Empir Software Eng (2010) 15:493–522
DOI 10.1007/s10664-009-9125-9

An empirical study on the efficiency of different design pattern representations in UML class diagrams

Gerardo Cepeda Porras · Yann-Gaël Guéhéneuc

Published online: 27 February 2010

Editor: Laurie Williams

Abstract Design patterns are recognized in the software engineering community as useful solutions to recurring design problems that improve the quality of programs. They are more and more used by developers in the design and implementation of their programs. Therefore, the visualization of the design patterns used in a program could be useful to efficiently understand how it works. Currently, a common representation to visualize design patterns is the UML collaboration notation. Previous work noticed some limitations in the UML representation and proposed new representations to tackle these limitations. However, none of these pieces of work conducted empirical studies to compare their new representations with the UML representation. We designed and conducted an empirical study to collect data on the performance of developers on basic tasks related to design pattern comprehension (*i.e.,* identifying composition, role, participation) to evaluate the impact of three visual representations and to compare them with the UML one. We used eye-trackers to measure the developers' effort during the execution of the study. Collected data and their analyses show that stereotype-enhanced UML diagrams are more efficient for identifying composition and role than the UML collaboration notation. The UML representation and the pattern-enhanced class diagrams are more efficient for locating the classes participating in a design pattern (*i.e.,* identifying participation).

This work has been partly funded by the Canada Research Chair on Software Patterns and Patterns of Software, a NSERC Discovery Grant, and a CFI Infrastructure Grant.

G. Cepeda Porras
Ptidej Team, Département d'informatique et de recherche opérationnelle,
Université de Montréal, Montréal, Canada
e-mail: cepedapg@iro.umontreal.ca

Y.-G. Guéhéneuc (✉)
Ptidej Team, Département de génie informatique et génie logiciel,
École Polytechnique de Montréal, Montréal, Canada
e-mail: yann-gael.gueheneuc@polymtl.ca

Keywords Eye-tracking · Design patterns · Visualization · Empirical study · UML class diagrams

1 Introduction

Program comprehension is needed to construct a mental representation of the architecture of programs and to develop and maintain programs efficiently (Kazman et al. 1998). Diagrams are essential visual tools to construct these mental representations, highlighting useful information about objects and their relations (Chabris and Kosslyn 2005). In object-oriented software engineering, where objects are represented by classes, UML class diagrams are the facto standard to represent programs (Object Management Group 1997). These class diagrams are thought to facilitate program comprehension by reducing developers' effort in building their mental representation. They have been extensively studied in the program comprehension literature (Purchase et al. 2002; Eichelberger and von Gudenberg 2003; Sun and Wong 2005) and there exist many tools to build or generate UML class diagrams.

Design Patterns (Gamma et al. 1998) are solutions to recurring problems when designing object-oriented programs. They summarize and make explicit good design practices. The software engineering community pointed out that the knowledge and good use of design patterns is useful to improve program comprehension and quality, for example (Gamma et al. 1998; Shalloway and Trott 2002; Aversano et al. 2007). Currently, a common representation to visualize design patterns is the UML collaboration notation (Object Management Group 1997), noted *UML* in the following and exemplified in Fig. 1. This representation is common since its first use in the GoF's book "Design Patterns" (Gamma et al. 1998) and its use has

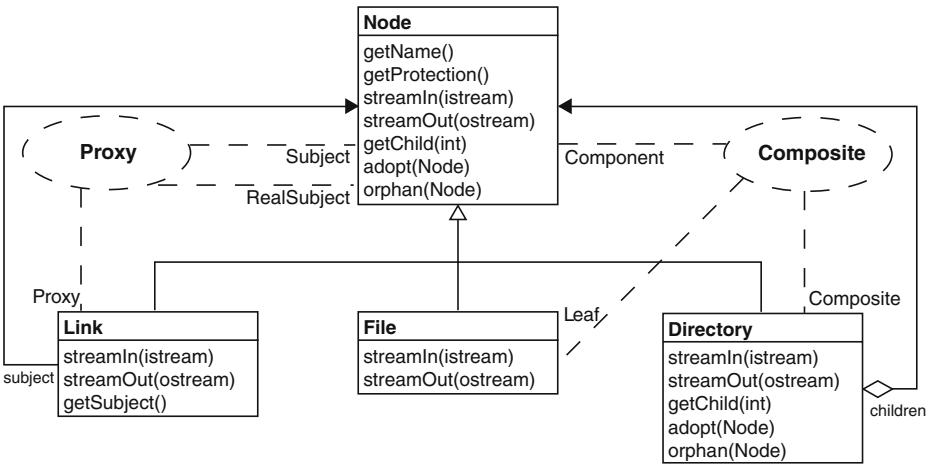

Fig. 1 UML collaboration notation *UML*, reproduced from Vlissides (1998), on a simple file system model

been advocated by respected designers/architects, including Rebecca Wirfs-Brock.[1] Previous work pointed out some limitations of this representation (as discussed in Section 2) and proposed alternative representations. These representations vary from strongly visual (Schauer and Keller 1998) to strongly textual (Dong et al. 2007). We can divide these representations in two groups: non-UML based representations (Eden et al. 1997; Mapelsden et al. 2002) and UML based representations (Gamma 1996; Lauder and Kent 1998; Schauer and Keller 1998; France et al. 2004; Trese and Tilley 2007; Dong et al. 2007). These two groups can be divided in two sub-groups: mono-diagram representations (Gamma 1996; Schauer and Keller 1998; Trese and Tilley 2007; Dong et al. 2007) and multi-diagram representations (Lauder and Kent 1998; France et al. 2004).

However, none of the previous pieces of work perform an empirical study to compare their proposed representations with the common one. Thus, conducting an empirical study to evaluate the efficiency of representations to visualize design patterns in UML diagrams is important: first, it gives a framework for comparing current and future notations; second, it shows that notations have advantages and weaknesses; and, third, its results could be use to motivate tool builders to include different notations for different tasks. In particular, it could help developers to choose the right representation for their tasks at hand and researchers by providing ground for future research in program comprehension to further improve existing representations.

In this study, we only consider UML-based mono-diagram representations because this group of representations is the most used now by the software engineering community. Thus, we retain three representations to analyze and compare the performance of developers with Representation *UML*. We choose these representations because they are the main representatives of the few attempts to propose an alternative to UML collaboration notation using UML-based mono-diagrams. These representations are:

- pattern-enhanced class diagrams, noted *Schauer*[2] in the following (strongly visual, see Fig. 2) (Schauer and Keller 1998);
- stereotype-enhanced UML diagrams, noted *Dong* in the following (strongly textual, see Fig. 3) (Dong et al. 2005, 2007);
- "pattern:role" notation, noted *Gamma* in the following (visual and textual, see Fig. 4) (Gamma 1996).

In our study, we design experiments to collect data to compare developers' performance while performing three basic tasks in design pattern comprehension:

- class participation, noted *Participation* in the following: identifying all the classes that participate in a design pattern;
- roles play, noted *Role* in the following: identifying the role a class plays in a given pattern;
- pattern composition, noted *Composition* in the following: identifying all design patterns in which a class participates.

[1] http://www.objectsbydesign.com/books/RebeccaWirfs-Brock.html

[2] In the following, for the sake of simplicity, we use the last name of the first author of a notation to denote its representation.

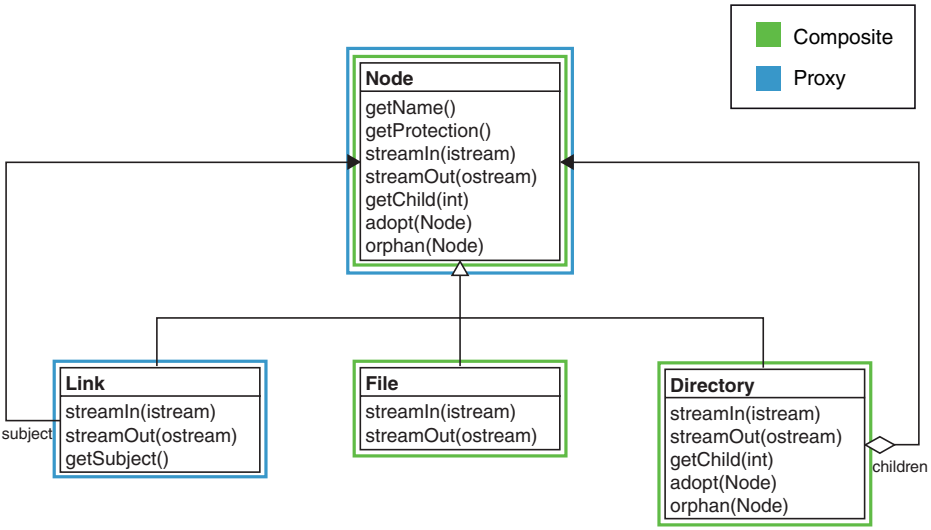

Fig. 2 Pattern-enhanced class diagrams, *Schauer*, on the same file system model used in Fig. 1

We measure performance in terms of the percentage of correct answers and of the developers' effort spend to perform the given tasks from data collected using eye-trackers: the less effort and the better percentage of correct answers, the greater the subject's performance. For each representation used in the study, we use the same UML class diagram to which we add the representations to visualize pattern-related information. Design patterns used in this study are: Composite, Prototype, Template Method, State, and Singleton. We also compare the effectiveness of each representation for diagrams with a small density of classes (15 classes) and with a larger density of classes (40 classes).

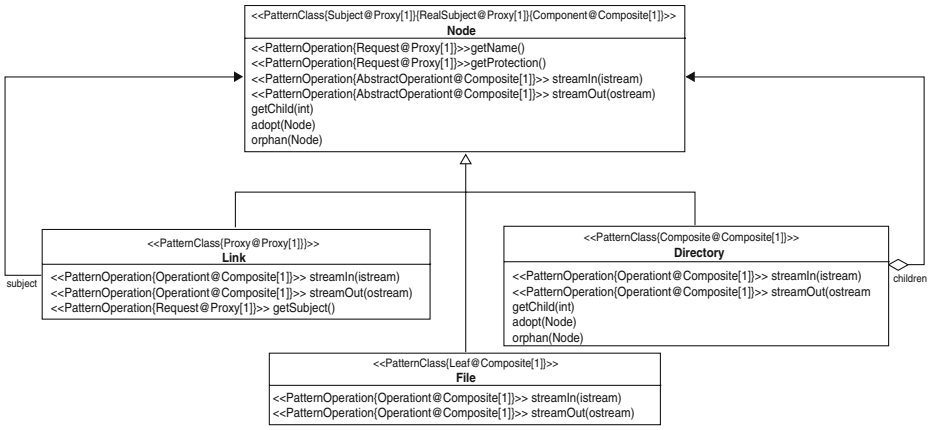

Fig. 3 Stereotype-enhanced UML diagrams, *Dong*, on the file system model used in Fig. 1

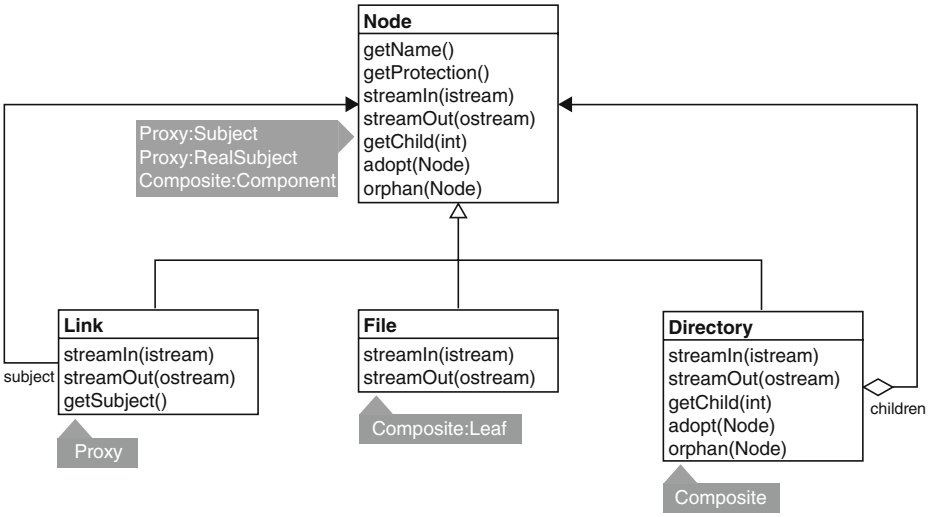

Fig. 4 Pattern:role notation, *Gamma*, reproduced from Vlissides (1998), on the same file system model used in Fig. 1

We collect data for 24 developers. We report that, for the diagrams of 15 classes, developers performed significantly better on Representation *UML* when compared to Representation *Dong* for Task *Participation*. However, for Tasks *Composition* and *Role*, Representation*Dong* performed significantly better than Representation *UML*. The other two representations did not show statistically significant differences when compared with Representation *UML*. However, Representation *Schauer* provides similar performance to Representation *UML* for Tasks *Participation* and *Composition*. Additionally, we report that the level of knowledge of design patterns could influence significantly the performance of users when performing Task *Composition* with representations *UML* and *Gamma*.

For the diagrams of 40 classes, we report that developers performed significantly better on Representation *UML* when compared to *Dong* for Task *Participation* similarly to the results of 15 classes diagrams. For the other two representations, we cannot report any statically significant differences. For the other two tasks, we cannot report any statically significant differences. We also point out the influence of readability on the results for the diagrams of 40 classes.

The remainder of this paper is organized as follows. In the following Section 2, we present related work. In Section 3, we present the experimental design and the running of our experiments. We analyze the collected data and present results in Section 4. We assess the validity of our study in Section 5 and conclude in Section 6.

2 Related Work

This paper relates to three fields of study: program comprehension, design pattern visualization on UML class diagrams, and eye-tracking studies.

2.1 Program Comprehension

Program comprehension is subject of several research works.

Several authors proposed and evaluated models to form and abstract a mental representation of a program to achieve program comprehension. For example, Soloway et al. (1988) suggested that developers use plans (Rich and Waters 1988) when comprehending a program. Von Mayrhauser (1995) described the process of program comprehension as a combination of top-down and bottom-up tasks, based on existing knowledge.

All developers use diagrams as a means to convey information to other developers or to better understand programs. Diagrams reduce the comprehension and learning effort by omitting irrelevant details and highlighting pertinent information about objects and their relations. The closer the information presented on diagrams is to the developer's mental representation, the easier it is to understand (Chabris and Kosslyn 2005). Several studies have been conducted about program comprehension using UML class diagrams. For example, Purchase et al. (2002) conducted a study on the preference of developers on aesthetics of UML class diagrams. They concluded on aesthetic criteria for UML class diagrams, including joined inheritance arcs and directional indicators. Eichelberger (2003) studied the relations between semantics in UML class diagrams, principles of human–computer interactions, and principles of object-oriented modelling. He presented aesthetic criteria to layout UML class diagrams to improve readability.

Sun and Wong (2005) classified selected criteria from previous work, using laws from the Gestalt theory of visual perception, the organizational perception theory, and the segregative perception theory (Moore and Flitz 1993). They retained 14 criteria to assess the quality of UML class diagram layouts. They used these criteria to evaluate the efficiency of layout algorithms of two commercial tools, Rational Rose and Borland Together. They concluded on the good quality of both tools, the difficulty of both tools to satisfy all criteria. They suggested characteristics to be improved in these two tools in the future. We use these criteria to layout our diagrams.

Previous work provides a good basis for building empirical studies on program comprehension and important criteria to layout our UML class diagrams.

2.2 Design Pattern Visualization

Expressing design decisions by highlighting the design patterns used in an existing architecture leads to a better understanding of how a program works. Conversely, the lack of pattern-related information could impede the program comprehension process.

A common representation to visualize design patterns is the UML collaboration notation, *UML* (Object Management Group 1997) (as illustrated in Fig. 1, also called parameterized collaboration diagrams). This representation uses dashed ellipses (with the patterns names) and lines (with the role names that classes play) to associate the patterns to their participating classes. However, too many dashed

lines lead to reading problems because information is being mixed (and sometimes cluttered) with other diagram elements.

With the goal of removing the cluttering dashed lines in *UML*, Vlissides (1998) proposed Representation *Gamma*, where all pattern-related information is contained in shaded boxes which are placed close to the classes participating in patterns (see Fig. 4). This representation first appears in the GoF's book (Gamma 1996), before being described in Vlissides (1998), hence its name. This representation is highly readable because it puts the pattern-related information in another plan with the diagram. However, this representation could increase significantly the size of the original diagram. Also, the combination of gray boxes with white typography could lead to reading problems on printed media.

Schauer and Keller (1998) implemented a prototype to ease program comprehension based on design pattern recognition and visualization techniques. To visualize design patterns, the prototype offers three views:

- the pattern-enhanced class diagrams, Representation *Schauer*, a mono-diagram UML-based representation that uses different colored borders to identify pattern participation and also uses the canonical representation (Gamma et al. 1998) to help users infer the roles each class plays on that pattern, see Fig. 2;
- the pattern-analysis view, a multiple view composed of the first view and a catalog of design patterns showing their intents, applications, and consequences;
- a dynamic view called pattern collaboration diagrams, which shows collaborations between implemented design patterns on a pattern-level and also in the class-level dynamically.

Trese and Tilley (2007), to improve readability and comprehension of programs, proposed the class participation diagrams, as illustrated in Fig. 5. In this representation, classes are clustered by design pattern (showed in their canonical representation) and grouped by categories: creational, behavioral, and structural (as defined in Gamma et al. (1998)). They also apply some aesthetics criteria in spacing and shading to ease readability. However, this representations lacks some important information, such as the name of the design patterns or the role a class play, that can lead to confusion.

Dong et al. (2007) proposed a UML profile with new stereotypes, tagged values, and constraints to visualize pattern-related information in UML class diagrams. Their Representation, *Dong*, uses tagged values to hold information about the roles that a class, a method, or an attribute plays in a design pattern and also deals with multiple instances of design patterns, as shown in Fig. 3. This representation has the advantage of expressing clearly pattern-related information. However, the text overload could increase considerably the size of classes as well as make the diagrams harder to read. To address this issue, they developed a Web service, VisDP, to dynamically visualize design pattern information on demand.

Previous work provides us with several representations for our study on design pattern visualization using UML class diagrams. These representations vary from mostly graphical to mostly textual. To the best of our knowledge, none of this previous work conducted empirical studies to evaluate the efficiency of their representations with the common representation.

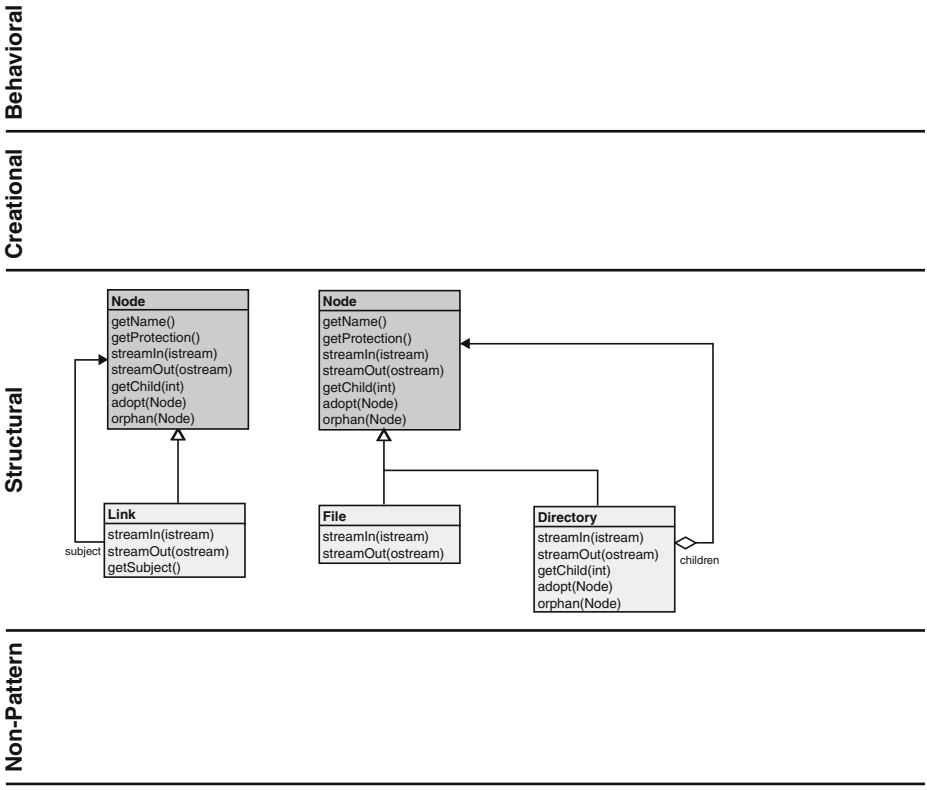

Fig. 5 Class participation diagrams, on the same file system model used in Fig. 1

2.3 Eye-tracking Studies

Eye tracking systems collect eye movement data to provide an insight into a subject's focus of attention, making it possible to draw conclusions about the underlying cognitive processes (Rayner 1998). These systems are based on the physiology of human visual capabilities and cognitive theories, like the theories on visual attention and visual perception (Duchowski 2003).

Eye trackers have been traditionally used in cognitive psychology (Rayner 1998). These systems are also increasingly used in other domains, such as marketing, industrial design, and computer science. In computer science, eye-trackers have been used in studies on graphic data processing, human–computer interfaces, and virtual reality (Duchowski 2003). The software engineering community started only recently to show an interest in using eye-tracking systems to study program comprehension.

For instance, Bednarik and Tukiainen (2006) proposed an approach to study trends on repeated measures of sparse data over a small data set of program comprehension activities captured with eye-trackers. Using this approach, they characterized program comprehension strategies using different program representations (code lecture and program execution). The second author of this paper also conducted an experiment with eye-trackers to study how software engineers acquire and

use information from UML class diagrams (Guhneuc 2006). He concluded on the importance of classes and interfaces and reported that developers seem to barely use binary class relationships, such as heritage or composition. Yusuf et al. (2007) conducted a similar study to analyze the utilization of specific characteristics of UML class diagrams (e.g., layout, color, and stereotypes) during program comprehension. They concluded on the efficiency of layouts with additional information as colors or stereotypes to improve program comprehension.

We follow this previous work in this study of the efficiency of representations for design pattern visualization on program comprehension.

3 Experimental Design

The design of our study aims at testing whether or not developers' performance improves when performing design pattern comprehension tasks using Representations *Dong*, *Gamma*, and *Schauer*, when compared to Representation *UML*. We measure performance in terms of the percentage of correct answers and developers' effort spend to perform the given tasks. To assess the developers' effort while executing the study, we use eye trackers to collect relevant data as in previous studies (Guhneuc 2006; Yusuf et al. 2007). Therefore, our design is directed by the use of eye-trackers. We first present and justify our choice for the three representations and three tasks used in this study. Then, we detail our hypotheses, objects, dependent and independent variables, and subjects. Finally, we present briefly our eye-tracking system and the procedure followed to carry out this study.

3.1 Representations

An analysis of the representations proposed in the literature leads us to choose: Representation *Schauer* proposed by Schauer et al. (1998) for its simplicity, its ease for visually identify design patterns, and its use of the canonical representation to identify key information about design patterns (see also Trese and Tilley's approach (Trese and Tilley 2007)). Representation *Dong* proposed by Dong et al. (2007) was selected because it is strongly textual. We retained Representation *Gamma* (Gamma 1996; Vlissides 1998) because it is both a visual and a textual notation. In addition, Representation *Gamma* shows the information relative to design patterns on a different plan (because of the shading effect) that could also reduce the developers' effort and ease the reading of diagrams. We reject Trese and Tilley's approach because it lacks some important information as explained in the previous section and also because it changes the original UML class diagram and we would not be able to compare it fairly to the other representations.

3.2 Tasks

In the cognitive approach of systems (Hutchins 1995), the goal is to find a means to facilitate data acquisition (Ware 2005). Following this approach, our tasks were designed to measure the cognitive charge, in terms of developers' effort, related to pattern comprehension activities. We choose 3 tasks for which a representation of design patterns could be useful, that are recurring in program comprehension, and

that keep experimental trials short in time, ensuring the highest accuracy of recorded data (Guan et al. 2006):

- CLASS PARTICIPATION, *Participation*: i.e., to identify all classes participating in a design pattern.
- PATTERN COMPOSITION, *Composition*: i.e., to identify all design patterns a class participates in.
- ROLES PLAYED, *Role*: i.e., to identify the roles a class play in a design pattern.

We will use these three tasks (which are key for pattern comprehension) to compare the different representations with *UML*. We rejected other tasks, for example the tasks of identifying multiple instances of the same design pattern, because it cannot be performed by developers satisfactorily using all the representations.

3.3 Hypotheses

We want to assess the following three null hypotheses when performing the Tasks *Participation*, *Composition*, and *Role*:

- H_{0_1}: There is no difference in the average effort and accuracy of subjects using Representation *UML* and subjects using Representation *Dong*.
- H_{0_2}: There is no difference in the average effort and accuracy of subjects using Representation *UML* and subjects using Representation *Gamma*.
- H_{0_3}: There is no difference in the average effort and accuracy of subjects using Representation *UML* and subjects using Representation *Schauer*.

If the previous null hypotheses are rejected, we could assume (with respect to the threats to the validity assessed in Section 5) that either one of each of the following alternative hypotheses are verified:

- $H_{\alpha_{3.1}}$: The average effort and accuracy is superior for subjects using Representation *Dong* than for subjects using Representation *UML*.
- $H_{\alpha_{3.2}}$: The average effort and accuracy is inferior for subjects using Representation *Dong* than for subjects using Representation *UML*.
- $H_{\alpha_{2.1}}$: The average effort and accuracy is superior for subjects using Representation *Gamma* than for subjects using Representation *UML*.
- $H_{\alpha_{2.2}}$: The average effort and accuracy is inferior for subjects using Representation *Gamma* than for subjects using Representation *UML*.
- $H_{\alpha_{1.1}}$: The average effort and accuracy is superior for subjects using Representation *Schauer* than for subjects using Representation *UML*.
- $H_{\alpha_{1.2}}$: The average effort and accuracy is inferior for subjects using Representation *Schauer* than for subjects using Representation *UML*.

We choose to compare the three representations *Schauer*, *Gamma*, and *Dong* against *UML* and not against one another for two reasons. First, a study of the different notations against one another would have required much more subjects.

Second, we do not compare the aggregated results of diagrams of 15 and 40 classes because we do not assess and ensure that they have the same complexity.

3.4 Objects

We choose the open-source program JHotDraw for our study. JHotdraw (2007) is a framework to implement technical and structured drawings. It was designed by Gamma and Eggenschwiler as a show case for the use of design patterns.

Because full documentation was not available, partial reverse engineering was performed on JHotDraw to obtain its design as a UML class diagram. We obtained one diagram from JHotDraw by reverse engineering and then we created two diagrams (of 15 and 40 classes) by selecting a set of consistent classes the use the following design patterns: Composite, Prototype, Template Method, State, and Singleton, as implemented in JHotDraw. We chose these six patterns among other patterns implemented in JHotDraw, for example Strategy, because they are representatives of creational, behavioural, and structural design patterns and because they are clearly distinguishable among themselves. Also, a previous study (Khomh and Guéhéneuc 2008) showed that these patterns have mostly a positive subjective impact of software quality characteristics (expendability, understandability, and reusability). It is therefore interesting to study if they also have a positive impact on effort and accuracy.

We follow the canonical representation of the chosen design patterns (i.e., as described in Gamma et al. (1998) as much as possible). However, there are slight differences mainly due to language implementation issues (e.g., the use of interfaces and abstract classes in Java). Both diagrams of 15 classes (see Fig. 6) and diagrams of 40 classes (available on-line at http://url.hidden-for-double-blind.review due to space constraints), have the same classes participating in design patterns. There is only a

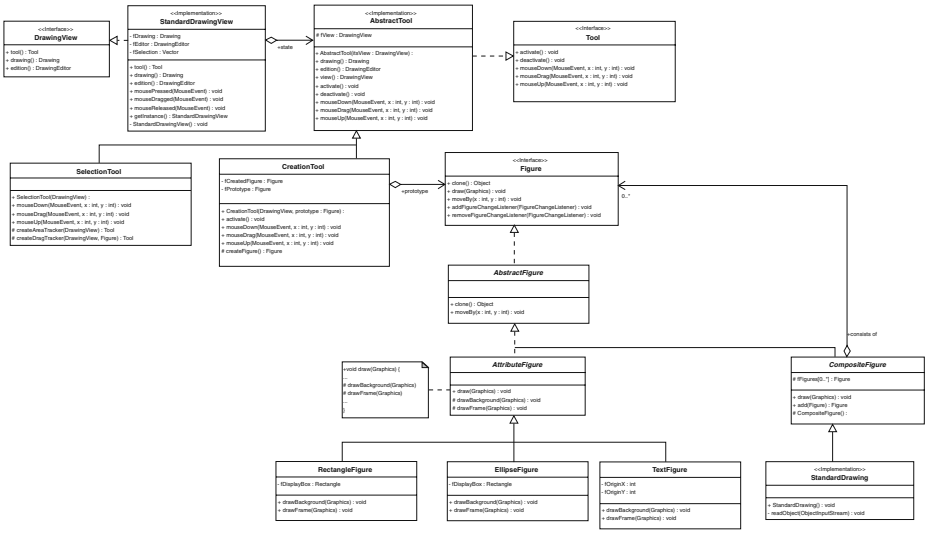

Fig. 6 JHotDraw diagram of 15 classes used in the study

slight change in the layout of the diagrams of 40 classes because of the relationship between the new classes added to the diagram. All representations are superposed on the same two UML class diagrams. All the diagrams used for the study are available on the companion web site.

3.5 Independent Variables

From the hypotheses, we identify the following independent variables:

- Representations: *UML*, *Schauer*, *Gamma*, *Dong* are the possible values for this variable, these values represent the four representations chosen in our study. We use the indexes 15 and 40 to distinguish between diagrams with small class density from diagrams with larger class density. We chose to analyze separately diagrams with different class density because the graph complexities are different.
- Tasks: *Participation*, *Composition*, *Role* are the values for this variable. these values represent the three tasks chosen in our study.

We retain two mitigating variables to study and better understand the results of our study:

- JHotDraw Knowledge: The subjects' knowledge of JHotDraw. The level is established using a questionnaire. Values are taken from [0,1,2] where 2 means that a subject has a good knowledge of JHotDraw, 1 that the subject has a basic knowledge of JHotDraw, and 0 that the subject has no knowledge of JHotDraw.
- DP Knowledge: The subjects' knowledge of design patterns. The level is also established using a questionnaire following the same method as for the previous variable.

3.6 Dependent Variables

The dependent variables are chosen according to our hypotheses and independent variables based on the capabilities of eye-tracking systems. We measure performance in terms of correct answer percentage *(CAP)* and the developers' effort spend to perform given tasks. We establish for each diagram ($Dong_{15}/Dong_{40}$, $Gamma_{15}/Gamma_{40}$ $Schauer_{15}/Schauer_{40}$, and UML_{15}/UML_{40}) a set of *area of interest* (AOI) and a set of *area of glance* (AOG). An *area of glance* is any class or notation element part of the diagrams. An *area of interest* is a *relevant* class or notation element in a diagram that should be the focus of the subjects' attention to perform a particular task *Participation*, *Composition*, or *Role*. Both sets vary with the task to perform. We collect data about fixations on AOI and AOG to compute developers' effort. From fixations collected (see Section 3.9), we use the following metrics:

- Average Fixation Duration (AFD): This measure is correlated with cognitive functions (Goldberg and Kotval 1999; Duchowski 2003). It is computed as follows:

$$AFD = \frac{\sum_{i=1}^{n} (ET(F_i) - ST(F_i)) \; in \; AOG}{n}$$

where $ET(F_i)$ and $ST(F_i)$ represent the end time and start time for fixation F_i and n represent the total number of fixations in AOG . Longer fixations mean that users are spending more time interpreting or assembling the representation elements to build their internal mental representation. Representations that require shorter fixations are thus more efficient.

– Ratio of "On_target:All_target" Fixation Time (ROAFT) (Goldberg and Kotval 1999): The ratio of the time passed in the AOI divided by the time passed in the AOG sets. It is computed as follows:

$$ROAFT = \frac{\sum_{i=1}^{n} (ET(F_i) - ST(F_i)) \; in \; AOI}{\sum_{j=1}^{m} (ET(F_j) - ST(F_j)) \; in \; AOG}$$

where $ET(F_i)$, $ET(F_j)$ and $ST(F_i)$, $ST(F_j)$ represent the end time and start time for fixation F_i or F_j respectively, n and m represent the total number of fixations in AOI and AOG respectively. Smaller ratios indicate lower efficiency.

– Ratio of "On_targer:All_target" Fixations (ROAF) (Goldberg and Kotval 1999): This ratio is a content-dependent efficiency measure of visual search (Duchowski 2003). It is computed as follows:

$$ROAF = \frac{Total \; Number \; of \; Fixations \; in \; AOI}{Total \; Number \; of \; Fixations \; in \; AOG}$$

Smaller ratios indicate lower efficiency caused by a greater effort needed to find the pertinent elements required to perform the task.

3.7 Subjects

The study was performed by 24 subjects doing their Ph.D. or M.Sc. studies at the Department of Informatics and Operations Research at University of Montreal. All subjects were volunteers. They have designed software architecture and used UML class diagrams and design patterns during their studies for at least two years.

We design our experiment as two between-subjects experiments on diagrams of 15 and 40 classes. The subjects are placed into balanced groups. Using balanced groups simplifies and strengthens the statistical analysis of collected data (Wohlin et al. 2000). Each subject performs the experience for the three different Tasks *Composition*, *Participation*, and *Role* over two different representations with different class densities. Table 1 show the subjects's distribution for the study.

Table 1 Subjects's distribution for the study

	15	40
Dong	$S_9, S_{11}, S_{12}, S_{21}, S_{23}, S_{24}$	$S_3, S_5, S_6, S_{15}, S_{17}, S_{18}$
Gamma	$S_6, S_8, S_{10}, S_{18}, S_{20}, S_{22}$	$S_2, S_4, S_{12}, S_{14}, S_{16}, S_{24}$
Schauer	$S_4, S_5, S_7, S_{16}, S_{17}, S_{19}$	$S_1, S_{10}, S_{11}, S_{13}, S_{22}, S_{23}$
UML	$S_1, S_2, S_3, S_{13}, S_{14}, S_{15}$	$S_7, S_8, S_9, S_{19}, S_{20}, S_{21}$

Table 2 Questions for tasks *composition*, *participation* and *role*

Pattern composition task
Q1. Mention all design patterns the class CreationTool participates in.
Q2. Mention all design patterns the class AttributeFigure participates in.
Class participation task
Q3. Mention all classes participating in the Composite design pattern.
Q4. Mention all classes participating in the State design pattern.
Roles played by a class task
Q5. Mention all roles played by the class AbstractFigure.
Q6. Mention all roles played by the class StandardDrawingView.

3.8 Questions and Stimulus

Choosing the appropriate question is particularly important for eye-tracking studies (Duchowski 2003) because eye movements are dependent on the nature of the task at hand. Therefore, we choose questions that allow the subjects to answer in a short delay (one to two minutes) using only the information given by the different representations. We defined two questions for each task, one for each class density. Table 2 shows the questions used for the experiment and Fig. 7 shows the questions' distribution in the study. These questions are appropriate to analyze the efficiency of the studied representations.

An object of attention viewed by a subject is called a "stimulus". We combine one question with one diagram to form a stimulus. Previous work highlighted the subjects' tendency to look to the stimulus' top-left corner (Goldberg et al. 2002;

S1	D15,UML	Q1	Q4	Q6
	D40,Schauer	Q3	Q2	Q5
S2	D15,UML	Q3	Q2	Q5
	D40,Gamma	Q1	Q4	Q6
S3	D15,UML	Q4	Q2	Q5
	D40,Dong	Q6	Q3	Q1
S4	D15,Schauer	Q6	Q1	Q4
	D40,Gamma	Q2	Q5	Q3
S5	D15,Schauer	Q5	Q3	Q2
	D40,Dong	Q4	Q1	Q6
S6	D15,Gamma	Q6	Q1	Q3
	D40,Dong	Q2	Q4	Q5

S7	D40,UML	Q1	Q4	Q6
	D15,Schauer	Q3	Q2	Q5
S8	D40,UML	Q6	Q1	Q3
	D15,Gamma	Q2	Q4	Q5
S9	D40,UML	Q5	Q3	Q2
	D15,Dong	Q4	Q1	Q6
S10	D40,Schauer	Q3	Q2	Q5
	D15,Gamma	Q1	Q4	Q6
S11	D40,Schauer	Q4	Q2	Q5
	D15,Dong	Q6	Q3	Q1
S12	D40,Gamma	Q5	Q3	Q2
	D15,Dong	Q4	Q1	Q6

S13	D15,UML	Q6	Q2	Q4
	D40,Schauer	Q3	Q5	Q1
S14	D15,UML	Q5	Q4	Q2
	D40,Gamma	Q1	Q3	Q6
S15	D15,UML	Q4	Q2	Q5
	D40,Dong	Q3	Q6	Q1
S16	D15,Schauer	Q6	Q1	Q4
	D40,Gamma	Q2	Q5	Q3
S17	D15,Schauer	Q2	Q4	Q6
	D40,Dong	Q5	Q1	Q3
S18	D15,Gamma	Q3	Q6	Q1
	D40,Dong	Q4	Q2	Q5

S19	D40,UML	Q4	Q1	Q6
	D15,Schauer	Q5	Q3	Q2
S20	D40,UML	Q1	Q4	Q6
	D15,Gamma	Q3	Q2	Q5
S21	D40,UML	Q3	Q2	Q5
	D15,Dong	Q6	Q4	Q1
S22	D40,Schauer	Q5	Q1	Q3
	D15,Gamma	Q4	Q6	Q2
S23	D40,Schauer	Q1	Q6	Q3
	D15,Dong	Q4	Q2	Q5
S24	D40,Gamma	Q4	Q1	Q5
	D15,Dong	Q2	Q3	Q6

Fig. 7 Questions' distribution in the study

Nommer tous les patrons de conception dans lesquels participe la classe CreationTool

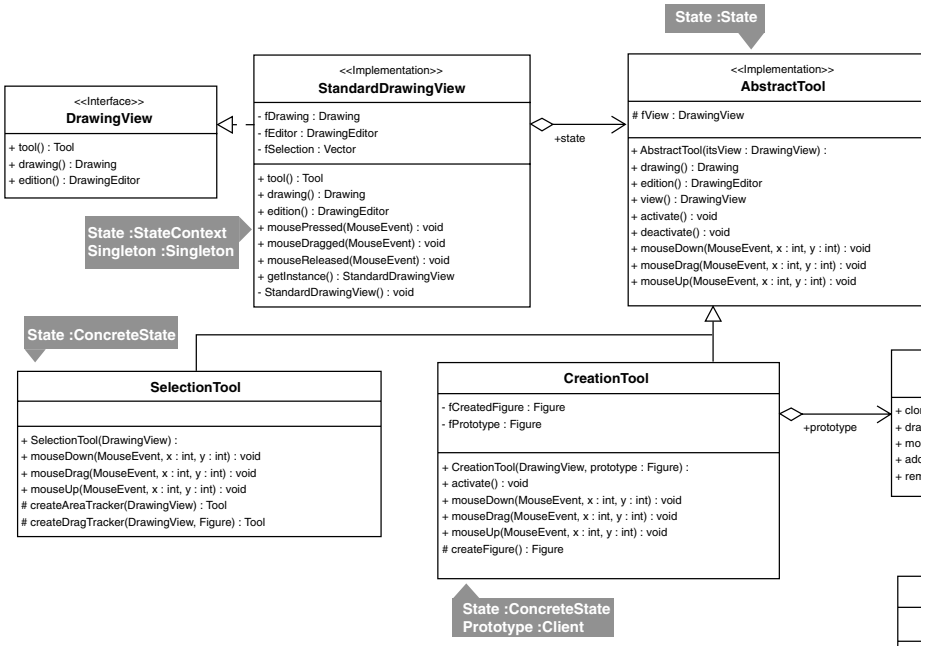

Fig. 8 Portion of the stimulus with Representation *Gamma*. (The question is in French because all subjects were French-speakers)

Bojko 2005). Therefore we placed the question on the top-left corner of the screen, the diagram filling the rest of the screen, as shown in Fig. 8, to prevent any bias towards elements of the representations placed in the top-left corner.

The subjects were requested to provide their answer aloud, to avoid head movements that could decalibrate the eye-tracker. Also, previous work (Guan et al. 2006) showed that asking subjects to provide their answers aloud after having first recorded their eye-movements to find the answer does *not* alter their gaze patterns. The experimenter used a check-list to assess the accuracy of each answer. The experiment was approved by the Ethical Review Board of Université de Montréal.

3.9 Equipment

We used the EyeLink II eye-tacking system from SR Research[3] to perform our study. This system has a high resolution (noise limited to 0.01°) and fast data rate (500 samples per second). Its precision has an average gaze position error $< 0.5°$.

[3] http://www.eyelinkinfo.com/

The eye-tracker is composed of two computers and a head-band. One computer is used for experiment execution and the other for system calibration and data processing. The two computers communicate by an Ethernet connection. The head-band includes two cameras and an infra-red emitter. The cameras use infra-red rays that are reflected on the subject's cornea to register eye-movements. Four sensors are placed on the subject's screen. These sensors work with the infra-red emitter and allow computing the position of the head-band with respect to the screen.These four sensors in combination with the cameras allow the system to compute precisely the position of the subject's gaze on the screen.

The communication between the two computers is based on the principle of "Action/Event" → "Reaction". When an event is emitted by the subject's computer, the experimenter's computer reacts and takes the control back. The experimenter's computer compute the position of the subject's gaze and record this data on the disk, in real time. When the experimentation is finished, the experimenter's computer sends back the whole data file to the subject's computer for future analysis. Fig. 9 shows all eye-tacking system's components.

The system can collect two types of data: raw positions and parsed positions. Parsed positions are given in terms of fixations and saccades based on physiological thresholds. A fixation is a stabilization of the eye during a gaze. A saccade is a quick movement of the eye from one fixation to another. The eyes are only sensitive to the details in the center of the visual field, visual information is only treated during fixations and not during saccades (Rayner 1998; Duchowski 2003; Ware 2005). Similarly to previous work (Guhneuc 2006; Yusuf et al. 2007), we use fixations as a measure of the amount of attention given by a subject to the different AOI and AOG of a diagram.

We use a dentist chair to configure the environment of the experiment easily: to align the subject's head with the four sensors, avoid movements, and give more comfort. We also use a travel pillow to give support to the subject's neck and further

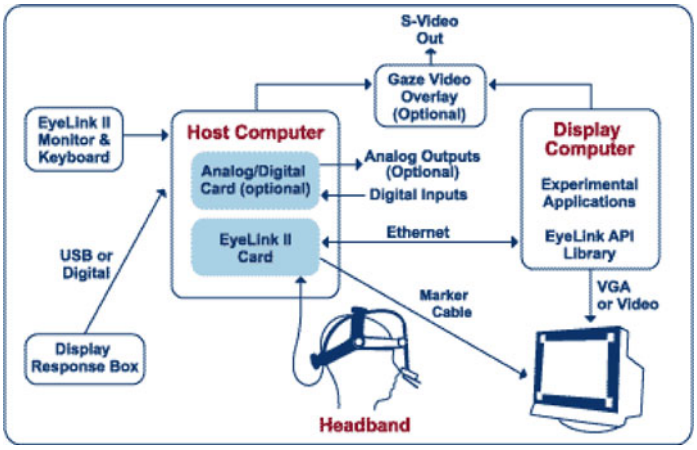

Fig. 9 Eye-tracker's components (from SR Research web site)

reduce head movements. We use a 17" CRT screen to show the stimulus. For each subject, a session takes about 40–50 min, including a presentation of all the steps of the study and legal issues, a tutorial on the representations and kinds of tasks, eye-tracker's calibration, data collection, and questionnaires.

3.10 Procedure

The experiments were conducted in a quiet room without any disturbances. The procedure is as follows:

1. First, the subject registers for the study in person, by email, or using the inscription Web site. The Web site gives a description of the study and a video showing the eye-tracking system.
2. We present a tutorial to introduce the four representations used for the study with some example diagrams to familiarize the subjects with the tasks they will perform later. This tutorial is helpful to alleviate anxiety and to give all subjects the same base knowledge necessary to use the representations (as shown by the high percentage of correct answers, CAP, detailed in Section 4). The tutorial is useful also to reduce the learning effect.
3. At the end of the tutorial, we give a presentation of the eye tracker used to collect data and important instructions on the use of the system, e.g., to avoid head movements while performing the tasks.
4. Then, we install the subject on a fixed dentist chair and we put a travel pillow around the subject's neck to make the subjects comfortable while avoiding head movements.
5. We explain the technical settings of the running of the study: each question is displayed on the screen at the top-left corner of the screen, a subject must give the answer aloud at the end of each task. Once a task finished and the answer given, the subject presses the "Escape" key to go to the next stimulus.
6. Before performing the tasks, the eye-tracker is calibrated. Calibration requires the subjects to fix one by one nine points on the screen.
7. After calibration, a short text presenting JHotDraw is shown to the subject, summarizing its intent and main characteristics. This text provides enough information to perform the tasks.
8. Then, data collection begins while the subjects perform the tasks. No time limit is set but subjects are requested to answer aloud as soon as possible. We use a check-list to assess the answers of each task. If all elements of the list are covered, then the subject's answer is correct, otherwise the answer is considered incorrect.
9. Finally, we provide each subject with a short written auto-evaluation questionnaire to evaluate their knowledge of design patterns and JHotDraw. We give this questionnaire after the experimentation not to reveal the purpose of our study.
10. At the end of the experiment, we give a symbolic present to the subjects for their participation. We ask them not to share any information with other potential subjects until the date of the end of the complete study.

Table 3 Collected data

Collected data	
Number of subjects (#)	24
Data (*Mb*)	36
Number of videos (#)	144
Total time of experiments (*hours*)	18
Total time eye-tracking (*min*)	112.4
Total number of fixations	21,395
Diagrams of 15 classes	
Average time on task *Composition*(*sec*)	36
Average time on task *Participation*(*sec*)	33
Average time on task *Role*(*sec*)	54
Diagrams of 40 classes	
Average time on task *Composition*(*sec*)	49
Average time on task *Participation*(*sec*)	52
Average time on task *Role*(*sec*)	58

4 Analysis and Results

We discuss here the results of testing our hypotheses. We use the Student *t*-Test after verifying that all the collected data is normal. The Student *t*-Test is a robust statistical test that can be used when the sample size is small. It only assumes random independent samples with normal distributions or distributions close to the normality.

We explore two factors that could mitigate these results, DP Knowledge and JHotDraw Knowledge. Table 3 summarizes some statistics of the collected data. Next sections will show the result of our analyses on the effects of adding the Representations *Dong*, *Gamma*, *Schauer*, and *UML* on UML class diagrams for the tasks *Participation*, *Composition*, and *Role* respectively.

4.1 Data Analysis of Task *Participation*, 15 Classes

Subjects performed better for the metric of correct answer percentage *CAP* using *UML* than using any of the other three representations. Diagrams using Representation *Dong* are effort-consuming with respect to *UML*: p-values around 0.007 and less for both the ratio of fixations, *ROAF*, and the ratio on fixation time, *ROAFT*, and differences in *CAP* and in the average fixation duration *AFD* showing the same tendency, see Tables 4 and 5. The mean differences between *Dong* and *UML* shows a difference of 0.25 and 0.29 in *ROAF* and *ROAFT*, meaning more effort

Table 4 Means of dependent variable values for Representations *Dong*, *Gamma*, *Schauer*, and *UML* on Task *Participation*, diagrams with 15 classes

Diagrams	Perf.	Effort measures		
	CAP (%)	AFD (ms)	ROAF (#)	ROAFT (ms)
Dong$_{15}$	50.00	277.61	0.54	0.55
Gamma$_{15}$	60.00	270.60	0.70	0.74
Schauer$_{15}$	83.33	273.85	0.84	0.87
UML$_{15}$	100.00	264.66	0.79	0.84

Table 5 Hypotheses testing for representations *Dong*, *Gamma*, *Schauer*, and *UML* on Task *Participation*, diagrams with 15 classes

Diagrams	P-values		
	AFD (ms)	ROAF (#)	ROAFT (ms)
H_{01}: **Dong$_{15}$** vs. **UML$_{15}$**	0.79	0.290	0.380
H_{02}: **Gamma$_{15}$** vs. **UML$_{15}$**	0.90	0.120	0.080
H_{03}: **Schauer$_{15}$** vs. **UML$_{15}$**	0.70	0.007	<0.001

with Representation *Dong*. A difference in *AFD* of roughly 13*ms* showing more effort with Representation *Dong*. Finally, *CAP* has a value of 50% with *Dong* in comparison to 100% with Representation *UML*.

Figure 10 shows that subjects working with Representation *Dong* exhibit clearly lower values for *ROAF* and *ROAFT* and a higher value in *AFD*. However, subjects using Representation *Dong* have a uniform performance, *i.e.*, flat box plots, compared to *Gamma*, *Schauer*, and *UML*. *Dong* is a representation rich in semantics and that could explain this little variance observed in the box plots.

We conclude that for Task *Participation*, the average effort is superior in subjects using Representation *Dong* than in subjects using *UML*. For the other two representations, we do not report statistically significant differences on the effort. Representation *UML* has more correct answers than the other two representations. However, it is important to mention that Representation *Schauer* shows similar performances to *UML* for this task. Representation *Gamma* shows a larger variance in subject's

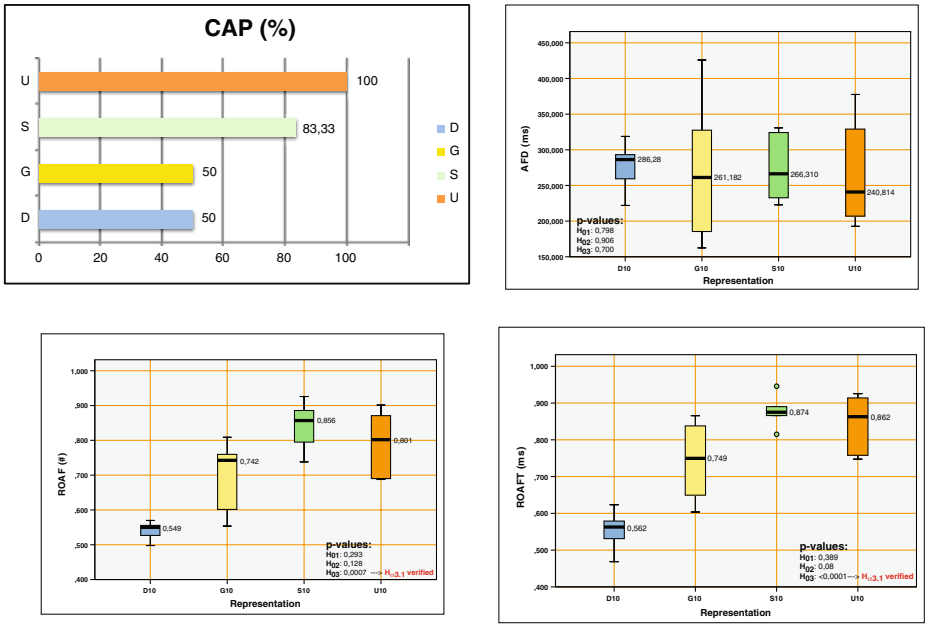

Fig. 10 Data distribution for Task *Participation*, diagrams of 15 classes

Table 6 Means of dependent variable values for representations *Dong*, *Gamma*, *Schauer*, and *UML* on Task *Composition*, diagrams with 15 classes

Diagrams	Perf.	Effort measures		
	CAP (%)	AFD (ms)	ROAF (#)	ROAFT (ms)
Dong$_{15}$	100.00	280.11	0.66	0.73
Gamma$_{15}$	50.00	279.76	0.65	0.70
Schauer$_{15}$	100.33	230.84	0.56	0.63
UML$_{15}$	83.33	245.12	0.42	0.45

effort in *AFD*, i.e., bigger box plots compared to the other representations, which could mean an influence of one or both of the mitigating factors on the performance of subjects for *Gamma*.

4.2 Data Analysis of Task *Composition*, 15 Classes Diagram

For Task *Composition*, most subjects performed well for *CAP*. The results on the metrics show that diagrams using Representation *Dong* are less effort-consuming than Representation *UML* (p-values around 0.01 for both *ROAF* and *ROAFT* and a slightly better *CAP* with respect to *UML*, see Tables 6 and 7). Even if Representation *Dong* requires more to build a mental representation (a mean difference in *AFD* superior of 35*ms* with respect to *UML*), this difference is not statistically significant. Moreover, the differences in *ROAF* (0.24) and *ROAFT* (0.28) shows less effort with Representation *Dong*. Values for *CAP* are slightly superior with Representation *Dong* with respect to Representation *UML* (100% with *Dong* versus 83,3% with *UML*).

As Representation *Dong*, Representation *Gamma* requires less effort from subjects than *UML*. However, the value of 50% in *CAP* for *Gamma* prevent us of drawing conclusions. If we look at Fig. 11, we can see again a dispersed variance in the box plots for *Gamma*, particularly for *AFD*. This variance leads us to think that one or both of the mitigating variables could have an influence on the performance of subjects for this representation. Representation *Schauer* has performed slightly better than Representation *UML* in all our metrics. Despite this fact, we do not report significant differences between Representations *Schauer* and *UML*.

In conclusion, the significance tests presented in Table 7 indicate that diagrams with Representation *Dong* require less effort for Task *Composition*. Having all pattern-related information inside the class (same space) in *Dong*, requires less effort

Table 7 Hypotheses testing for representations *Dong*, *Gamma*, *Schauer*, and *UML* on Task *Composition*, diagrams with 15 classes

Diagrams	P-values		
	AFD (ms)	ROAF (#)	ROAFT (ms)
H_{01}: **Dong$_{15}$** vs. **UML$_{15}$**	0.56	0.02	0.090
H_{02}: **Gamma$_{15}$** vs. **UML$_{15}$**	0.42	0.03	0.020
H_{03}: **Schauer$_{15}$** vs. **UML$_{15}$**	0.33	0.01	0.008

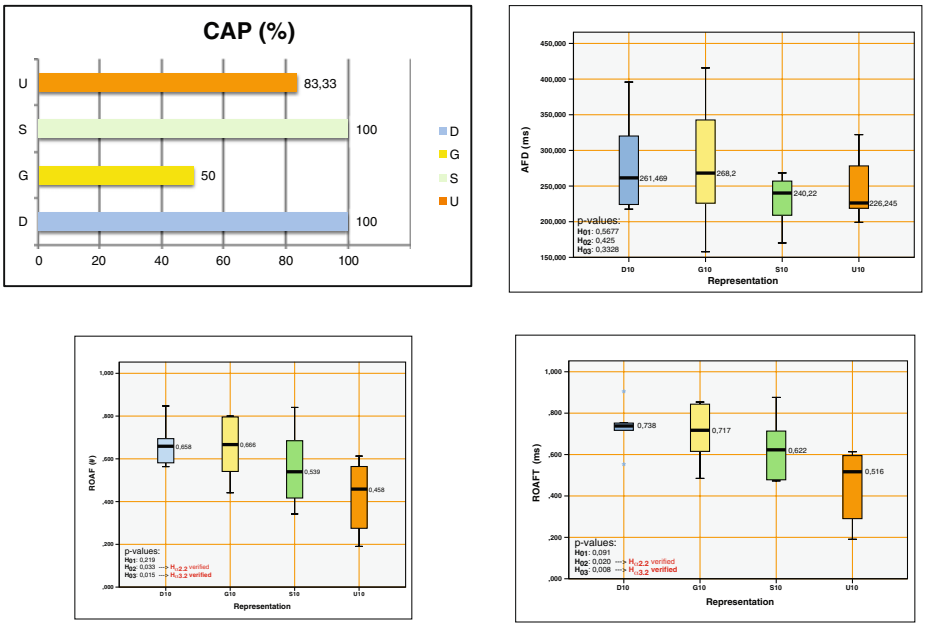

Fig. 11 Data distribution for Task *Composition*, diagrams of 15 classes

from the subjects compared to the effort of visually searching for pattern related information in *UML*.

4.3 Data Analysis of Task *Role*, 15 Classes Diagram

Subjects using Representation *Dong* performed better in *CAP* than using any of the other three representations. Tables 8 and 9 shows that Representation *Dong* requires less effort than Representation *UML*. The difference of roughly 0.22 in *ROAF* and *ROAFT* show less effort with *Dong*. Values in *CAP* show also better performances with *Dong* (100% against 50% for *UML*). Here again, Representation *Dong* is slightly more effort-consuming. Values in AFD for Representation *Dong* shows a mean difference superior of 54*ms* with respect to *UML*. However, this difference is not statistically significant. We had similar performances between users using Representations *Gamma* and *UML*. Figure 12 shows clearly less effort from

Table 8 Means of dependent variable values for Representations *Dong*, *Gamma*, *Schauer*, and *UML* on Task *Role*, diagrams with 15 classes

Diagrams	Perf.	Effort measures		
	CAP (%)	AFD (ms)	ROAF (#)	ROAFT (ms)
Dong$_{15}$	100	327.06	0.73	0.79
Gamma$_{15}$	40	268.43	0.54	0.60
Schauer$_{15}$	0	289.36	0.87	0.87
UML$_{15}$	50	272.85	0.52	0.58

Table 9 Hypotheses testing for Representations *Dong*, *Gamma*, *Schauer*, and *UML* on Task *Role*, diagrams with 15 classes

Diagrams	P-values		
	AFD (ms)	ROAF (#)	ROAFT (ms)
H_{01}: **Dong$_{15}$** vs. **UML$_{15}$**	0.57	0.005	0.004
H_{02}: **Gamma$_{15}$** vs. **UML$_{15}$**	0.91	0.570	0.810
H_{03}: **Schauer$_{15}$** vs. **UML$_{15}$**	0.14	0.010	0.020

subjects using Representation *Schauer* when compared to *UML*. However, we can see a higher effort with Representation *Schauer*. Moreover, all subjects gave wrong or incomplete answers using this representation. Thus, even if we have statistically significant differences in *ROAF* and *ROAFT* when comparing Representation *Schauer* to *UML*, we cannot say that *Schauer* performs better than *UML*, due to poor subject performance in *CAP* when using *Schauer*.

For Task *Role*, we conclude that diagrams using Representation *Dong* requires with statistical significance less effort and have better answer performance than *UML*. Representation *Gamma* showed similar effort than Representation *UML*. Representation *Schauer* showed less effort. However, we conjecture that due to the lack of information in this representation, subjects were giving up faster. Looking at the poor performance of Representation *Schauer*, we conjecture that just showing design patterns with the same structure as described in (Gamma et al. 1998) without any additional information could be a source of confusion for the subjects to really understand the intent of the class participating in a pattern.

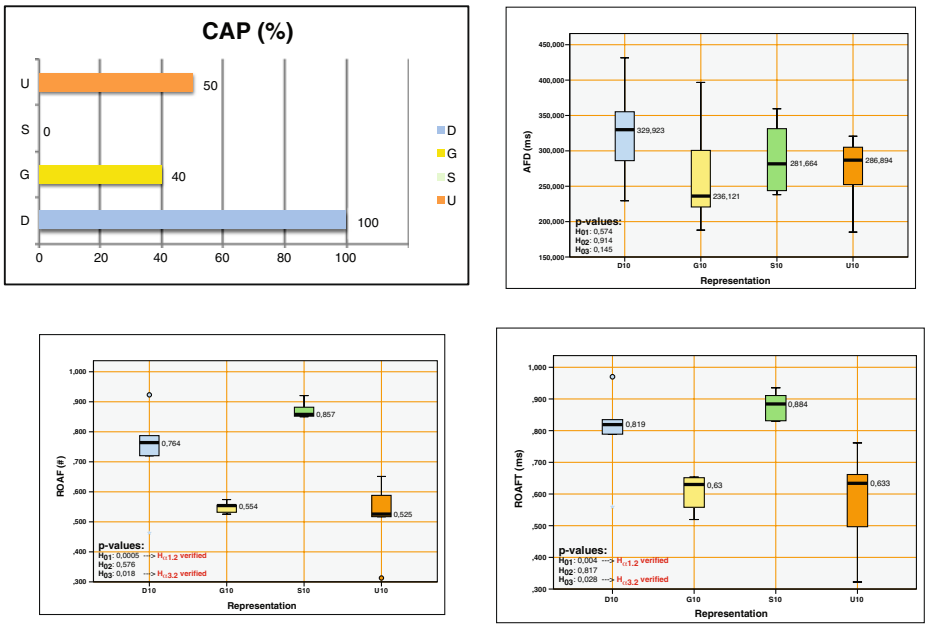

Fig. 12 Data distribution for Task *Role*, diagrams of 15 classes

4.4 Data Analysis of the Impact of Secondary Factors in 15 Classes Diagrams

The results presented in Sections 4.1, 4.2 and 4.3 show:

- For Task *Participation*: a better performance in *CAP* for users using Representation *UML* over all other representations and also less effort in subjects with respect to *Dong*. No statistically significant differences on subjects' effort were reported when comparing *UML* to the other two representations.
- For Task *Composition*: subjects performed well in most representations for *CAP*, despite this fact, only Representation *Dong* requires less effort when compared to *UML*.
- For Task *Role*: Representation *Dong* has better values in *CAP* than the other three representations and was again the only representation to require less effort when compared to *UML*.

Considering the variances in the metrics reported in previous sections, we investigated if the levels of knowledge in design patterns could mitigate the results. We eliminated the mitigating variable of JHotDraw Knowledge because more than 80% of the subjects where classified on the basic level. Therefore, we are sure that this variable will not have an impact on the presented results.

For simplicity reasons and because none of our subjects had knowledge of design patterns, we decided to group design patterns knowledge in two categories: 1 for basic-medium level and 2 for the expert level. We also chose to study only the values of answers given and *ROAF*.

Figure 13 (left) shows the impact of design pattern knowledge on Task *Composition*. With Representation *Gamma*, subjects who have very good design pattern knowledge (level 2) perform better than subjects having basic or average knowledge

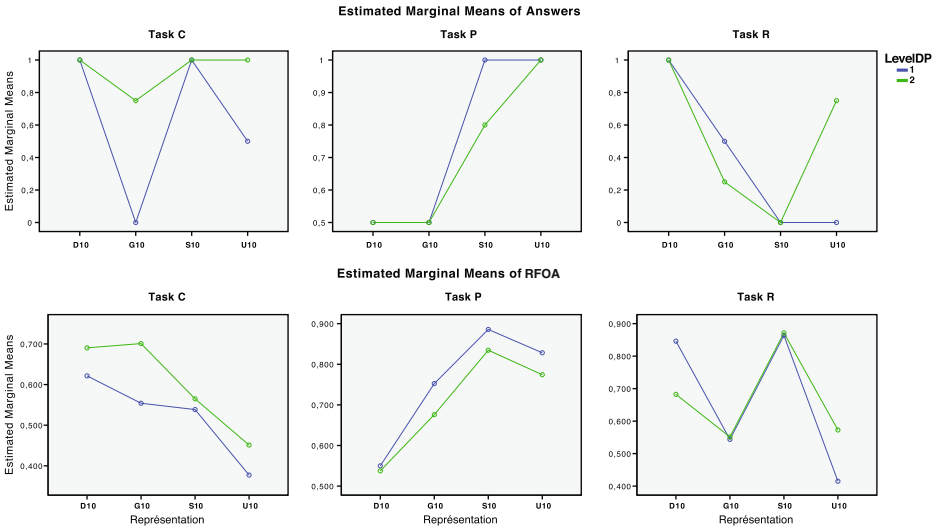

Fig. 13 Combined impact of answers or design pattern knowledge and the different representations used, 15 classes diagram

Table 10 Impact of design patterns knowledge, 15 classes diagram

	Answers (C)	ROAF (C)	Answers (P)	ROAF (P)	Answers (R)	ROAF (R)
DP knowledge	0.02	0.31	0.82	0.21	0.45	0.95
Combined	0.13	0.95	0.98	0.93	0.15	0.06

(level 1). We can see the same tendency for Representation *UML*. This same tendency is present for the metric of *ROAF* for all representations even if is not statistically significant. A 2-way ANOVA test (see Table 10) shows a statistically significant impact of design patterns knowledge only on the correctness of the answers for Task *Composition*. We can see more ore less the same tendency in the correctness of responses for the other two tasks even if there is no statistically significant impact for these two tasks.

For Task *Participation*, we find two interesting situations. The first situation is for Representation *Schauer* in the correctness of answers where novices seems to perform better than experts. The second is in the values for *ROAF* where all experts seem to need more effort than novices when performing the task. These situations could be caused by a deeper analysis made by the experts. Despite the results mentioned before, we report no significant impact of combining representation and design pattern knowledge on subjects' performance.

4.5 Data Analysis of Diagrams of 40 Classes

For Task *Participation* as with 15 classes diagrams, we found that the effort with Representation *Dong* is greater than with *UML*. Representation *Schauer* shows similar performances than *UML*. For tasks *Composition* and *Role*, we report no statistically significant results. Due to readability issues caused by the high class density, we cannot draw further conclusions. The statistical analysis and the descriptive statistics for these diagrams are available at http://url.hidden-for-double-blind.review.

5 Threats to Validity

Following (Wohlin et al. 2000), we identified some threats to the validity of our study and mitigated or accepted them. Some threats are related to the use of human subjects and some others to the use of the equipment.

5.1 Internal Validity

We identified three possible threats to internal validity of our study: *maturation*, *instrumentation*, and *diffusion of the treatments*. To mitigate the maturation threat, we addressed the learning effect by (1) illustrating the representations and the kinds of tasks to be performed by the subjects during the tutorial and (2) showing the tasks to subjects in different orders. Showing tasks in different orders avoids favoring the tasks presented at the end. This random ordering also prevents the fatigue effect that

could disadvantage tasks always given at the end. We also mitigated the fatigue effect with the design of experiment, limiting the subjects' effort to perform the experiment to between 12 and 15 minutes.

The instrumentation threat is related to the use of the equipment. Subjects have to wear a fairly heavy head-band and must minimize head movements to avoid decalibration. Head movements are unavoidable. People tend to move their heads while they are concentrating, causing small coordinate offsets. To deal with this threat, we used a dentist chair and a travel pillow to give support to the subject's neck and head. We also analyzed eye-movement movies recorded during the experiment of each subject to detect any coordinate offset. Coordinates offset can be fixed easily, only simple drift correction is needed to the data to create a coordinates translation. We had some cases of coordinates offset and they were fixed with drift correction. We also identified and accepted one threat to the instrumentation validity, the readability issues with diagrams of 40 classes.

Finally, to prevent subjects to diffuse any information about the study, we asked the subjects not to talk about the study with other people before the end of the study. We are confident that our instructions were followed by the subjects.

5.2 Construct Validity

We addressed four threats to construct validity: *mono-operation bias*, *mono-method biases*, *hypothesis guessing*, and *apprehension*.

We accepted the risk of having a mono-operation bias caused by the utilization of one single system (JHotDraw) in our study. However, regarding the amount of variables to test, we decided to accept the risk instead of increasing the complexity of our study.

The mono-method bias is the risk to have a bias in the measures of our experiments as a consequence of using only a single type of measures. We used four dependent variables, i.e., *CAP*, *ROAF*, *ROAFT*, and *AFD*, and we cross-checked each measure against the others in our analyses to draw conclusions.

In order to avoid hypothesis guessing, we did not inform the subjects about the goal of the study. We just explained them in the tutorial that they had to perform tasks on different UML class diagrams with different design pattern representations.

To prevent the apprehension threat, we first detailed to the subjects the eye-tracker operation and we reassured them about the absence of risks related to infrared emissions directed towards their eyes. Second, subjects were confirmed about the anonymous nature of their answers and their identity. Finally, we decided not to set a predefined time to perform the tasks, instead, we just asked subjects to answer as soon as possible.

5.3 External Validity

Two threats were addressed, *interaction of selection and treatment* and *interaction of setting and treatment*. The issue with the interaction of selection and treatment is to assure that the subjects in this study are representative of software professionals. This issue has been discussed in several studies (e.g., by Briand et al. (2005)). In our study, subjects are graduate students with a good knowledge of UML and the majority of

them has a comparable knowledge of UML software modelling and design patterns to software professionals.

For the interaction of setting and treatment issue, we considered the size and complexity of the used diagrams. We chose and reverse-engineered JHotDraw, a good example of a program that extensively uses design patterns. Diagrams used in the study contains four different design patterns with a maximum class participation density of two design patterns. We presented diagrams containing 15 classes, which enters in the range of recommended number of classes for effective program comprehension activities (Ambler 2005). We also presented diagrams containing 40 classes with the aim of studying the representations on more complex diagrams. We cannot state that our results will apply for all diagrams and for all design patterns. Specific replications in industry settings is needed to draw such conclusions.

5.4 Conclusion Validity

Threats to conclusion validity identified and addressed are: *violated assumptions of statistical tests*, *reliability of measures*, *random irrelevancies in experimental setting*, and *random heterogeneity of subjects*. To prevent violating assumptions of statistical tests, we verified and respected all the assumptions on which the tests used in our analysis relies.

To address the reliability of measures, we chose well-documented measures and we took care of calibrating well the eye-tracker for every subject before collecting data.

Regarding the random irrelevancies in experimental setting, our study was performed in a quiet laboratory without any distraction. We also performed preliminary tests with some other subjects (not included in our study) to detect any other factor that could influence the results.

Finally, considering the choice of subjects, our sample is heterogeneous enough in terms of design pattern knowledge to reflect the target population. Moreover, to avoid subject knowledge from being mainly related to the results, we use the former as a mitigation variable and verified that its impact is less important than the use of different design pattern representations.

6 Conclusion, Discussion, and Future Work

Representing design patterns used in a program ease the understanding of its design and facilitate program comprehension in general. Indeed, a good understanding of pattern-related information is needed to develop and maintain programs efficiently. Currently, a common representation to visualize design patterns is the UML collaboration notation. Previous work highlighted limitations in this representation and proposed alternative representations. However, none of these previous pieces of work made an empirical study to assess whether their representations facilitate more the comprehension of programs than the common one.

We conducted an empirical study to evaluate the efficiency of one set of alternative representations (pattern enhanced class diagrams (Schauer and Keller 1998), stereotype enhanced UML diagrams (Dong et al. 2007), and “pattern:role” notation (Gamma 1996; Vlissides 1998)) compared to the UML collaboration notation.

We designed and performed experiments to collect data to compare subjects's performance while performing three basic tasks (class participation, pattern composition and roles played by class) required for design pattern comprehension using the different representations overlapped on the same UML class diagrams. We used the following design patterns: Composite, Prototype, Template Method, State and Singleton. Design patterns were shown as described in Gamma et al. (1998) with some slight differences due to language implementation issues. We measured performance in terms of correct answers and effort that subjects spend to perform given tasks. We collected effort data using eye trackers on diagrams with small density (15 classes) and with larger density (40 classes).

The analyses showed that stereotype-enhanced UML diagrams (Dong et al. 2007), with their semantic richness, are more efficient for Tasks *Composition* and *Role* than the UML collaboration notation for diagrams of 15 classes. The UML collaboration notation and the pattern-enhanced class diagrams, are more efficient for locating the classes participating in a design pattern (Task *Participation*).

Looking at the poor performance of the pattern enhanced class diagrams in Task *Role*, we may think that just showing design patterns with the same structure as described in Gamma et al. (1998) without any additional information is a source of confusion for the subjects. We also report that 40 class diagrams are difficult to read and, thus, we cannot draw conclusions on the results from these diagrams. Therefore, other experiments are required to confirm our findings and generalize for more design patterns.

Thus, the importance of this empirical study, evaluating the efficiency of representations to visualize design patterns in UML diagrams, is three-fold. First, it provides a framework for comparing current and future notations. We attempted to provide all necessary details to replicate our experiments and–or apply them on other notations, in particular, all the material is accessible on-line at http://url.hidden-for-double-blind.review. Second, it shows that notations have advantages and weaknesses. Therefore, it provides ground to devise new notations that would further overcome identified limitations while combining the best of current notations. Third, the results of this empirical study could be use to motivate tool builders to include different notations for different tasks: for example, tool vendors could decide to include Dong's notation to describe patterns while keeping the UML collaboration notation for other tasks and for locating classes participating in design patterns. Finally, the results could influence educators in choosing to teach to their students different notations, emphasizing that none of the existing notation fits all possible tasks. They could also help them to highlight to their students the importance of carefully investigating notations, even if backed by industrial consortium, such as the UML collaboration notation.

In future work, we will replicate the study with other diagrams and using other design patterns to confirm our observations and to address more threats to the validity. We will conduct scan-paths studies in pattern comprehension activities to compare experts and novices trying to find diagram-reading patterns. We will use the results obtained in this work to propose a new representation and compare its efficiency executing an empirical study. We will also study dynamic-visualization techniques in design pattern comprehension as those proposed by Dong et al. (2007) and Schauer and Keller (1998).

Acknowledgement The authors thank Rocco Olivieto for the fruitful discussions and suggestions.

References

Ambler SW (2005) The elements of UML 2.0 style. Cambridge University Press, Cambridge

Aversano L, Canfora G, Cerulo L, Del Grosso C, Di Penta M (2007) An empirical study on the evolution of design patterns. In: Proceedings of the the 6th European software engineering conference and symposium on the foundations of software engineering. ACM, New York, pp 385–394

Bednarik R, Tukiainen M (2006) An eye-tracking methodology for characterizing program comprehension processes. In: Proceedings of 5th symposium on eye tracking research & applications. ACM, New York, pp 125–132

Bojko A (2005) Eye tracking in user experience testing: how to make the most of it. In: Proceedings of the 14th annual conference of the usability professionals association. Usability Professionals' Association, Bloomingdale

Briand LC, Labiche Y, Di Penta M, Yan-Bondoc H (2005) An experimental investigation of formality in UML-based development. Trans Soft Eng 31(10):833–849

Chabris CF, Kosslyn SM (2005) Representational correspondence as a basic principle of diagram design. In: Knowledge and information visualization. Springer, New York, pp 36–57

Dong J, Yang S, Zhang K (2005) Visdp: a web service for visualizing design patterns on demand. In: Proceedings of the 6th international conference on information technology: coding and computing.

Dong J, Yang S, Zhang K (2007) Visualizing design patterns in their applications and compositions. Trans Soft Eng 33(7):433–453

Duchowski AT (2003) Eye tracking methodology: theory and practice. Springer, New York

Eden AH, Yehudai A, Gil J (1997) Precise specification and automatic application of design patterns. IEEE Computer Society, Piscataway, pp 143–152

Eichelberger H (2003) Nice class diagrams admit good design? In: Proceedings of the 1st symposium on software visualization. ACM, New York, pp 159–ff

Eichelberger H, von Gudenberg JW (2003) Uml class diagrams—state of the art in layout techniques. In: Proceedings of the 1st SOFTVIS workshop on visualizing software for understanding and analysis. ACM, New York, pp 30–34

France RB, Kim D-K, Ghosh S, Song E (2004) A uml-based pattern specification technique. Trans Soft Eng 30(3):193–206

Gamma E (1996) Applying design patterns in Java. Java Rep 1(6):47–53

Gamma E, Helm R, Johnson R, Vlissides J (1998) Design patterns—elements of reusable object-oriented software. Addison-Wesley, Reading

Goldberg JH, Kotval XP (1999) Computer interface evaluation using eye movements: methods and constructs. Int J Ind Ergon 24(6):631–645

Goldberg JH, Stimson MJ, Lewenstein M, Scott N, Wichansky AM (2002) Eye tracking in web search tasks: design implications. In: Proceedings of the 1st symposium on eye tracking research & applications. ACM, New York, pp 51–58

Guan Z, Lee S, Cuddihy E, Ramey J (2006) The validity of the stimulated retrospective think-aloud method as measured by eye tracking. In: Proceedings of the 12th conference on human factors in computing systems, pp 1253–1262

Guhneuc Y-G (2006) Taupe: towards understanding program comprehension. In: Proceedings of 16th IBM center for advanced studies conference. ACM, New York, pp 1–13

Hutchins E (1995) Distributed cognition. MIT, Cambridge

JHotdraw (2007) JHotdraw: a java GUI framework for technical and structured graphics. http://www.jhotdraw.org

Kazman R, Klein M, Barbacci M, Longstaff T, Lispon H, Carriere J (1998) The architecture tradeoff analysis method. In: Proceedings of the 4th international conference on engineering of complex computer systems. IEEE Computer Society, Piscataway, pp 68–78

Khomh F, Guéhéneuc Y-G (2008) Do design patterns impact software quality positively? In: Proceedings of the 12th conference on software maintenance and reengineering. IEEE Computer Society Press, Piscataway

Lauder A, Kent S (1998) Precise visual specification of design patterns. In: Proceedings of the 12th European conference on object-oriented programming. Springer, New York, pp 114–134

Mapelsden D, Hosking J, Grundy J (2002) Design pattern modelling and instantiation using dpml. In: Proceedings of the 14th international conference on tools. Australian Computer Society, Canberra, pp 3–11

Moore P, Flitz C (1993) Gestalt theory and instructional design. J Tech Writ Commun 23(2): 137–157
Object Management Group (1997) Unified modeling language specification, version 1.1. http://www.omg.org
Purchase HC, Carrington DA, Allder J-A (2002) Empirical evaluation of aesthetics-based graph layout. Empir Soft Eng 7(3):233–255
Rayner K (1998) Eye movements in reading and information processing: 20 years of research. Psychol Bull 124(3):372–422
Rich C, Waters RC (1988) The programmer's apprentice. Computer 21(11):10–25
Schauer R, Keller R (1998) Pattern visualization for software comprehension. In: Proceedings of the 6th international workshop on program comprehension. IEEE Computer Society, Piscataway, pp 4–12
Shalloway A, Trott JR (2002) Design patterns explained: a new perspective on object-oriented design. Addison-Wesley, Reading
Soloway E, Pinto J, Letovsky S, Littman D, Lampert R (1988) Designing documentation to compensate for delocalized plans. Commununications 31(11):1259–1267
Sun D, Wong K (2005) On evaluating the layout of uml class diagrams for program comprehension. In: Proceedings of the 13th international workshop on program comprehension. IEEE Computer Society, Piscataway, pp 317–326
Trese T, Tilley S (2007) Documenting software systems with views V: towards visual documentation of design patterns as an aid to program understanding. In: Proceedings of the 25th international conference on design of communication. ACM, New York, pp 103–112
Vlissides J (1998) Notation, notation, notation. C++ Report
von Mayrhauser A, Vans AM (1995) Program comprehension during software maintenance and evolution. IEEE Comput 28(8):44–55
Ware C (2005) Visual queries: the foundation of visual thinking. Springer, New York
Wohlin C, Runeson P, Höst M, Ohlsson MC, Regnell B, Wesslén A (2000) Experimentation in software engineering: an introduction. Kluwer Academic, Dordrecht
Yusuf S, Kagdi H, Maletic JI (2007) Assessing the comprehension of uml class diagrams via eye tracking. In: Proceedings of the 15th international conference on program comprehension. IEEE Computer Society, Piscataway, pp 113–122

Gerardo Cepeda Porras holds a M.Sc. in computer science from Université de Montréal, Canada, since 2008 and an Engineering Diploma from Universidad de las Américas Puebla, Mexico, since 2003. He worked in the Ptidej team on empirical software engineering studies to understand program comprehension using eye-trackers. He is now software analyst and programmer at Giro Inc., a leader in software solutions for public transit and postal distribution.

Yann-Gaël Guéhéneuc is associate professor at the Department of computing and software engineering of Ecole Polytechnique of Montreal where he leads the Ptidej team on evaluating and enhancing the quality of object-oriented programs by promoting the use of patterns, at the language-, design-, or architectural-levels. In 2009, he was awarded the NSERC Research Chair Tier II on Software Patterns and Patterns of Software. He holds a Ph.D. in software engineering from University of Nantes, France (under Professor Pierre Cointe's supervision) since 2003 and an Engineering Diploma from École des Mines of Nantes since 1998. His Ph.D. thesis was funded by Object Technology International, Inc. (now IBM OTI Labs.), where he worked in 1999 and 2000. His research interests are program understanding and program quality during development and maintenance, in particular through the use and the identification of recurring patterns. He was the first to use explanation-based constraint programming in the context of software engineering to identify occurrences of patterns. He is interested also in empirical software engineering; he uses eye-trackers to understand and to develop theories about program comprehension. He has published many papers in international conferences and journals.