KATHOLIEKE UNIVERSITEIT
LEUVEN

Faculty of Business and Economics

Software patterns to improve knowledge transfer: an experiment

L. De Rore, M. Snoeck, G. Poels and G. Dedene

DEPARTMENT OF DECISION SCIENCES AND INFORMATION MANAGEMENT (KBI)

KBI 0903

Software Patterns to Improve Knowledge Transfer: an Experiment

Lotte De Rore[1], Monique Snoeck[1], Geert Poels[2], and Guido Dedene[1,3]

[1]Katholieke Universiteit Leuven, Department of Decision Sciences & Information Management, Naamsestraat 69, B-3000 Leuven (Belgium), {lotte.derore;monique.snoeck;guido.dedene}@econ.kuleuven.be
[2]Universiteit Gent , Faculty of Economic and Applied Economic Sciences, Hoveniersberg 4, 9000 Gent (Belgium) , geert.poels@ugent.be
[3]Universiteit van Amsterdam, Amsterdam Business School, Information Management, Roetersstraat 11, 1018 WB Amsterdam (The Netherlands)

Abstract

Patterns for software development have been a hot topic for some time within the object-oriented community. Patterns are part of a software engineering problem-solving discipline. It all started with Design Patterns [11], but gradually patterns were used in a larger number of areas of system development. The goal of patterns within the software community is to create a body of literature to help software developers resolve recurring problems encountered throughout all areas of software development. Patterns help to create a shared language for communicating insight and experience about these problems and their solutions [4].

In this research report, first, a definition of software patterns is given, including some history, an overview of the different kinds of software patterns, the elements of a pattern and the different pattern formats. Secondly, as patterns claim to improve transfer of knowledge, we performed an experiment to test this hypothesis. This experiment is described in Section 2. Finally, Section 3 formulates the conclusions about this experiment.

Contents

1 Software Patterns

1.1 Definition and History

Patterns were conceived by an architect, Christopher Alexander, who used them to document recurring problems and corresponding solutions in architectural design and urban planning [13]. In his book 'A pattern language' [3], the concept of patterns is defined as follows:

> "Each pattern describes a problem which occurs over and over again in our environment, and then describes the core of the solution to that problem, in such a way that you can use this solution a million times over without ever doing it the same way twice." [3]

For Alexander, patterns are not an end in themselves, they are only a means to an end, namely to generate the *quality without a name* (QWAN) [2]. This quality is the feeling of satisfaction and pleasure you get from some great buildings. Usually these are old buildings and they are timeless. It is difficult to explain why these buildings have this quality and more recent buildings don't have this anymore. Alexander wrote down in patterns [3] how modern architects and builders can recapture the qualities inherent in these great buildings. The patterns in 'A pattern language' [3] describe his ideas about how to design cities, towns, neighborhoods, houses, rooms etc. in such a way that they have this QWAN. In order to be able to build again in a timeless way, we first need to know what this quality without a name is; next we need to construct a pattern language as a gate to this QWAN and finally the way to timeless building is to use these patterns [2]. More recently Alexander changed this QWAN into the name *Wholeness* [1].

As in the software engineering communities, several people have searched for an answer to the problem of reinventing the wheel over and over again, the work of Alexander inspired them. W. Cunningham and K. Beck, the pioneers of patterns in software development, tried to adapt Alexander's ideas to software development [7]. The concept of Software patterns was introduced at large in the object-oriented community at the OOPSLA '94 conference [17].

Besides the definition of Alexander, a more general definition of pattern is given by Riehle and Zullighoven:

> "A pattern is the abstraction from a concrete form which keeps recurring in specific non-arbitrary contexts" [16]

Both definitions are clear: software patterns represent problems as well as solutions, and as such they are a way to analyze solutions to recurring problems, to make those solutions reusable and to communicate them. They are much more than just a solution. A traditional misconception about software patterns is to view them as *generic* solutions which should be mapped against problems for which they might be potential solutions. No, they also deliver a context (when and where can the pattern be used), the forces that are involved in applying the patterns (such as, the trade-offs between the alternatives, the misfits, the goals and the constraints) and a resolution of the forces (how and why the solution balances the forces) [4].

1.2 Types of Patterns

Design patterns, used to document solutions to recurring problems in software design, are probably the most well known kind of software patterns. Although, patterns were first adopted in the object oriented community, design patterns should definitely not be restricted to this domain. The authors of 'Patterns of Software Architecture' [7] classify their design patterns on the range of scale and the level of abstraction. As such, they distinguish three categories: architectural patterns, design patterns, and idioms.

> "An *architectural pattern* expresses a fundamental structure organization schema for software systems. It provides a set of predefined subsystems, specifies their responsibilities, and includes rules and guidelines for organizing the relationships between them." [7]

Architectural patterns can be used at the beginning of high-level design to specify the fundamental structure of an application. The Model-View-Controller [7] is an example of an architectural pattern. It addresses the problem of user interfaces that are prone to change requests. To solve this problem, the MVC pattern divides the interactive application into three components: the model which contains the core functionality and data; the views which display information to the user and controllers which handle the users input.

> "A *design pattern* provides a scheme for refining the subsystems or components of a software system, or the relationships between them. It describes a commonly-recurring structure of communicating components that solves a general design problem within a particular context [11]." [7]

A design pattern is applicable at the end of the high-level design stage or during the detailed design stage. They are smaller in scale than the architectural patterns, but they are still language and implementation independent. For example, the PUBLISHER-SUBSCRIBER design pattern [7] solves the problem of keeping the state of cooperating components synchronized by defining one publisher who notifies any number of subscribers about changes to its state.

> "An *idiom* is a low-level pattern specific to a programming language. An idiom describes how to implement particular aspects of components or the relationships between them using the features of the given language." [7]

Idioms or coding patterns are the lowest level patterns and can be used in the implementation phase of a project. They describe how to solve an implementation specific problem in a programming language. Most coding patterns will be language-specific. Sometimes a design pattern can provide a source of idioms when the pattern is considered from the perspective of a specific programming language.

However, design patterns are not the only patterns in software engineering. *Analysis patterns* can be used to document recurring and reusable analysis models [9] [12]. *Organizational patterns* give solutions about how to structure your organization or projects [8]. *Process patterns* document solutions for software process design problems. And besides all these pattern types, one can find *domain specific patterns* for each possible domain.

1.3 Elements of a Pattern

Patterns are all about communicating knowledge. In order to make patterns easier to understand and apply, some essential information is necessary. These elements, the *mandatory elements*, are: the context, the problem, the forces and the solution. Additionally, a good name for the pattern is required. However, sometimes these mandatory elements will not be enough to contain all the essential information of the pattern. The *optional elements* of a pattern form give pattern writers considerable flexibility in which additional information they present and how they structure it to maximize the readers' understandability. Figure 1 lists the different elements. [4] [16]

1.3.1 Mandatory Elements

Pattern Name The pattern name is a word or short phrase that describes the essence of the pattern. Good names enhance communication, especially

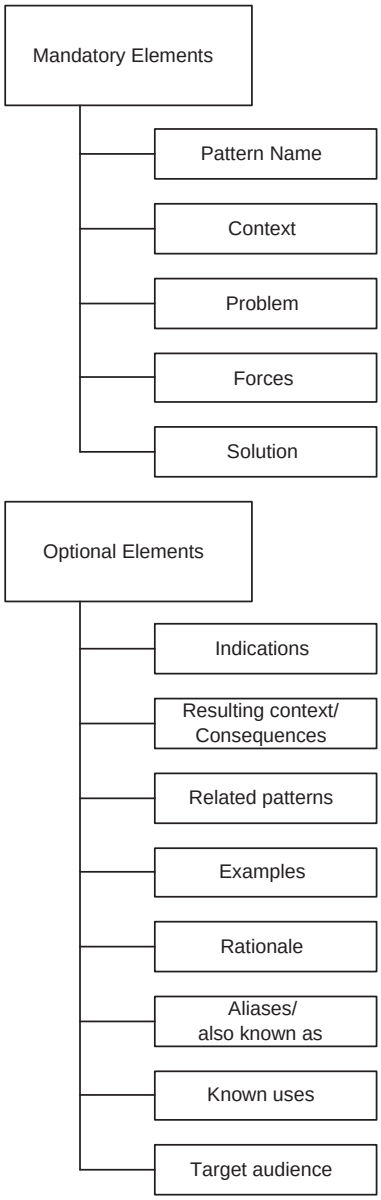

Figure 1: Elements of a pattern

when you can guess the intent from the name. A good name allows one to use a single word or short phrase to refer to the pattern and the knowledge and structure it describes. A good pattern name will become part of the vocabulary for discussions.

Context The context is the preconditions under which the problem and its solution seem to recur, and for which the solution is desirable. This tells the pattern's applicability as the circumstances in which the problem is being solved will impose constraints on the solution. It can be thought of as the initial configuration of the system before the pattern is applied to it.

The context is often described via a "situation" rather than stated explicitly. Sometimes, the context is described in terms of the patterns that have already been applied. The relative importance of the forces (those that need to be optimized at the expense of others) is determined by the context.

Problem This section describes the specific problem that needs to be solved, namely the intent of the pattern: the goals and objectives it wants to reach within the given context and forces. Often the forces oppose these objectives as well as each other.

Forces The forces section gives a description of the relevant forces and constraints and how they interact/conflict with one another and with the goals one wishes to achieve. Forces describe why the problem is hard and why the obvious solution(s) would not work. Forces describe what makes the problem a problem: an absence of tension between the forces suggests an absence of a problem. Forces are often contradictory considerations that must be taken into account when choosing a solution to a problem. The relative importance of the forces (those that need to be optimized at the expense of others) is implied by the context.

Solution In this section, the proposed method of solving the problem is described. Not only a solution description is given, but also some instructions or guidelines to keep in mind when constructing or implementing this solution. Sometimes more than one solution is possible for a given problem and the 'goodness' of a solution to a problem is affected by the context in which the problem occurs. Each solution takes certain forces into account. It resolves some forces at the expense of others. It may even totally ignore some forces. The most appropriate solution to a problem is the one that best resolves the important forces as determined by the particular context.

1.3.2 Optional Elements

Indications In this section, the symptoms that might indicate that the problem exists can be described.

Resulting Context/ Consequences The resulting context describes what happens when the solution is applied, including the consequences (the potential benefits and liabilities) and other problems that may arise from the new context. It describes the post-conditions and side-effects of the pattern. This includes a description of which forces have been resolved, which ones remain unsolved and which patterns may now be applicable.

Documenting the resulting context produced by one pattern can help to correlate the pattern with the initial context of other patterns, such as the next pattern(s) in a pattern language.

Related Patterns A pattern is almost never isolated. In this section, other patterns that may be of interest to the reader can be described: predecessor patterns whose application lead to this pattern; successor patterns whose application follows from this pattern (e.g. patterns that solve some of the problems in the resulting context); alternative patterns that describe a different solution to the same problem; more general or (possibly domain) specific variations of the pattern.

Examples Patterns can be very abstract; adding a section with concrete examples illustrates the application of the pattern and helps the reader understand the pattern's use and applicability. The sample applications of the pattern illustrate a specific initial context, how the pattern is applied and transforms that context and the resulting context. Easy-to-comprehend examples from known systems are usually preferred. An example may be supplemented by a code sample to show one way how to implement the pattern.

Rationale An explanation of why this solution is most appropriate for the stated problem within this context. It is a justifying explanation of steps or rules in the pattern, and also of the pattern as a whole in terms of how and why it resolves its forces in a particular way to be in alignment with desired goals, principles and philosophies. It tells us how the pattern actually works, why it works, and why it is 'good'.

Aliases/ Also Known As In this section, other names by which this pattern might be known are given.

Known Uses This section describes known occurrences of the pattern and its application within existing systems. This can serve as a validation of the pattern by verifying that it is indeed a proven solution to a recurring problem. One should have at least 3 known uses of independent instances of 'real world' successes.

Target Audience This section identifies a primary target audience with whom you would like to communicate the solution. One should keep this audience in mind while writing the pattern. It may be useful to explicitly describe the target audience in the pattern introduction. This helps set the expectations of the reader by telling them 'up front' that they are (not) the intended audience. It also helps people to determine which meaning of an ambiguous term you had intended.

1.4 The Pattern Form

Patterns are all about communicating and sharing knowledge. Therefore, it is important that the knowledge in the patterns is easily transferable to the reader. Consistent structures where the reader can easily find the information needed will definitely help. The *pattern form* is the written template and style through which a pattern is presented. Nevertheless, there is no single universal pattern form. Pattern forms can be free and narrative, almost prose, or tightly structured with many standard sections. The main purpose of a pattern form is to give structure to the pattern [19]. Despite the differences in presentation, each pattern form should contain the essential parts of a pattern, namely name, context, problem statement and solution. In addition, these essential parts should be clearly recognizable upon reading the pattern. Therefore, a pattern format mainly helps the writer to be sure all the information needed is in his/her pattern.

In the following, three pattern forms will be described: the Alexandrian form, the GOF format and the Coplien form. As stated before, there are different kinds of patterns. Each has another kind of format that is most appropriate to be used to communicate their content. For example, including code in a narrative form can be very difficult while, in the format with a lot of sections, there is a risk that the essential concepts get lost and scattered across several sections. With the choice of a pattern form, one should keep in mind the target audience one wants to address and the kind of pattern being written. Structure is important as it is the strength of patterns, however, the structure should not restrict the writer.

PATTERN NAME *
Introduction giving context
**
Problem description
Essence of problem (1-2 sentences)
Body of problem
Solution - stated in form of an instruction
Diagram giving solution graphically
Examples of use
**
Resulting context

Table 1: Alexandrian Form

1.4.1 Alexandrian Form

The *Alexandrian form*, as used in 'A pattern language' by Alexander [3], is a very narrative form with relatively few headings, see Table 1. As a result, it is easy to follow for the reader. This pattern form is convenient for non-technical patterns, for example: organizational patterns, pedagogical patterns, business patterns. For technical patterns, however, it will be difficult to write down the pattern in this form. Sometimes, the writer gives an indication of the confidence he/she has in the pattern by a star next to the pattern name. This format is often seen as the form for more experienced authors. For a first time pattern writer, this is not the most appropriate form to begin with.

1.4.2 GOF Format

The *GOF format*, as used in the Gang of Four pattern book [11], is a very structured form, breaking up the pattern into many headings, see Table 2. This format is very useful for very detailed and technical patterns. The solution section is broken up in several parts, namely: structure, participants and collaborations. For a first time writer, this is not the most appropriate format as the risk exists to be choked by the structure.

1.4.3 Coplien Form

The last format we describe, the *Coplien form*, is called that way because it is most identified with its creator, Jim Coplien [10]. The form is a good starting point for a first time author. The key elements are headed sections for problem, context, forces and solution, see Table 3. Most authors will add

Pattern name
Intent
Also known as
Motivation
Applicability
Structure
Participants
Collaborations
Consequences
Implementation
Sample code and usage
Known uses
Related patterns

Table 2: GOF Format

Pattern name
Problem
Context
Forces
Solution
Resulting context
Rationale

Table 3: Coplien Form

a few extra sections when needed. Each section consists a few paragraphs, with the forces section commonly a list of bullet points.

1.5 Pattern Languages

Individual patterns are useful, but to address real sized problems, *pattern languages* are essential. A pattern is almost never isolated but connected with other patterns. A collection of patterns forms a vocabulary for understanding and communicating ideas. A whole set of patterns can be related with each other and solve together a more complex problem. This is what Alexander calls a *pattern language* [3]. If a pattern is a recurring solution to a problem in a context given by some forces, then a pattern language is a collective of such solutions which, at every level of scale, work together to resolve a complex problem into an orderly solution according to a predefined goal.

1.6 Anti-Patterns

If a pattern represents a 'best practice', then an *anti-pattern* represents a 'lesson learned'. An anti-pattern is not just a 'bad' solution, but presents a solution that 'sounds' good but doesn't work [17]. An anti-pattern can be very useful simply because knowing what doesn't work (and why) can be very useful. There are two notions of anti-patterns: on the one side those that describe a bad solution to a problem which resulted in a bad situation and on the other side those that describe how to get out of a bad situation and how to proceed from there to a good solution. These latter ones are the most useful anti-patterns [4].

2 Patterns Experiment

2.1 Purpose of the Experiment

Patterns are a way to transfer and communicate knowledge. By describing a good and proven solution for a problem in a specific context, the user of the pattern can reuse this good solution and as such improve the quality of the own work. With this experiment we want to test the hypothesis that knowledge can be transferred more easily through the use of patterns. The experiment, conducted with the collaboration of students, consists of a modeling exercise and a pattern that can be used in this exercise. The knowledge in the pattern is also course material of the students and can therefore be considered as already known by the students. Can a pattern be helpful to improve modeling capabilities? Is it easier to transfer knowledge that is provided in the course material through patterns?

2.2 Experiment 1

2.2.1 Participants

The first experiment is performed in collaboration with students of the course 'Ontwikkeling van Bedrijfstoepassingen' in March 2008. The experiment took place as part of an exercise session of the class. The group consisted of approximately 60 people. Not everyone had the same background: the students from the faculty of business and economics (FBE) had no prior modeling knowledge and the students from the computer science faculty had some modeling experience.

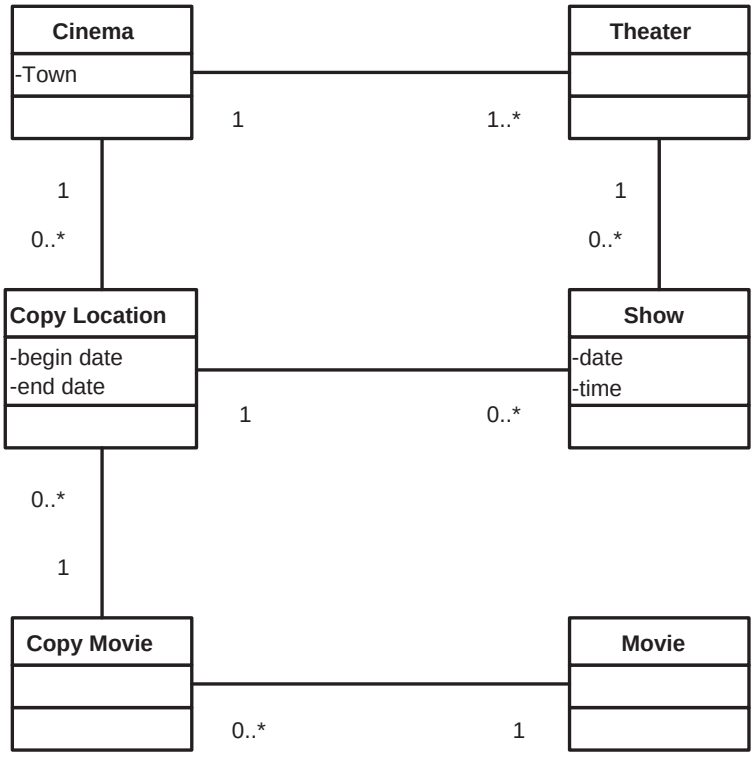

Figure 2: Kinepolis Case

2.2.2 Set-up of Experiment 1

The experiment was build around the Kinepolis case. To create a model for this case, the Association Object pattern of L.L. Boyd [14] can be used. This pattern explains how a class can be created to record history and information about an association between two classes.

> **Kinepolis case:** Kinepolis owns several cinemas in different towns. Each cinema has several theaters in which several shows can be programmed per day. Every cinema needs its own copy of the movie, and possibly more than one if a movie is to be played simultaneously in more than one theater. A copy will only be shown in the theaters of the cinema where the copy is located. Consequently, the programming should be planned in advance in order to timely move copies of movies between cinemas as required by the programming. (Figure 2)

Due to the different background and prior knowledge of the participants, the students should not be mixed into one group. However, by splitting up

the students, the groups become too small to further divide them into a test group (with pattern) and a control group (without pattern).

An alternative would be to let the participants solve the case twice, once without and once with the pattern. However, this is not a viable option as there will be learning effects: the fact that the students already solved the case without the pattern will influence their second solution and the changes might not be due to the help of the pattern or the pattern might have less effect than expected. Therefore, we opted to create a second case: the Kidney case. To have comparable results, this second case should be completely similar to the Kinepolis case both in terms of the ideal class model as in the way the requirements are formulated.

> **Kidney case:** The hospital UZ Kidney, specialized in haemodialysis, has several geographically spread campuses. Each kidney patient is assigned to the personally best located campus. Each campus has an own haemodialysis center for its patients. The treatment of a patient consists of repeated use of haemodialysis machines and invoicing is based on the use of the machines. There are several types of machines and considering the high cost of such machines, the hospital purchased only a couple of machines of each type. Consequently, the treatments should be planned in advance and the machines have to move between the campuses according to the demand. Logically, a patient can only use a machine when it is on the right campus. (Figure 3)

Although the two cases are completely similar, the difficulty degree of the case and the domain knowledge of the participants might still influence the results. Therefore, two versions of the experiment were build with a different order of the cases. One part of the participants solved the Kinepolis case without pattern and then the Kidney case with pattern, while the other part solved the Kidney case without pattern and then the Kinepolis case with pattern. As such, we remove any case-dependent improvement.

As an observation, we also included a third element in the experiment. After the second case, the participants did not only read the pattern, but are expected to have used it. Therefore, we let them look back at their first case to evaluate whether they now could improve upon their solution. The set-up of the experiment is as follows:

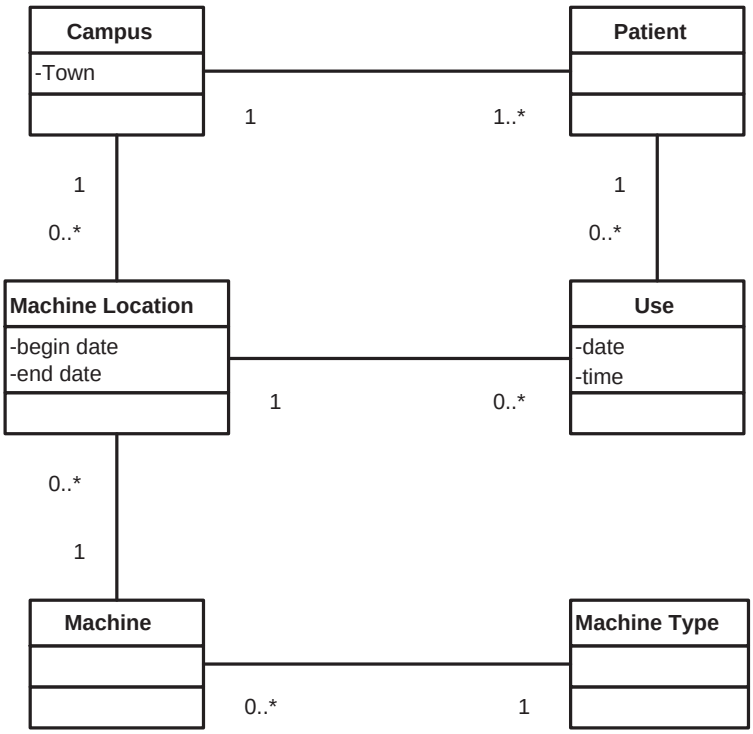

Figure 3: Kidney Case

Version 1:

Question 1: Kinepolis case without pattern
Question 2: Kidney case with pattern
Question 3: Kinepolis case with pattern

Version 2:

Question 1: Kidney case without pattern
Question 2: Kinepolis case with pattern
Question 3: Kidney case with pattern

2.2.3 Evaluation of Experiment 1

In the Kinepolis case, the ASSOCIATION OBJECT pattern can be used to create the class *copy location* of the association between *cinema* and *copy film* on the one hand and the class *show* of the association between *theater* and *copy* on the other hand. As such, one can keep track on which location a copy is and which movies are planned in which theaters. Similar, with the Kidney case, the pattern can be used to create the class *machine location* of the association between *campus* and *machine* on the one hand and the class *use* of the association between *patient* and *machine*.

To evaluate the solution of a participant, we score both associations: 0 when no association is found, 1 when an association is identified, 2 when this is identified as a many to many association and 3 when they used the pattern and made a class of the association. The fact that the association between *show* and *copy movie* should be replaced by an association *show* and *copy location* as a show can only happen when the copy of the movie is available in the right cinema and similarly, the association between *use* and *machine* should be replaced by an association between *use* and *machine location* as a patient can only use a machine that is located at the right campus, is not scored as this is not part of applying the pattern. The scoring sheets for both the Kinepolis case and the Kidney case are shown in Table 4 and Table 5.

We expect the participants to model the classes *show* and *use* even without knowing the pattern as these classes are explicitly mentioned in the problem description.

2.2.4 Results of Experiment 1

In total, 35 students participated in the experiment. Of them, 13 students are from the computer science faculty, while the other 22 students were from the faculty of business and economics (16 Business Engineer - Management Informatics and 6 Applied Economic). The students of FBE had, except for this course, no prior knowledge about modeling.

Between *cinema* and *copy movie*:	
no association	0 points
association	1 point
* - * association	2 points
class *copy location*	3 points
Between *theater* and *copy movie*:	
no association	0 points
association	1 point
* - * association	2 points
class *show*	3 points

Table 4: Scoring Kinepolis Case

Between *campus* and *machine*:	
no association	0 points
association	1 point
* - * association	2 points
class *machine location*	3 points
Between *patient* and *machine*:	
no association	0 points
association	1 point
* - * association	2 points
class *use*	3 points

Table 5: Scoring Kidney Case

Table 6 shows the results for the informatics students. For each association where the pattern can be used, a score is given according the scoring sheets in Table 4 and Table 5. Some students created a ternary association. This means, according to them, there can only be an association between two of the three classes when there is an association between the three classes (e.g. no *copy* can be assigned to a *cinema* unless it is also assigned to a *theater*). This is, of course, an incorrect assumption. Therefore, students who created a ternary association between the three classes receive 0,5 points and students who created a ternary class receive 1,5 points. Depending on the version of the experiment, question 1 and question 3 is the Kinepolis and question 2 the Kidney case for the first version. For the second version, question 1 and 3 is the Kidney case and question 2 the Kinepolis case.

When a score of 2 is given, this indicates the student identifies a many to many association, which is an association that changes over time, but the student does not make a class of it to keep track of history. In other words, the student sees the context but does not apply the pattern ASSOCIATION OBJECT.

To see whether the pattern improves the solution of the cases, the results of question 1 (without the pattern) and question 2 (with the pattern) for both cases are compared. This means, for the Kinepolis case, we have to compare the results of question 1 of version 1 with the results of question 2 of version 2 and for the Kidney case, the results of question 1 version 2 and the results of question 2 version 1.

For the Kinepolis case, we can see an improvement in the sense that no students have modeled a many to many association without making a class of it. Also for the Kidney case, there is only one many to many association that is not transformed into a class. So, we can conclude that the pattern has helped to model keeping track of information of an association that changes over time. However, we noticed also some missing associations. Some students do not model (anymore) the association between e.g. *cinema* and *copy*. It is possible that the set up of the experiment causes this. By providing a pattern, students focus too much on the application of this pattern and once they found one application, they are satisfied with their solution and as such lose some creativity. They do not contemplate anymore whether their solution is complete or whether they could apply the pattern twice in their model.

When we compare question 1 with question 3 for each participant, we investigate the improvement on an individual basis after knowing and applying the pattern. These improvements (subtracting the score of question 3 with the score of question 1) are shown in the last two columns of Table 6. For most associations, there is an improvement. For the associations

student	version 1 2	question 1 copy loc. mach. loc.	 show use	question 2 mach. loc. copy loc.	 use show	question 3 copy loc. mach. loc.	 show use	improvement copy loc. mach. loc.	 show use
1	1	2	1	3	2	3	1	1	0
2	1	2	2	3	0	3	2	1	0
3	1	0	2	3	0	0	3	0	1
5	1	1	0	0	1	1	0	0	0
9	1	2	3	1	3	3	3	1	0
11	1	0.5	3	1.5	1.5	1.5	1.5	1	-1.5
12	1	2	3	3	0	3	0	1	-3
14	1	2	3	1.5	1.5	0	3	-2	0
6	2	1	3	0	3	3	3	2	0
7	2	1	3	0	3	1	3	0	0
8	2	1	2	0	3	3	3	2	1
10	2	1	3	3	3	3	3	2	0
13	2	1	2	3	0	3	0	2	-2

Table 6: Results Experiment Informatics Students

where there is no improvement (negative score), we see that the main reason is that the participant deleted the association completely from the model which corresponds with the missing associations discussed above.

We also kept track of the attributes participants added to their model, this can be seen as a quality improvement of the solution. After the pattern was introduced, more participants added attributes to their models.

For the students from FBE, it was not easy to evaluate their solutions. They made many mistakes, even basic modeling mistakes. It was not possible to evaluate the solutions according our predefined scores. Therefore, we excluded the results of this group completely.

2.2.5 Conclusions of Experiment 1

For the experienced modelers, the pattern had a positive influence on their model. Their models improved by (1) adding time (i.e. many to many association instead of an association in one point of time), (2) keeping information of the changes over time (i.e. using the ASSOCIATION OBJECT pattern) and (3) inserting more relevant attributes. We can conclude that the pattern not only helped to model the association object, but it made the participants also contemplate the forces. As such, more participants noticed that for example the association between *copy* and *cinema* changes over time and the importance of adding this information to the model. However, the pattern might also influence the creativity of the modeler. As a consequence, the modeler rather thinks about applying the pattern instead of concentrating on the case and all the requirements.

For the non-experienced modelers, too many mistakes were made with respect to basic modeling. From this we can conclude that without the basic modeling capabilities, the pattern will not help the participants.

2.3 Experiment 2

2.3.1 Participants

The second experiment is performed with students of the course 'Architecture and Modeling of Management Information Systems' (AMIS) in November 2008. The group consists of approximately 50 people. The experiment took place as part of an exercise session during class.

2.3.2 Set-up of Experiment 2

As noticed in the first experiment, people that do not have enough modeling experience or knowledge could distort our results. Therefore, we included

a pre-test in the experiment. With this test we test whether the students are familiar enough with ER formalism to understand an ER-model and its cardinalities [15]. Students that score less than 8 on this pre-test are excluded from the experiment.

The AMIS students learn to model with Merode [18], this implies they model a class diagram using only associations that express existent dependency between different object types. As a consequence, these students will not model a many to many association but immediately create an extra association class dependent of the main classes that participate in the association, meaning that they have learned to automatically apply the ASSOCIATION OBJECT pattern. Hence, the cases and the pattern used in experiment 1 are not useful for these students, as we can assume them to immediately model the correct solution. The ASSOCIATION OBJECT pattern will therefore not lead to an improvement of the solution for these students.

For this reason, two new cases are created: the Sunrise case and the SAI case. These two cases are similar in terms of the ideal class model as in the way the requirements are formulated. The pattern that can be used in both cases is the INSTANTIABLE RESOURCE sub-pattern. This is part from the QUANTIFY THE RESOURCE pattern of the pattern language for business resource management [5, 6]. The students are provided with the QUANTIFY THE RESOURCE pattern. This means, they need to figure out themselves which of the four sub-patterns can be used in the cases.

> **Sunrise case:** Travel agency Sunrise sells trips. Every trip comes with a description of the destination and the available accommodation. Every trip has several departure dates. The price of a trip will depend on the departure date and the chosen duration. On request of the customer, the travel agency makes a proposal depending on the destination, dates and the number of travelers. The proposal will be based on the price information given in the description of the trip, and will include discounts based on the characteristics of the booking (number and age of travelers). (Figure 4)

> **SAI case:** SAI (Studiecentrum Automatische Informatieverwerking) is a Belgian Association of computer users and computer specialists. Each year, members have to pay their membership fee and then they can attend courses. From time to time, courses are announced and people can start to register. The courses are very successful, but can only accept a limited number of participants. Therefore, the same course can be organized several times

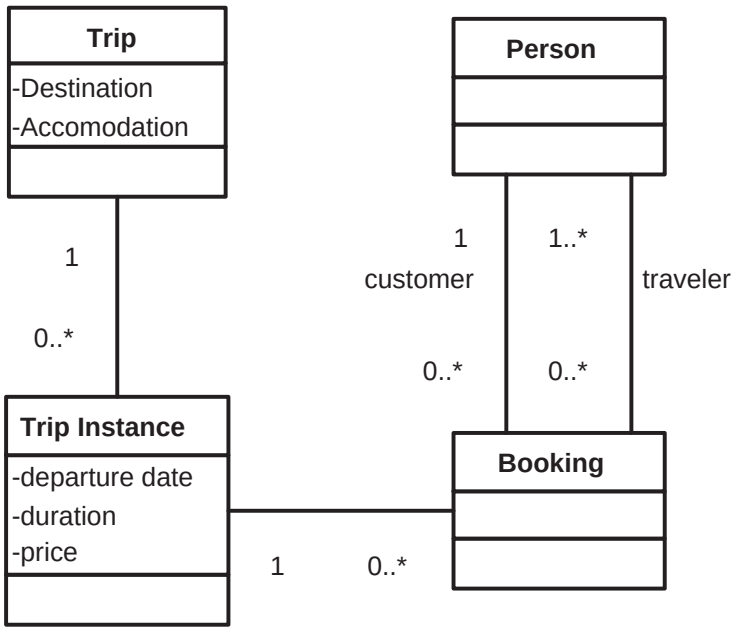

Figure 4: Sunrise Case

and people need to choose which course they register for. (Figure 5)

The experiment has the same set-up as experiment 1. There are two versions of the experiment depending on the order of the cases. The first case needs to be solved without pattern. Before solving the second case, the students are provided with the pattern. In a last question, the students can reconsider the first case and make corrections if needed.

Version 1:

Question 1: pre-test
Question 2: Sunrise case without pattern
Question 3: SAI case with pattern
Question 4: Sunrise case with pattern

Version 2:

Question 1: pre-test
Question 2: SAI case without pattern
Question 3: Sunrise case with pattern
Question 4: SAI case with pattern

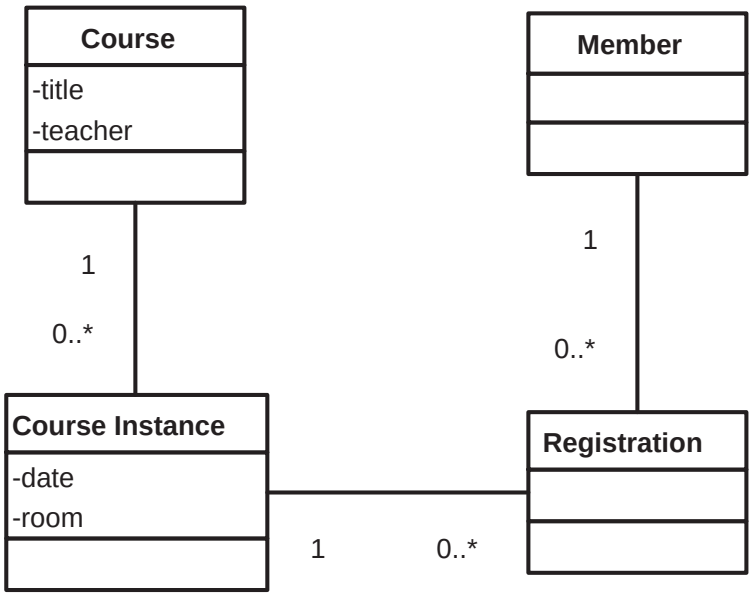

Figure 5: SAI Case

2.3.3 Evaluation of Experiment 2

In the Sunrise case, the INSTANTIABLE RESOURCE pattern can be used to make several instantiations of a *trip* according to the departure date and duration. Similar with the SAI case, the pattern can be used to make several instantiations of the *course* according to the time the course is organized. As such, information that is mutual for the trip or course can be kept on the level of trip/course, while information that is specific for that instance, such as departure date or time can be kept on the level of *trip instance* or *course instance*.

To evaluate the solution of the participants, we use the scoring sheets in Table 7 and Table 8. Students that mix up the concept of a trip/course and the instantiation of a trip/course (for example: model a class *dates* who records all possible departure dates for all possible trips) get 0 points. When they use the SINGLE RESOURCE sub-pattern [5, 6] and model each trip instance and course instance as a separate instance of trip and course respectively, they get 1 point. When they use INSTANTIABLE RESOURCE, they get 2 points. And when they notice they need to link a booking or a registration with the instance object and not with the *resource type* object, they get 3 points.

concept of trip and trip instance mixed up	0 points
only class *trip*	1 point
INSTANTIABLE RESOURCE applied	2 points
link with *trip instance* in stead of *trip*	3 points

Table 7: Scoring Sunrise Case

concept of course and course instance mixed up	0 points
only class *course*	1 point
INSTANTIABLE RESOURCE applied	2 points
link with *course instance* in stead of *course*	3 points

Table 8: Scoring SAI Case

2.3.4 Results of Experiment 2

In total 42 students participated in the experiment. Of these, 10 students did not pass the pre-test and were excluded from the results. The results of the 32 participants who scored 8 or more on the pre-test are shown in Table 9.

At first sight, the improvement column (difference between Q4 and Q2), indicates for some students an improvement (positive number), for some a status quo (improvement=0) and only one worsening (negative number, student 52). The reason for the large amount of status quo (improvement=0) is that a lot of students already came up with a good model (score 3) in Q2, without the pattern. Obviously, their solution could not be improved by using the pattern.

T-tests are performed to test the significance of the improvements we notice. The first test is to check whether the availability of the pattern has an influence on the score of each of the two cases. Therefore, we compare the results for question 2 and question 3 for both cases separately.

Sunrise case: The students of version 1 solved the Sunrise case without pattern (in question 2) and the students of version 2 solved the Sunrise case with pattern (in question 3). These are the results we have to compare with each other. The null hypothesis is that the mean of the scores with and without pattern are equal, while the alternative hypothesis is that the mean of the scores with pattern (version 2) is higher than the mean of the scores without pattern (version 1).

$$H_0 : \mu_{Sunrise-1} = \mu_{Sunrise-2}$$

$$H_a : \mu_{Sunrise-1} < \mu_{Sunrise-2}$$

student	version	Q1	Q2	Q3	Q4	improvement
1	1	10	3	3	3	0
2	1	10	0	3	3	3
3	1	8	0	3	3	3
4	1	10	0	2	3	3
5	1	9	0	0	1	1
6	1	8	1	1	1	0
11	1	10	0	3	0	0
14	1	9	3	3	3	0
15	1	10	1	3	3	2
16	1	10	0	3	0	0
17	1	9	0	3	0	0
18	1	10	3	3	3	0
19	1	8	0	3	0	0
20	1	10	0	3	3	3
21	1	10	0	3	3	3
31	2	10	3	3	3	0
32	2	9	0	3	3	3
33	2	9	3	0	3	0
34	2	10	1	3	3	2
35	2	10	1	3	3	2
37	2	8	1	3	1	0
38	2	10	3	3	3	0
39	2	9	3	3	3	0
40	2	10	3	1	3	0
41	2	9	3	3	3	0
42	2	10	3	3	3	0
44	2	8	1	1	3	2
45	2	9	3	0	3	0
46	2	10	1	1	1	0
48	2	10	3	3	3	0
49	2	10	3	3	3	0
51	2	10	3	3	3	0
52	2	8	3	2	2	-1

Table 9: Results Experiment 2

Test of difference>= 0 versus one-tailed alternative	
Hypothesized mean difference	0,000
Sample mean difference	-1,544
Pooled standard deviation	1,171
Std error of difference	0,410
Degrees of freedom	31
t-test statistic	-3,771
p-value	$< 0,001$
Test of equality of variances	
Ratio of sample variances	1,176
p-value	0,371

Table 10: Two Sample Analysis for Sunrise-1 - Sunrise-2

The results of the two-sample one-tailed t-test are shown in Table 10. We can reject the null hypothesis of equality of means at the 1% significance level. Hence, the results with pattern are significantly better than the results without pattern.

SAI case: The students of version 2 solved the SAI case without pattern (in question 2) and the students of version 1 solved the SAI case with pattern (in question 3). These are the results we have to compare with each other. The null hypothesis is that the mean of the scores with and without pattern are equal, while the alternative hypothesis is that the mean of the scores with pattern (version 1) is higher than the mean of the scores without pattern (version 2).

$$H_0 : \mu_{SAI-1} = \mu_{SAI-2}$$

$$H_a : \mu_{SAI-1} > \mu_{SAI-2}$$

The results of the two-sample one-tailed t-test are shown in Table 11. We can not reject the null hypothesis of equality of means at the 10% significance level. This result was expected as most of the students solved the SAI case with a score of 3 even without the pattern.

Additionally, we also test whether there is an improvement on individual student basis. Therefore, we compare the score of Q2 with the score of Q4 or the improvement column.

$$H_0 : \mu_{Q4-Q2} = 0$$

$$H_a : \mu_{Q4-Q2} > 0$$

Test of difference<= 0 versus one-tailed alternative	
Hypothesized mean difference	0,000
Sample mean difference	0,322
Pooled standard deviation	1,003
Std error of difference	0,351
Degrees of freedom	31
t-test statistic	0,919
p-value	0,183
Test of equality of variances	
Ratio of sample variances	1,392
p-value	0,269

Table 11: Two Sample Analysis for SAI-1 - SAI-2

paired test of difference <= 0 versus one-tailed alternative	
Hypothesized mean difference	0,000
Sample mean difference	-0.788
Pooled standard deviation	1.2688
Std error of difference	0.2209
Degrees of freedom	32
t-test statistic	-3.57
p-value	0,0006

Table 12: Paired t-test for Q2-Q4

This can be performed with a paired t-test. The results are shown in Table 12. We can reject the null hypothesis at the 1% significance level. This means, we measure a significant improvement per student between question 2 (without pattern) and question 4 (with pattern).

2.4 Experiment 3

2.4.1 Participants

The third experiment is performed with students of the course 'Beleidsinformatica' also in November 2008. The group consists of approximately 180 people and are students from 3th Bachelor Applied Economics, 3th Bachelor Commercial Engineer or from MBE (master in business and economics). The experiment took place as a test quiz.

2.4.2 Set-up of Experiment 3

The set-up of the experiment is completely similar to the set-up of experiment 1. However, also for this experiment, we included the ER formalism knowledge pre-test as in experiment 2.

Version 1:

- Question 1: pre-test
- Question 2: Kinepolis case without pattern
- Question 3: Kidney case with pattern
- Question 4: Kinepolis case with pattern

Version 2:

- Question 1: pre-test
- Question 2: Kidney case without pattern
- Question 3: Kinepolis case with pattern
- Question 4: Kidney case with pattern

2.4.3 Evaluation of Experiment 3

The scoring of the Kinepolis and Kidney case are a little adjusted compared to the first experiment. More students modeled an association as a class. As they see in that case that there is information that is dependent on the association and not on the two main classes, these solutions should score better than an association or a many to many association. Therefore, to evaluate the solution of a participant, both association are scored as follows: 0 when no association is found, 1 when an association is identified, 2 when this is identified as a many to many association, 3 when an association as a class is modeled of the association, 4 when they used the pattern and made a class of the association and 5 when they added time attributes to that association class. The new scoring sheets for both the Kinepolis case and the Kidney case are shown in Table 13 and Table 14.

2.4.4 Results of Experiment 3

182 students participated in the experiment. 84 students were excluded as they scored less than 8 on the pre-test. The results of the 98 students are shown in Table 15. For each association where the pattern can be used, a score is given according the scoring sheets in Table 13 and Table 14. Some students created a ternary association. Similar as in experiment 1, students who created a ternary association between the three classes receive 0,5 points and students who created a ternary class receive 2,5 points. Depending on the version of the experiment, question 2 and question 4 are the Kinepolis

Between *cinema* and *copy movie*:	
no association	0 points
association	1 point
* - * association	2 points
association as a class	3 points
class *copy location*	4 points
time attributes	5 points
Between *theater* and *copy movie*:	
no association	0 points
association	1 point
* - * association	2 points
association as a class	3 points
class *show*	4 points
time attributes	5 points

Table 13: Scoring Kinepolis Case

Between *campus* and *machine*:	
no association	0 points
association	1 point
* - * association	2 points
association as a class	3 points
class *machine location*	4 points
time attributes	5 points
Between *patient* and *machine*:	
no association	0 points
association	1 point
* - * association	2 points
association as a class	3 points
class *use*	4 points
time attributes	5 points

Table 14: Scoring Kidney Case

case and question 3 the Kidney case for the first version. For the second version, question 2 and question 4 are the Kidney case and question 3 the Kinepolis case.

Table 15: Results Experiment 3

student	version 1 2	question 2 copy loc. mach.loc.	 show use	question 3 mach.loc. copy loc.	 use show	question 4 copy loc. mach.loc.	 show use
101	1	1	1	5	3	4	1
102	1	1	4	5	2	1	4
103	1	3	3	3	3	3	3
104	1	1	5	5	2	2	5
108	1	0	4	5	0	0	5
109	1	2	5	1	3	2	5
110	1	1	0	1	0	4	0
112	1	2	1	5	1	5	1
113	1	1	5	2,5	2,5	0	5
114	1	0	3	1	5	0	5
116	1	1	1	2	3	1	1
117	1	1	5	3	3	3	5
121	1	2	3	1	5	2	5
123	1	1	3	5	0	1	3
124	1	0	1	5	1	0	5
126	1	2	4	2	5	1	5
127	1	1	1	5	2	5	5
128	1	2	1	5	0	0	5
129	1	1	3	5	5	1	5
130	1	1	5	2,5	2,5	1	5
131	1	1	5	5	2	1	4
135	1	0	4	1	4	0	4
136	1	1	1	5	5	5	5
137	1	1	1	0	3	5	0
138	1	1	1	5	0	5	0
141	1	1	1	5	0	5	1
142	1	0	2	5	0	0	5
143	1	1	5	1	5	1	2
144	1	1	5	5	0	5	0
147	1	1	1	5	0	0	5
148	1	0	1	1	1	0	4

Table 15: (Continued)

student	version 1 2	question 2 copy loc. mach.loc.	 show use	question 3 mach.loc. copy loc.	 use show	question 4 copy loc. mach.loc.	 show use
150	1	3	3	3	0	3	3
151	1	0	5	1	0	0	5
153	1	1	0	2,5	2,5	5	0
154	1	1	1	1	1	1	1
155	1	2	3	3	4	3	4
156	1	3	4	5	1	5	4
157	1	2	1	1	5	2	5
159	1	0	1	1	0	0	1
162	1	0	0	0,5	0,5	0	0
166	1	0	1	5	0	0	5
168	1	3	3	3	0	3	3
169	1	1	3	5	3	5	3
170	1	0	4	2,5	2,5	0	5
171	1	0	0	2	0	0	0
172	1	0	4	4	0	5	4
173	1	0	4	5	3	0	5
175	1	3	0	4	0	5	0
176	1	0	5	5	0	0	0
178	1	1	4	5	3	5	4
179	1	1	0	5	0	5	0
180	1	0	5	5	5	5	5
181	1	0	2	5	0	0	5
182	1	0	1	1	2	0	1
183	1	1	0	5	0	1	5
185	1	1	1	0	0	1	1
187	1	1	0	1	5	0	5
193	1	0	5	0	5	0	5
202	2	1	0	0	5	1	5
203	2	1	3	5	4	4	3
206	2	2	1	0	5	2,5	2,5
208	2	1	3	0	5	5	3
210	2	2	4	0	5	4	0
211	2	2	4	0	3	2	5
216	2	3	3	0	5	3	3

Table 15: (Continued)

student	version 1 2	question 2 copy loc. mach.loc.	 show use	question 3 mach.loc. copy loc.	 use show	question 4 copy loc. mach.loc.	 show use
217	2	2	1	2,5	2,5	2,5	2,5
221	2	3	3	1	5	1	5
223	2	0,5	0,5	2	5	5	0
225	2	0	3	0	5	0	5
226	2	1	0	0	0	5	0
227	2	2	1	1	5	2	0
228	2	1	3	0	5	1	5
232	2	2	3	0	5	0	5
233	2	1	0	5	0	5	0
234	2	1	2	0	5	5	2
235	2	2	0	0	5	2	0
236	2	1	2	0	5	2,5	2,5
240	2	3	0	0	5	5	0
242	2	1	1	0	4	1	4
248	2	0	4	1	0	2	4
253	2	2	2	0	5	2	2
254	2	1	2	0	0	4	2
257	2	1	2	5	5	1	2
258	2	1	0	0	5	5	0
261	2	1	2	0	5	1	5
265	2	0	0	0	5	5	0
269	2	0,5	0,5	1	5	2,5	2,5
276	2	0	4	0	5	0	5
278	2	1	3	0	5	0	5
279	2	1	0	0	5	1	1
285	2	1	3	5	5	5	3
286	2	2	2	2	5	5	5
287	2	2	0	1	5	5	0
288	2	1	2	0	5	5	5
289	2	0	0	5	0	5	5
291	2	2	0	4	0	2	0
292	2	2	2	0	5	5	2
295	2	1	1	0	0	5	1
298	2	1	1	0	0	1	1

Test of difference>= 0 versus one-tailed alternative		
Hypothesized mean difference	0,000	
Sample mean difference	-0,022	
Pooled standard deviation	1,309	NA
Std error of difference	0,267	0,295
Degrees of freedom	97	55
t-test statistic	-0,083	-0,075
p-value	0,467	0,470
Test of equality of variances		
Ratio of sample variances	3,731	
p-value	$< 0,001$	

Table 16: Two Sample Analysis for Copy Location-1 - Copy Location-2

T-tests are performed to test whether there is a significant difference in the scores with or without pattern. We first test the difference on association level for each of the cases. Therefore, we compare the results of question 2 and question 3 for each of the associations.

Kinepolis case: The students of version 1 solved the Kinepolis case without pattern (in question 2) and the students of version 2 solved the Kinepolis case with pattern (in question 3). The null hypothesis is that the mean of the scores for association *copy location* with and without pattern are equal, while the alternative hypothesis is that the mean of the scores with pattern (version 2) is higher than the mean of the scores without pattern (version 1).

$$H_0 : \mu_{copylocation-1} = \mu_{copylocation-2}$$

$$H_a : \mu_{copylocation-1} < \mu_{copylocation-2}$$

The results of the two-sample one-tailed t-test are shown in Table 16. The null hypothesis of equality of variances can be rejected on 10% significance level, hence we need to look at the results in the last column for the t-test. We can not reject the null hypothesis, there is no significant difference between the means of question 2 and question 3 for the *copy location* association. Indeed, Table 15 shows that both in question 2 and 3, a lot of students score badly (score 0 or 1) on the *copy location* association.

A similar null hypothesis and alternative hypothesis can be expressed for the *show* association in the Kinepolis case.

$$H_0 : \mu_{show-1} = \mu_{show-2}$$

$$H_a : \mu_{show-1} < \mu_{show-2}$$

Test of difference>= 0 versus one-tailed alternative	
Hypothesized mean difference	0,000
Sample mean difference	-1,366
Pooled standard deviation	1,894
Std error of difference	0,386
Degrees of freedom	97
t-test statistic	-3,534
p-value	$< 0,001$
Test of equality of variances	
Ratio of sample variances	1,206
p-value	0,255

Table 17: Two Sample Analysis for Show-1 - Show-2

The results of the two-sample one-tailed t-test are shown in Table 17. The null hypothesis of no difference in the means between question 2 (without pattern) and question 3 (with pattern) can be rejected at 1% significance level. Hence, the results with pattern are significantly better than the results without pattern.

Kidney case: The students of version 2 solved the Kidney case without pattern (in question 2) and the students of version 1 solved the Kidney case with pattern (in question 3). These are the results we have to compare with each other for both associations (*machine location* and *use*). The null hypotheses are that the mean of the scores with and without pattern for each of the associations are equal, while the alternative hypotheses are that the mean of the scores with pattern (version 1) is higher than the mean of the scores without pattern (version 2).

$$H_0 : \mu_{machinelocation-1} = \mu_{machinelocation-2}$$

$$H_a : \mu_{machinelocation-1} > \mu_{machinelocation-2}$$

$$H_0 : \mu_{use-1} = \mu_{use-2}$$

$$H_a : \mu_{use-1} > \mu_{use-2}$$

The results of the two-sample one-tailed t-test are shown in Table 18 for *machine location* and in Table 19 for *use*. For *machine location*, the null hypothesis can be rejected at 1% significance level. Hence, the results with the pattern are significantly better for this association than the results without pattern. For *use* however, the null hypothesis can not be rejected at

Test of difference<= 0 versus one-tailed alternative		
Hypothesized mean difference	0,000	
Sample mean difference	1,940	
Pooled standard deviation	1,513	NA
Std error of difference	0,309	0,274
Degrees of freedom	97	83
t-test statistic	6,284	7,076
p-value	$< 0,001$	$< 0,001$
Test of equality of variances		
Ratio of sample variances	5,296	
p-value	$< 0,001$	

Table 18: Two Sample Analysis for Machine Loc.-1 - Machine Loc.-2

Test of difference<= 0 versus one-tailed alternative		
Hypothesized mean difference	0,000	
Sample mean difference	0,247	
Pooled standard deviation	1,691	NA
Std error of difference	0,345	0,327
Degrees of freedom	97	97
t-test statistic	0,715	0,754
p-value	0,238	0,226
Test of equality of variances		
Ratio of sample variances	1,904	
p-value	0,017	

Table 19: Two Sample Analysis for Use-1 - Use-2

Test of difference<= 0 versus one-tailed alternative	
Hypothesized mean difference	0,000
Sample mean difference	-1,388
Pooled standard deviation	2,099
Std error of difference	0,428
Degrees of freedom	97
t-test statistic	-3,241
p-value	0,001
Test of equality of variances	
Ratio of sample variances	1,242
p-value	0,223

Table 20: Two Sample Analysis for Kinepolis-1 - Kinepolis-2

10% significance level. Again, this result can be expected as a lot of students still score badly in question 3 (version 1) for the *use* association.

Next, we test whether there is a difference on the level of the case, where the score for a case is defined as the sum of the score on both associations. The null hypotheses are no difference between the means, while the alternative hypotheses are a better score for the version with the pattern (version 2 in the Kinepolis case and version 1 in the Kidney case).

$$H_0 : \mu_{kinepolis-1} = \mu_{kinepolis-2}$$

$$H_a : \mu_{kinepolis-1} < \mu_{kinepolis-2}$$

$$H_0 : \mu_{kidney-1} = \mu_{kidney-2}$$

$$H_a : \mu_{kidney-1} > \mu_{kidney-2}$$

The results of the two-sample one-tailed t-test are shown in Table 20 for the Kinepolis case and in Table 21 for the Kidney case. For both cases, the null hypotheses can be rejected at 1% significance level. This means, for both cases, the solution where the pattern was available is significantly better than the solution where the pattern was unavailable.

In a last test, we check whether there is a significant improvement measurable on student level by comparing the score of question 2 with the score of question 4. The results of the paired t-test for the first association (*copy location* and *machine location*) is shown in Table 22, the second association (*show* and *use*) in Table 23 and for the total score in Table 24. For the three tests, the null hypothesis of equal scores can be rejected at a 1% significance

Test of difference<= 0 versus one-tailed alternative		
Hypothesized mean difference	0,000	
Sample mean difference	2,187	
Pooled standard deviation	1,981	NA
Std error of difference	0,404	0,383
Degrees of freedom	97	97
t-test statistic	5,409	5,713
p-value	$< 0,001$	$< 0,001$
Test of equality of variances		
Ratio of sample variances	1,921	
p-value	0,016	

Table 21: Two Sample Analysis for Kidney-1 - Kidney-2

paired test of difference <= 0 versus one-tailed alternative	
Hypothesized mean difference	0,000
Sample mean difference	-1.293
Pooled standard deviation	1.9286
Std error of difference	0.1938
Degrees of freedom	98
t-test statistic	-6.67
p-value	$< .0001$

Table 22: Paired t-test for Q21-Q41

level. Hence, for both associations and also for the total case, a significant improvement is measurable between the solution without pattern (question 2) and the solution with pattern (question 4).

3 Conclusions Experiment

From all three experiments we can conclude that the pattern induces a significant improvement in the model. Two hypotheses were tested and confirmed. Firstly, the score of participants with pattern is significantly better than the score of participants without pattern. And secondly, after participants read and applied the pattern, they are able to correct and improve their prior model significantly.

The second experiment shows us that although the concept of instantiable resources is known by the students (which can be deduced from the fact that a lot of them model this correctly in the SAI case even without reading the

paired test of difference <= 0 versus one-tailed alternative	
Hypothesized mean difference	0,000
Sample mean difference	-0.778
Pooled standard deviation	1.8711
Std error of difference	0.1881
Degrees of freedom	98
t-test statistic	-4.14
p-value	$< .0001$

Table 23: Paired t-test for Q22-Q42

paired test of difference <= 0 versus one-tailed alternative	
Hypothesized mean difference	0,000
Sample mean difference	-2.071
Pooled standard deviation	2.3615
Std error of difference	0.2373
Degrees of freedom	98
t-test statistic	-8.72
p-value	$< .0001$

Table 24: Paired t-test for Q2-Q4

INSTANTIABLE RESOURCE pattern), they are not able to apply this knowledge in different situations (e.g. in the Sunrise case). Hence, the pattern helps them to reveal the 'generic' knowledge and to make them able to apply this in different cases. However, maybe the cases in this second experiment were too small and made it too obvious for the participants how and where they could apply the pattern. We compensated for this by providing the participants with distinct sub-patterns.

In the third experiment, for both cases, there is only one of the two associations that experiences the improvement . Probably, this can be linked to the influence of the pattern on the creativity of the modeler. As the pattern has been applied on the other association, the participants are satisfied with their solution and do not look further to see whether their solution is complete. Nevertheless, the overall score for each case is significantly better with pattern.

References

[1] C. Alexander. *The Nature of Order: An Essay on the Art of Building and the Nature of the Universe.* Center for Environmental Structure, 2002.

[2] C. Alexander et al. *The timeless way of building.* Oxford University Press New York, 1979.

[3] C. Alexander, S. Ishikawa, M. Silverstein, et al. *A pattern language: towns, buildings, construction.* Oxford University Press, 1977.

[4] B. Appleton. Patterns in a Nutshell, The bare essentials of Software Patterns. 1999. `www.enteract.com/bradapp/docs/patterns-nutshell.html`.

[5] Rosana T. Vaccare Braga, Ferno S. R. Germano, and Paulo Cesar Masiero. A pattern language for business resource management. In *Procs. of PloP'99, http://st-www.cs.uiuc.edu/ plop/plop99*, pages 1–33, 1999.

[6] Rosana T. Vaccare Braga, Ferno S. R. Germano, and Paulo Cesar Masiero. Extension of the pattern language for business resource management, 2001. unpublished, obtained from the author.

[7] F. Buschmann, R. Meunier, H. Rohnert, P. Sommerlad, and M. Stal. *Pattern-oriented software architecture: a system of patterns.* John Wiley & Sons, Inc. New York, NY, USA, 1996.

[8] J.O. Coplien and N.B. Harrison. *Organizational patterns of agile software development*. Prentice Hall, 2004.

[9] M. Fowler. *Analysis Patterns: Reusable Object Models*. Addison-Wesley Professional, 1997.

[10] Martin Fowler. Writing software patterns, 2006. `http://martinfowler.com/articles/writingPatterns.html`.

[11] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design patterns: elements of reusable object-oriented software*. Addison-Wesley, 1995.

[12] David Hay. *Data Model Patterns: Convention of thoughts*. New York, 1996.

[13] M.L. Manns. Patterns: A Promising Approach To Knowledge Management. *First Annual ABIT Conference*, pages 3–5, 2001.

[14] R.C. Martin, D. Riehle, and F. Buschmann. *Pattern languages of program design 3*. 1997.

[15] G. Poels, A. Maes, F. Gailly, and R. Paemeleire. Pattern recognition of resource-event-agent conceptual modelling structures. Working paper 2006/369, Faculty of Economics and Business Administration, Ghent University, 2006.

[16] D. Riehle and H. Zuellighoven. Understanding and Using Patterns in Software Development. *Theory and Practice of Object Systems*, 2(1):3–13, 1996.

[17] L. Rising. *The Patterns Handbook: Techniques, Strategies, and Applications*. Cambridge University Press, 1998.

[18] M. Snoeck, G. Dedene, M. Verhelst, and A. M. Depuydt. *Object-oriented Enterprise Modeling with MERODE*. Leuven University Press, 1999.

[19] J. Vlissides. Seven Habits of Successful Pattern Writers. *C++ Report*, 8(10), 1996.