interactive applications and therefore it is important for inclusion in 6th grade. Since the complexity of this pattern clearly exceeds that of the composite pattern, the following serves primarily as background information for the teacher of the 6th grade. Nevertheless, it could be used in grades 9-11 in the context of a detailed discussion of object-oriented modelling techniques.

3.1 Introductory example

Consider a typical part of an office application. The user interface contains the menu items "insert", "copy" and "exit". These menu items interact with the document and the clipboard in the following manner:

- The menu item "copy" is active only when a textual object is marked in the document. When performing a mouse click on this menu item the content will be copied into the clipboard.
- The menu item "insert" is active only if the clipboard is not empty. This menu item allows the insertion of the contents of the clipboard into the document.
- When activating the menu item "exit" the content of all open documents should be saved. The objects of the clipboard may be saved for later use.

Now, we can model the relationships between the objects "Document", "Clipboard", "Copy", "Insert" and "Exit". The elements of the user interface should provide event-handling methods. These should use appropriate methods of the document or clipboard. In this way the user interface can be separated from the functionality of the document or the clipboard.

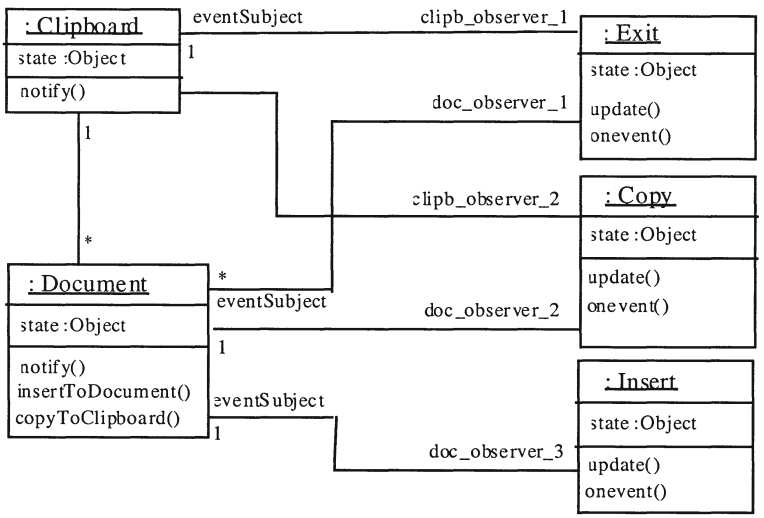

Figure 7. Object diagram of a subsystem of an office application

Now look at the object diagram (Figure 7). It describes the objects and the relationships between them. On the right-hand side the object diagram

contains several observers. These elements of the user interface observe the clipboard and the documents. The clipboard and documents are called observables. Depending on the state of the observed object the observers take on a certain state, for example, the object "Copy" is active when a part of the document is marked. The updating of an observer results from the notify method - as soon as the state of an observable changes this method will be called. Then the notify method calls the update methods of all associated observers.

3.2 Abstraction

Abstraction from the above example results in the observer pattern described in Figure 8. It is significant that every call of a method of the observable results in a call of the method notify. This method in turn causes all observers to update themselves.

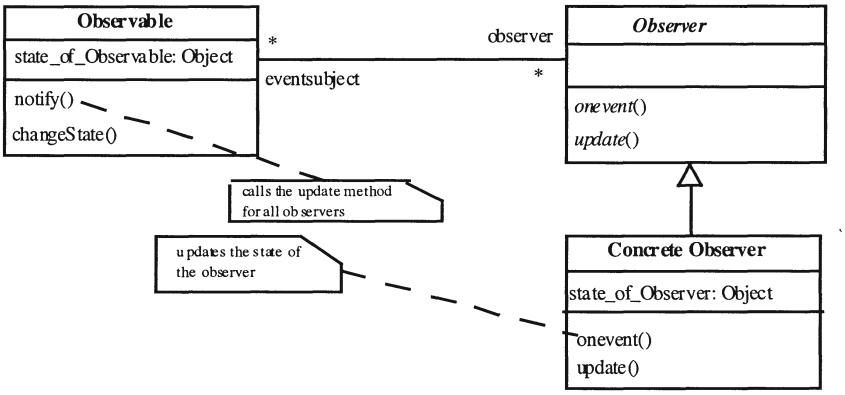

Figure 8. The observer pattern

In the class diagram (Figure 8) we introduced an abstract superclass. For an introductory understanding of the observer pattern it is not necessary to do so, since one might associate the observables with the concrete observers directly. However, for more profound understanding the use of an abstract class is helpful. It reinforces the fact that the event handling method and the update method are properties of all observers.

The observer pattern is a pattern that describes the interaction between objects of the user interface and the application. Therefore, the dynamic behaviour of the involved objects is of particular interest. The sequence diagram (Figure 9) shows the behaviour of the objects "Document", "Clipboard" and "Copy".

First a certain text area is marked in the document. This modification of the documents state activates the notify method. This in turn calls the update method of the observer object "Copy". Now, if the object "Copy" registers an event, the event-handling method of this object calls the method of the document and inserts the marked text area into the clipboard. The clipboard in turn notifies all observers by calling the respective update method.

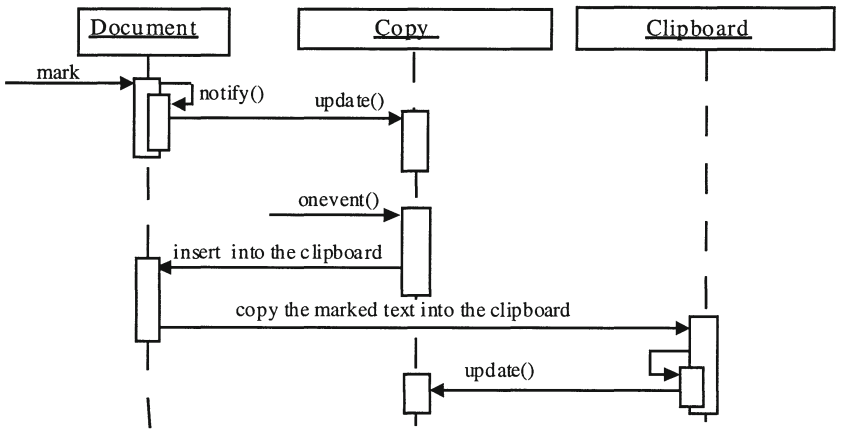

Figure 9. Scenario with observers and observables

3.3 More interesting subsystems of interactive applications with observer pattern features

In the following, we briefly outline how the observer pattern occurs in some subsystems of interactive applications:

- A document has been created. The menu items "Save" and "Save as" remain inactive as long as the user has made the first modifications;
- A document can be closed only when the content of the document has been saved.
- When opening a document the "Ok" button is not activated until a file name has been selected or declared.
- The form of the mouse button varies according to the position.

3.4 More conclusions

The observer pattern is complex and it cannot be used in pure form in the grades 6, 9 and 10. However, in a simplified form, for example without the use of abstract classes or in the form of an object diagram, this pattern might be discussed.

As is apparent from the above illustrations the modelling techniques that are used cause no difficulties. It is more likely that the methods used in the observer pattern could be a problem since substantial features of these methods cannot be modelled graphically.

Separation of the user interface and the core of the application

With regard to object-oriented modelling (proposed for the mandatory subject informatics at the Bavarian Gymnasium) it is essential that the student should be aware that the user interface is simply an interface to the methods of the application and is not the application itself. Once aware of this separation students will be able to understand the meaning of object-oriented modelling and can develop the cognitive structures which lead to a profound understanding of informatics. This concept of separation is described by the observer pattern and this pattern is therefore an ideal tool for stimulating the development of the above mentioned cognitive structures.

4. ARE THERE MORE PATTERNS WITH RELEVANCE FOR THE SUBJECT OF INFORMATICS?

For the composite and observer pattern the relationship with topics of the projected subject informatics is clear. The question arises, which other design patterns might be also of relevance for this subject? There are candidates like the interpreter pattern in the context of formal languages or the state pattern to model the state-dependent behaviour of an object. The more substantial question in this context is - Do we have general criteria to select the patterns whose use might be of value to the subject of informatics?

The following criteria are suggested:

- The pattern has to represent a substantial concept of informatics.
- The pattern has to be of general interest. Special patterns, which are utilised in some specific context only, are normally not suited.
- The pattern has to provide real support to the teaching of the represented concept and therefore provide a didactical function.
- The pattern should not be too complex such that the students would not understand it easily or fully.

- The understanding of the respective pattern should not require too much background information about a specific programming language.

These criteria could be classified into two categories. On the one hand we have a contextual category (criteria 1 and 2) and on the other hand we have a didactical category (criteria 3, 4 and 5). For our purposes a systematic analysis of design pattern has to clarify the following questions:

- How can the design patterns be classified using the categories - contextual, didactical?
- What are the consequences of this classification with respect to the use of the patterns in the subject of informatics?
- How can these patterns be integrated into the subject of informatics?

4.1 Known classification schemes

Both Gamma and Buschmann proposed a system of patterns. These systems are not ideal for the purposes described above.

Gamma et al divided the patterns into structural, behavioural and creational patterns. Creational patterns are patterns which systematise the instantiation process. Since this definitely involves questions on the level of a specific programming language, creational patterns might not be suitable for use in schools.

The other classification proposed by Buschmann et al divides the pattern into architectural pattern, design pattern and idioms. Idioms deal with the details of the implementation and therefore can be excluded for our purposes. The criteria for the classification of the architectural and design patterns are again too coarse and unspecific to estimate their use globally.

5. CONCLUSION

In summary we conclude that the two known classification systems are helpful in pre-selecting appropriate patterns. However, for tuition we need systematic analysis that classifies the patterns in the manner outlined above. The application of such a classification could clarify the questions and issues such as which design patterns ought to be used at school, which basic concepts of informatics are represented by the selected patterns and in which manner will these patterns be introduced in the mandatory subject of informatics?

REFERENCES

Booch, G., Rumbaugh, J. and Jacobson I. (1997) *Unified modelling Language; Semantics and Notation Guide 1.0*; Rational Rose Software Corp., San Jose, California.

Broy, M. (1998) Informatik. *Eine grundlegende Einführung*, Auflage, Springer.

Buschmann, F., Meunier, R., Rohnert, H., Sommerlad, P. and Stal M. (1996) *A system of pattern*. Wiley and Sons.

Gamma, E., Helm, R., Johnson, R. and Vlissides J. (1995) *Design patterns; Elements of Reusable object-oriented Software*. Addison Wesley.

Hubwieser P. (1998) *Didaktik der Informatik*. Springer.

Hubwieser P. (2000) *Rekursion im diaktischen Querschnitt*. Habilitationsvortrag.

Hubwieser P. (2001) Objektorientierte Modellierung von Standardsoftware im Anfangsunterricht Informatik. *INFOS 2001*.

Hubwieser P. (2002) Mental Models enhancing user skills. Unpublished paper.