Using a Follow-on Survey to Investigate Why Use of the Visitor, Singleton & Facade Patterns is Controversial

Cheng Zhang, David Budgen and Sarah Drummond
School of Engineering & Computing Sciences
Durham University
Durham DH1 3LE, U.K.
{cheng.zhang2;david.budgen;sarah.drummond}@durham.ac.uk

ABSTRACT

Context: A previous study has shown that software developers who are experienced with using design patterns hold some conflicting opinions about three of the more popular design patterns: *Facade*, *Singleton* and *Visitor*.
Aim: To identify the characteristics of these three patterns that have caused them to generate such differing views.
Method: We employed a qualitative follow-on survey of those developers who had taken part in the earlier survey about design patterns.
Results: We received 46 usable responses from a possible total of 188, with nearly 85% of respondents having six or more years of experience with design patterns. Of these, 27 also provided comments and descriptions of experiences about the patterns, which we categorised.
Conclusions: All three patterns can easily be misused and in each case, the consequences of misuse are regarded as being particularly significant.

Categories and Subject Descriptors

D2.2 [**Design Tools and Techniques**]: Object-Oriented Design Methods

Keywords

design pattern; survey; empirical

1. INTRODUCTION

The concept of the *design pattern* forms a well-established and widely advocated mechanism for aiding the OO design process, with the textbook by the 'Gang of Four' (*GoF*) providing a widely-known and much-cited catalogue of patterns [4]. However, relatively little research has been done to identify how effective the concept actually is in practice, and to determine the conditions under which it might be appropriate (or inappropriate) to use specific patterns. In two previous studies we have:

- conducted a *Systematic Literature Review* (SLR) in the form of a mapping study, in order to identify how extensively the use of design patterns have been investigated through the use of empirical studies, and to identify any knowledge about their use and limitations that may have been identified [14];
- undertaken a *survey* of 216 experienced software developers to draw upon their experiences with using the 23 design patterns in the *GoF* [13].

We discuss the outcomes from these studies more fully in the next section. As might be expected, both the experiences and the opinions from the survey were quite varied. We did identify groups of patterns for which there was general agreement about their usefulness or otherwise. However, we also identified three patterns where significant differences emerged in the assessments of their value: *Facade*, *Singleton* and *Visitor*. (We might also note that within the patterns community, *Singleton* and *Visitor* have attracted considerable debate, and that one member of the *GoF* has gone on record to express doubts about the value of *Singleton*[1].)

A problem that was encountered in both studies was that of obtaining *causal* links that could provide a clear derivation of knowledge and opinions from experiences. In our survey we did ask for qualitative comments that could help with this, but this element received relatively few responses.

We therefore decided to investigate these three patterns more fully, to see if we could identify the characteristics that made them controversial. To do so, we conducted a second survey that asked respondents to express views about the characteristics of these three patterns. Since we were seeking to probe deeper into the reasons that lay behind the results of the first survey, the sampling frame used for this survey was the set of respondents to the first survey.

So, for this second survey, our research question was:

> "*What characteristics of these three patterns cause developers to hold widely differing views about their use?*"

A supplementary question we also asked was:

> "*Is a follow-on survey a useful way to investigate these characteristics?*"

Our second survey was conducted in early summer 2011. In the following sections we provide a little more background about the original studies, describe the design of this follow-on survey and its conduct, present our results and discuss how far they answer the research questions .

[1] http://www.informit.com/articles/article.aspx?p=1404056

ESEM'12, September 19-20, 2012, Lund, Sweden.

2. BACKGROUND

This study forms the third in a sequence of three investigations into design patterns. We begin by briefly outlining the nature of a design pattern, and then describe the outcomes from each of the preceding studies that helped to motivate the question for the one described here.

2.1 Studying Design Patterns

The very nature of a design pattern makes it difficult to conduct empirical studies of its use or misuse, at least through the vehicle of conventional laboratory experiments. Essentially a pattern forms a *knowledge schema* through which the expertise of a group of designers can be recorded in such a way that it can be reused by others [2]. Hence the use of a pattern involves a creative activity, during which the pattern is interpreted within the context of the particular design task. This is turn means that for any experimental studies of design patterns, the experience and skills of the participants will form quite significant confounding factors.

There is therefore good reason to study design patterns by using a wide range of empirical study forms: experiments, case studies (especially in an industry setting), and surveys—and then to triangulate between the outcomes of these as fully as possible. We examine this range of options more fully in the discussion that follows.

2.2 The Mapping Study

A mapping study is a form of Systematic Literature Review that has as its research question the issue of determining the range and size of the set of studies addressing a fairly broad topic [6]. One of the aims of such a study is to determine whether there are enough primary studies available to provide scope to undertake an actual SLR with a much more focused research question. Obviously some blurring of the boundaries can occur—as was the case for our own mapping study where we did analyse and interpret the data for some of the patterns in more detail [14].

We conducted a systematic and thorough search over the period 1995-2009, which identified 611 candidate papers, although the number that met our inclusion/exclusion criteria was quite small. The final set for analysis included only 10 papers, describing 11 experimental studies. To augment this, we then re-examined the papers that we had classified as 'observational' (i.e. lacking experimental rigour) and included seven of these that offered reasonably good causal links between the reported observations and the conclusions. All of the studies involved patterns that were catalogued in the *GoF* (which includes 23 patterns in all), but only a sub-set of these were studied, and only three (*Composite*, *Observer* and *Visitor*) had been studied very extensively.

The added value provided by the observational studies, and particularly by one that was very clearly reported [11], suggested to us that a survey of experienced developers could potentially provide a fuller profile of the complete set of patterns from the *GoF*.

2.3 The First Survey

We conducted this over the summer of 2010, using an online data collection form (SurveyMonkey). Our respondents came from three groups.

- The authors of all of the papers that we identified as being about design patterns in our mapping study (not just empirical papers). We sent out 877 invitations, although returned e-mails indicated that only 681 of the addresses were still valid. We refer to this group as the *Authors*.

- People who were recommended by the Authors (our invitation to the Authors did ask them to pass it on to anyone who might have appropriate knowledge). We refer to these respondents as the *Snowball* group.

- Members of a research mail-list that was identified in the course of our survey, termed the *Mail-list* group.

After removing unusable responses, we were left with 206 responses: 128 Authors; 41 Snowball; and 37 Mail-list. Analysis of the demographic questions addressing such issues as years of experience with object-oriented development, and years of experience with patterns indicated that the profiles of the three groups were sufficiently close for them to be analysed as a single dataset. We also examined whether the respondents who categorised themselves as primarily working as developers had significantly different views from those who considered themselves as being primarily researchers and teachers (see below).

Respondents were asked to complete a 'rating' question by providing an assessment of 'usefulness' for all 23 of the *GoF* patterns (there was also an option for indicating 'no experience'). They were then asked to perform a 'ranking' exercise by identifying up to three patterns that they considered to be particularly useful, and also up to three patterns that they did not consider to be useful. As might be expected, there were more positive 'votes' than negative ones. We also provided an option for respondent's to comment on why they considered a pattern to be useful or not useful, and encouraged them to use this option to provide causal reasoning for their choices.

Our findings are reported elsewhere [13], where we review the assessments for all 23 of the patterns. The two patterns that were rated most highly, with relatively few reservations being entered, were *Observer* and *Composite*. At the other end of the spectrum there was a distinct group of patterns that were considered to be of little or no use, most notably *Memento* (with no positive votes at all), and *Flyweight* which had 21 negative votes and only two positive ones.

While the views of the different groups (developers versus the combined group of teachers and researchers) expressed in the *ranking* question were similar for most patterns, *Facade* is one where there was some difference. Developers all viewed it positively, with all of the negative votes coming from teachers and researchers. Developers also tended to be more negative about *Singleton*, although both groups held rather mixed views overall. The responses to the *rating* question were generally consistent with these profiles.

In assessing the validity of the outcomes we do need to recognise that our sample included many people with higher degrees, particularly among the teachers and researchers, and also a large proportion of people who had authored patterns. So this group may not be fully representative of the target population (software developers) for which they acted as a surrogate, and may have a more favourable (or critical) attitude towards design patterns.

Table 1 shows the number of 'votes' cast for the three patterns that we examine in this study. The figures in brackets indicate the percentage of the total of 389 positive and 113 negative votes that were made across the 23 patterns.

Table 1: Votes Cast for Each Pattern

Pattern	Positive Votes	Negative Votes
Visitor	26 (6.7%)	11 (9.7%)
Singleton	22 (5.7%)	14 (12.4%)
Facade	26 (6.7%)	7 (6.2%)
Total	74 (19.1%)	32 (28.2%)

While these three patterns attracted a broadly similar proportion of positive votes to many other patterns, they also attracted a disproportionately large number of negative votes. The qualitative comments also indicated mixed views, especially for *Singleton* and *Visitor* and were used as the basis for our second survey, described in the next section. The purpose of the second survey was to collect qualitative data that could identify the characteristics of these three patterns that led to such contrasting views about their value.

For the reader who may be unfamiliar with these three patterns, we provide a very brief summary of each of them below, to help with interpreting the comments we report.

2.3.1 The Visitor Pattern

This is a behavioural pattern that acts as an intermediary to the services provided by a composite set of objects, translating requests it receives into a set of requests to the objects. Its use makes it possible to add new functionality (for a client of these objects) simply by modifying/enhancing the *Visitor*. In exchange, the ability to modify the composite set of objects is made more difficult by its use, and the *Visitor* has to have knowledge about the 'state vectors' of those objects [3]. There are some implementational issues for *Visitor*, particularly with the use of 'double despatch' for many languages, which "lets visitors request different operations on each class of element" [4]. The use of this, along with the use of polymorphism, significantly complicates testing of systems that employ *Visitor* [1].

2.3.2 The Singleton Pattern

This is a creational pattern that "ensures a class only has one instance", with a single global access point. (A good example of a situation where this might be appropriate is for a spooler object managing a printer.) There are subtle issues with *Singleton* though, see reference [3] for a fuller discussion of these.

2.3.3 The Facade Pattern

Facade is a structural pattern that is intended to "provide a unified interface to a set of interfaces in a subsystem" [4]. Essentially, this can act as 'glue code' to hide individual components that are only of specialist interest and to provide a simple interface for a complex subsystem. *Facade* accepts a request from another object and delegates elements of this to appropriate objects in the subsystem, without the user needing to know that this is being done.

3. METHOD

Following up an initial study with a second, deeper, study that uses the same participants, or a subset of these, is categorised as a *multi-method study*. An example of this is to follow up a largely quantitative survey (such as our first survey) with a set of semi-structured interviews with selected participants, in order to obtain more in-depth explanatory (qualitative) data, as described by Mingers [8]. This approach has the benefit that the researchers retain control of the sampling process—and so are able to identify the most appropriate interview candidates.

The nature of our first survey, where the sampling frame was determined largely by expertise, rather than by location, made the use of semi-structured interviews impractical. We therefore decided to employ a second, more qualitative follow-on survey of the group as a surrogate for using interviews, with its form reflecting this role. As a consequence, we retained little control of the sampling process, as well as being unable to probe and follow through on issues as deeply as would be possible using interviews.

For our follow-on survey, the population of interest (the sampling frame) consisted of those people who had taken part in our first survey. By re-surveying this group, we were able to probe more deeply on the issues of interest, while accepting that we would probably have a much lower number of responses (the norm for surveys is usually quoted as 10-20% [9]). Our first survey did include a question at the end to ask if respondents were willing to take part in any further study on design patterns. 11 respondents declined, leaving us with a sampling frame of 195 experienced pattern users.

In the rest of this section, we describe the design of the follow-on survey, and then report on how it was conducted.

3.1 Design of the Follow-on Survey

Guidelines for conducting semi-structured interviews suggest providing a short explanation of the purpose of the research at the start of an interview [9, 10]. Therefore, to help motivate possible respondents for the follow-on survey, we produced a four-sided 'executive summary' of the key outcomes from the first survey, extracted from the full report provided in reference [13]. This summary provided brief details of the demographic profiles and presented a chart showing the votes 'cast' in favour of, or against, the usefulness of each pattern in response to the ranking question. The purpose of the summary was to provide a context for the specific questions asked in the follow-on survey, and to show why these three patterns were considered as being of particular interest. A copy of the summary was attached to the e-mails sent to the respondents from the first survey who had indicated a willingness to take part in further study, inviting them to complete this further survey.

We again adopted an on-line model for applying the survey instrument, but since we aware that any potential respondent had already spent quite substantial time filling in the first survey, we aimed to keep this as short and focussed as possible. We therefore confined the demographic element to one question asking about length of experience with working with design patterns. Our reasons for doing so were two-fold: firstly, we were concerned that asking respondents to provide such material again would deter them from completing the survey, as they had already entered it once; and secondly, we expected to be able to retrieve much of this information from the data of the first survey, by matching the internet addresses used by the respondents to those recorded when they completed the first survey.

To provide a quantitative element we sought to determine how much consensus there was about the set of statements describing the characteristics of each pattern that we had extracted from the first survey. So, the respondents were first presented with each statement and asked to express a level

Table 2: Statements About Characteristics Used in the Questions for Each Pattern

Pattern	Statement provided from the first survey
Visitor	V1. "We conduct different analyses on Abstract Syntax Trees. *Visitor* can provide a unique way to do them and operations are completely independent."
Visitor	V2. "The ability to create many visitors for the same data model. This is very useful for web development. This combined with the MVC pattern means that we can create a lot of views with ease. For example, we have an HTML table printer, csv table printer etc., for the same table of information."
Visitor	V3. "The visitor is very useful in the context of language processors. I have used it primarily to support AST/ASG traversals. The time and effort involved in modifying a visitor hierarchy can be prohibitive but these factors are balanced by its nature suitability for tree/graph traversals."
Visitor	V4. "I prefer to use multiple despatch, however in systems with multiple dispatch you have to emulate it with a visitor. The resulting code using visitor is easy to get wrong, hard to maintain, and difficult to understand.
Visitor	V5. "Only useful to ship data structures and algorithm separately. But then, you have to fix either a set of data structures or a set of algorithms (depending on who visits who). So it can be a pain to manage."
Visitor	V6. "To avoid procedural dependencies of conditionals, I prefer to use idioms that utilize polymorphism to resolve the state of conditionals and the message to send as a consequence of that state. The visitor pattern requires too much awareness of handshaking to be practical. Supporting implementation details need to be invisible so they don't distract from the focus of the role being designed, and provide less opportunity for defects to be injected into the system."
Singleton	S1. "Singleton is natural for use in logging libraries and to maintain user configurations. It provides the convenience of a global variable (without the dirty feeling)."
Singleton	S2. Singleton is a basic pattern, but still it needs some discipline. In code analysis, global entities (e.g. symbol table) may need to have a single instance."
Singleton	S3. "I rarely have need of a Singleton as most domain objects are not unique. In maintenance, this is often used to hold global variables."
Singleton	S4. "Singleton is more an anti-patten and introduces global state."
Singleton	S5. "This pattern introduces 'temporal coupling' (the worst kind). I NEVER use this pattern. The last time I saw it and had to deal with it was 5 years ago and it was extremely painful to retrofit unit tests in that project, because of the Singletons."
Facade	F1. "Facade helps defining a proper architecture for a distributed system, by enforcing high cohesion and low coupling."
Facade	F2. "Make a good contract between lower level of application and upper level of application."
Facade	F3. "Nice pattern to help us deal with existing, problematic code. For third-party code, it enables us to rearrange (group, split, ...) a broken API so that it resembles what we want/need. For in-house legacy code, it's a way to extensively introduce tests by hiding dark sides of the existing implementation using an ideal interface."
Facade	F4. "Acts as a gateway between a subsystem and the rest of the system. Huge fan-in and fan-out, increasing coupling between parts of the system."
Facade	F5. "Used as simple passthrough classes rather than to actually make the life of a subsystem user any easier. Coupling is reduced but with no value added and at the expense of another class to maintain."

of agreement with it, based on a three-point scale (agree, no opinion, disagree). The set of statements are shown in Table 2. (We should note that each one was provided by a different respondent from the first survey.)

The main purpose of the second survey was to collect qualitative data that could identify the characteristics of these three patterns that led to such contrasting views. So, for each pattern we also invited the respondents to:

- add their own observations about that pattern, and any thoughts about why it attracts conflicting views;
- provide any examples of good or bad use of that pattern, based on their own experiences.

3.2 Conduct of the Follow-on Survey

Since the population to be surveyed was small, we did not attempt any sampling within this, and sent our request to all of the 195 respondents from the first survey who had indicated willingness to help further, providing them with the executive summary and inviting them to participate in the second survey. We also provided a follow-up message a few weeks later, as this is generally recommended as good practice with surveys, and one that can help improve the response rate. Seven addresses we used were no longer valid, reducing the size of the population to 188. We received 48 responses. One person completed only the first (demographic) question, while another answered 'no opinion' to all questions, and offered no comments. After excluding these, we were left with 46 responses, corresponding to a response rate of 24.5%.

No issues arose while conducting the survey.

4. RESULTS

4.1 Profile of the Respondents

The only demographic question included was about the degree of experience that our respondents had with using design patterns in general. The responses to this are shown in Table 3. From this we can see that 84.8% of respondents had six or more years of experience of working with patterns. (In the first survey, 67.4% of respondents had six or more years of experience with patterns.)

Table 3: Profile of Respondents' Experience with Using Design Patterns

Years of Experience	Frequency	Percentage
11 to 15 years	20	43.5
6 to 10 years	19	41.3
3 to 5 years	7	15.2
Total	46	100.0

We had originally expected to be able to undertake fuller profiling by matching up responses with those from the first survey, using the respondents' internet addresses, but because so many of these were allocated dynamically, this unfortunately proved to be impractical.

4.2 The Quantitative Responses

As noted above, these responses provided a measure of how far the respondents agreed with the statements selected from the original survey. A few respondents did not provide answers for a small number of these, and these 'null entries' were omitted from the analysis rather than treating them as if they were 'no opinon'. We examine the profile of responses for each pattern in turn.

4.2.1 The Visitor Pattern

Figure 1 shows the profile of the responses to the six statements about the *Visitor* pattern. Only three respondents

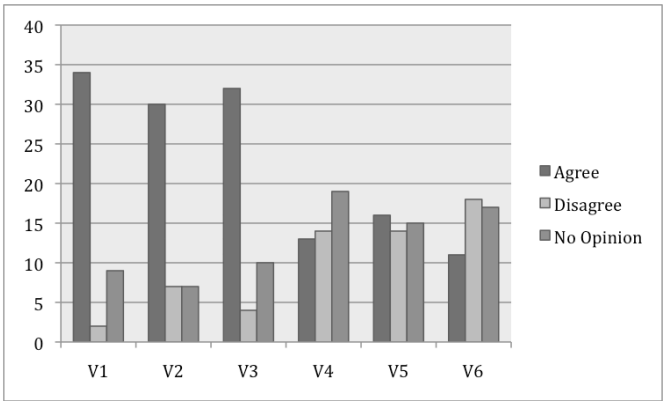

Figure 1: Responses to the Statements About Visitor

expressed 'no opinion' for all six statements. Two of these had 6–10 years of experience, the third had 3–5 years. Since V1–V3 are positive statements about the use of *Visitor*, and the overall profile for this pattern in the first survey was positive rather than negative, it is perhaps not surprising that these are well supported, with relatively few negative responses. Equally, V4–V6 were statements about the disadvantages of using *Visitor*, and here we see a much lower level of general agreement, with all three receiving more or less equal responses for each option.

4.2.2 The Singleton Pattern

The profile of responses for *Singleton* is shown in Figure 2. Only one respondent (with 6–10 years experience) expressed 'no opinion' for all five statements. S1–S2 are fairly

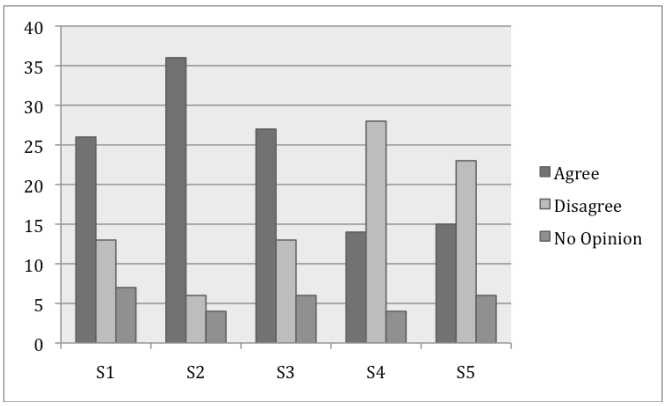

Figure 2: Responses to the Statements About Singleton

positive statements, S3 and S4 express different views about the use of *Singleton* to hold global data (regarded by many as misuse, a view expressed in S4, while S3 is more neutral), while S5 is decidedly negative about the pattern. Perhaps the most interesting answers are those to S3 and S4, which seem to imply that many respondents seem to consider the use of *Singleton* for holding global data as being an acceptable role.

4.2.3 The Facade Pattern

The profile of responses for *Facade* is shown in Figure 3. For this pattern, there were two respondents (both with 6–

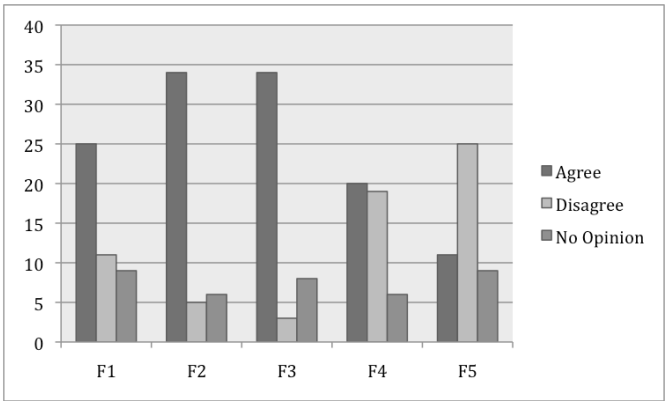

Figure 3: Responses to the Statements About Facade

10 years experience) who expressed 'no opinion' for all five statements and one (with 11-15 years) who did not respond to the questions about this pattern. Again, we see general agreement with the positive views expressed in F1–F3. Statements F4 and F5 express rather contradictory views about its effect upon coupling, and the responses show much more support for the view that coupling is increased by the use of Facade (F4).

4.3 The Qualitative Responses

We received a spread of comments and descriptions of experiences for each pattern. Table 4 shows the distribution

Table 4: Distribution of Qualitative Responses by Years of Experience

No. responses provided	Years of Experience			Total
	3-5 years	6-10 years	11-15 years	
6	1	4	7	12
5	0	1	2	3
4	0	2	0	2
3	0	0	4	4
2	0	1	0	1
1	1	2	2	5
0	5	9	5	19
Total	7	19	20	46

of the number of responses per respondent against years of experience (these were relatively evenly spread across the three patterns). From this, we can see that only 27 of the 46 respondents provided any qualitative responses at all. With one exception, it was also those with more experience who provided these responses.

When considering how to analyse these, we noted that, as observed by Seaman [10], there is "little guidance in the literature for the intellectual process of finding patterns and trends in qualitative data". For this study, we had six distinct datasets ('comments' and 'experiences' for each of the three patterns), and three reviewers. We adopted the following process for analysing each of the datasets.

1. All three of us individually read the responses and then wrote a short list of common issues (the categories).
2. We then met and merged our lists, generating an agreed set of short statements to describe each category.
3. Each of us independently coded the responses using the agreed set of categories.
4. Finally, we met and reviewed the merged codings, discussing any for which there were different interpretations, or where there are not at least two coders in agreement. Where necessary, we revised the wording of the statements describing the categories.

The process of discussion involved in steps 2 and 4 above proved to be a very important element. Responses often addressed multiple issues, and hence required us to assign multiple codes to them. Sometimes an issue was addressed indirectly (as might occur when reference was made to the comments provided from the first survey), and sometimes the structure of a response could be difficult to parse. A few were also considered to offer no clear or usable data and were discarded (see Table 5 for the distribution of these). For some responses, the distinction between a 'comment' (upon the statements provided) and an 'experience' (derived from the respondent's own use of the pattern) could also be indistinct, although we did manage to create separate lists for each.

The above process worked satisfactorily for all but the 'comments' received for the *Singleton* pattern. The initial set of statements we produced proved to be difficult to use. They were redrafted by one of us (DB), after which we successfully repeated steps 3 and 4 for the relevant responses.

To provide a check on this process we calculated *Kappa* (κ) values to provide an assessment of inter-rater consistency. As we had revised the *Visitor* categories during discussion, we omitted that group of ratings, and for simplicity,

Table 5: Distribution of Discarded Responses

Pattern	Category	Discards
Visitor	Comments	6
Visitor	Experiences	0
Singleton	Comments	6
Singleton	Experiences	2
Facade	Comments	4
Facade	Experiences	3

Table 6: Kappa Values for Inter-rater Agreement

Pattern	Form of response	Kappa value	Interpretation
Visitor	Experiences	0.64	Good
Singleton	Comments	0.69	Good
Singleton	Experiences	0.76	Good
Facade	Comments	0.68	Good
Facade	Experiences	0.45	Moderate

we also calculated the level of agreement between the two more experienced coders (DB and SD). The κ values for the remaining five sets of responses are shown in Table 6, and as can be seen, we generally achieved a good level of agreement in our coding.

In the rest of this section we review each set of responses.

4.3.1 Responses to the Comments for Visitor

We began by using a set of four issues, but when we discussed the coding these were restructured and extended into a set of seven. The responses provided good support for all of these. We have listed them in Table 7, and for each comment, the associated count indicates the number of responses related to that issue. (Since we were asking why the responses for the first survey were mixed, it is perhaps not surprising that these tended to emphasise any problems with the use of Visitor.)

4.3.2 Experiences with Visitor

Our original set of agreed statements identified four positive and four negative issues. However, in coding, we made little use of one of the positive issues and hence dropped this from the set, on the basis that the concepts concerned were adequately addressed by the other three. Table 8 summarises these.

As might be expected, these largely reflect the same issues that were identified in the original comments.

4.3.3 Responses to the Comments for Singleton

As noted above, our first attempt at coding these proved difficult to use and so we adopted a revised set of wordings

Table 7: Issues Identified in Comments About the Visitor Pattern

Code	Categories	Responses
VC1	Very specific and constrained application domain	5
VC2	Complexity impedes understanding and use	6
VC3	Structure constrains design options	2
VC4	Has a negative effect upon maintenance	4
VC5	Can increase encapsulation and abstraction	3
VC6	Complicated implementation	3
VC7	There may be better solutions	4

Table 8: Issues Identified in Experiences With the Visitor Pattern

Code	Categories	Responses
	Positive Issues	
VE1	Aids creation/use of data structures	4
VE2	Easy to extend functionality	3
VE3	Provides for good abstraction/separation of concerns	5
	Negative Issues	
VE4	Hard to change/extend structures	5
VE5	Complex to understand/use	6
VE6	Open to misuse in the wrong situation	3
VE7	Complication of use of double despatch in some languages	4

Table 9: Issues Identified in Comments About the Singleton Pattern

Code	Categories	Responses
SC1	Gets employed to provide global variables	8
SC2	Easy to understand but requires discipline in use	6
SC3	Needs care with implementation	3
SC4	Complicates testing and maintenance	4
SC5	OK for shared constants, but not for mutable data	2

for the categories, after which coding proceeded without any real problems. Table 9 shows these.

4.3.4 Experiences with Singleton

Again, although we originally identified eight categories (three positive, five negative), after coding and discussion, we decided that three of these (one positive, two negative) could either be subsumed within the others or was of too limited a scope to be worth retaining (i.e. only one respondent raised that issue). Our summary of these is given in Table 10.

4.3.5 Responses to the Comments for Facade

Facade attracted quite a number of comments although clearly, the respondents felt less strongly about this pattern than about the other two. Table 11 provides a summary of these. Although we originally identified six candidate categories, only three of these were addressed by more than one comment. These were noticeably less negative than those for the other two patterns.

4.3.6 Experiences with Facade

Again, we identified fewer categories for this pattern, as illustrated by Table 12. There were only three issues that were raised by more than one respondent.

Table 10: Issues Identified in Experiences With the Singleton Pattern

Code	Categories	Responses
	Positive Issues	
SE1	Useful for a limited role	4
SE2	Separates class and instance in the design	2
	Negative Issues	
SE3	Complicates testing	5
SE4	Gets misused to provide global variables	3
SE5	Increases maintenance/coupling	2

Table 11: Issues Identified in Comments About the Facade Pattern

Code	Categories	Responses
FC1	Concept of Facade is commonly misused or wrongly designed	7
FC2	Useful for integrating legacy code	4
FC3	Its use can reduce coupling	3

Table 12: Issues Identified in Experiences With the Facade Pattern

Code	Categories	Responses
	Positive Issues	
FE1	Means of encapsulating legacy code/data	3
FE2	Benefit of providing good abstraction	7
	Negative Issues	
FE3	Hard to maintain, loses structure	2

5. RELATED WORK

A survey about the effect of design patterns upon software quality that was conducted in 2007 by Khomh and Guéhéneuc is reported in [5]. They report upon the views of 20 respondents when asked about a set of ten quality attributes. In terms of comparison with this study, their paper reports in detail on only three patterns, none of which are those discussed here. They also provide a summary of the views about three attributes (expandability, understandability and reusability) for all patterns. Overall, we found little scope for any direct comparison with the our own results.

A recent paper by Williams [12] describes the use of a follow-on survey on a software engineering topic, which in this case is to probe views about agile development. However, this paper focuses on reporting the outcomes of the study, with no real discussion about the methodological aspects.

6. DISCUSSION

We first assess the threats to validity and then consider the implications for each pattern that arise from the results. Finally, we review our experiences from using a follow-on survey as our research approach.

6.1 Threats to Validity—the Survey

For a survey, two key issues that we need to consider are the design of the survey instrument itself and our sampling of the population.

Kitchenham and Pfleeger have identified a number of possible validity issues that can occur with a survey instrument [7]. For the first survey we were particularly concerned with *content validity*, and subjected our instrument to a review process to ensure that its questions were relevant. For a follow-on survey such as this, and particularly one that has the form of asking respondents whether or not they agree with a number of statements, this is less of an issue, especially as the statements were not designed by us, but extracted from the first survey. We therefore did not undertake a formal review process, and while the questions were simple, our failure to do so could be considered as a shortcoming. Overall though, we were unable to identify much guidance on how this type of second probe might be designed and evaluated. (In reference [8] the author notes that multi-method studies are not particularly common, even in

Information Systems research, which also seems to apply to follow-on studies of this form.)

Our survey population was well-defined and tightly constrained, with no sampling being needed (we simply mailed to all of the original respondents). As noted above, we were unable to profile the respondents to this survey as well as we had expected. However, having already established that the population formed a reasonably consistent group, and having obtained some limited data about the demographics pertaining to our sample, we have no reason to believe that it is unrepresentative of the population. However, there is one small caveat here, in that in reporting on our original survey [13], we did recognise that our sample in the first survey might not be wholly representative of the wider community of software developers who employ design patterns in any way. We should also note that although we received 46 responses, only 27 of these (59%) actually provided the qualitative information which formed the main reason for conducting the follow-on survey.

It is also possible that respondents may have been commenting upon statements that they had themselves provided in the first survey. However, as each of the statements we used was provided by a different person, it seems unlikely that this would create any significant bias.

6.2 Threats to Validity—the Analysis

The analysis process for a qualitative form of survey such as this also constitutes a potential threat to validity, not least because the open nature of the questions led to a wide range of responses. Our approach to analysis was obviously open to possible failings in both our *selection* of the categories and also our process of *assignment* to them.

As reported above, selection of the categories was a shared process, and when we reviewed the outcomes of our assignments to these, we did revise the categories where necessary. For the experience responses we actually reduced the number of categories used for each pattern (as a reminder, we did not retain any category with fewer than two responses assigned to it). For the comments, we extended the set of categories for *Visitor*, revised it for *Singleton* and reduced it for *Facade*. For all three, we were able to come to a good level of agreement about these assignments.

For assignment to categories we used a mix of independent coding and then discussion—with the discussion ensuring that we came to a shared interpretation of the more delphic responses (interpretation of these was often the reason why different coders might assign them to different categories). While it is difficult to assign a clear measure for the effectiveness of this process, we would observe that all of us did modify our position over various interpretations. (There were 22 cases where two of us changed our position to agree with the third. For 15 of these, the most experienced researcher in the team (DB) began in the minority position, while the remaining 7 were apportioned as 5 (CZ) and 2 (SAD).) In addition, as indicated in Table 6, we did usually achieve a good level of inter-rater agreement in our initial coding of responses.

6.3 Implications for the Three Patterns

For each pattern we have examined the statements taken from the original survey, as well as the quantitative and qualitative responses to this survey in order to seek an answer to our research question(s). We address these for each pattern in turn below.

6.3.1 *Visitor*

For the first survey, this was very much the 'middle case' among these three patterns. In terms of our main research question, the quantitative responses from this second survey showed good agreement with statements V1–V3 (the 'role' of the pattern), but a much wider spread for the other three (which tended to focus upon practical problems that could arise). If we turn to the qualitative responses, then similarly, the positive qualities identified are largely *design*-related (VC5, VE1, VE2, VE3), while many of the reservations about the pattern either relate to its *implementation* (VC6, VE4, VE7) or to its longer-term effects in terms of constraining design options (VC2, VC3, VC4, VC7, VE4, VE5). In contrast, we might note that the textbook by the *GoF* does recognise some implementation-related limitations, but does not discuss any design or maintenance issues related to its use.

The comments themselves have a strong flavour of rather conditional support for its use, as in the quotations below, which emphasise that, while useful, it needs to be viewed as a means to an end and one that is only really applicable to a limited range of situations.

- "I suspect that the issue is that there are sometimes other (perhaps better) ways to solve the 'visitation' problem."
- "Visitors are useful in languages that miss a language mechanism to extend existing encapsulations (such as Java)."
- "The visitor patterns applies to a relatively small number of supposed 'recurring problems' in OO design."
- "It is a pattern with limited application."
- "The visitor pattern tackles occasions when data structures drive the processes."
- "In programming languages without multiple despatch, visitor is the only way to achieve something similar."

Clearly too, the use of *Visitor* comes at a price, as indicated by the following.

- "Enforces a rigid view of algorithms and data structures."
- "For structures with complex relationships, or with elements of varying types, interfaces can likewise become cumbersome to design and implement."
- "Double despatch implementation can be cumbersome."
- "This is a complicated pattern that has its room in our toolbox, but [we] need to resist using it sometimes."
- "Poor: complexity, contorted data flow."

So maybe the comment from one respondent to the effect that "Visitor is useful when nothing else will do the job, but better avoided otherwise" sums up this set of comments well. Many of our respondents had clearly applied *Visitor* effectively, but were well aware that its use could imply substantial design trade-offs.

What therefore can we conclude from this? For both surveys, the responses clearly indicate that it is valued but that its use should carry a 'health warning', in that used outside of a well-constrained context, it is likely to increase complexity and complicate implementation, testing and maintenance. Essentially then, there is a clear message that this is a pattern where the price of misuse is particularly high.

6.3.2 Singleton

As indicated by its scores in Table 1, in the first survey this pattern appears to have polarised the views of our respondants more strongly than any other. This is reflected in the quantitative responses to the second survey, with few respondents having 'no opinion' and rather mixed responses to S3 and S4, where its use for global variables is raised. Turning to the qualitative responses, while there were some caveats about implementation issues and their consequences (see SC3, SC4, SE3, SE4 and SE5), the key concern seemed to be about exercising discipline when using it (SC2, SC3, SE1), a term also used in S2 (from Table 2) of course. While this might be reasonably be regarded as an issue for any pattern, in the case of *Singleton* there seems to be significantly different ideas about what that discipline should encompass, particularly with regard to global structures.

(The *GoF* textbook only discusses what they interpret as positive consequences of its use, but users seem to have more reservations, particularly regarding global structures.)

One respondent (bravely) stated that: "I quite like Singleton and have used it often in my development work...ensuring one and only one instance of queue, monitor, factor etc.". However, the comment that "Singleton solves a very specific problem with about as much grace as can be expected" is probably closer to a median view (if such a position is possible for *Singleton*). It was noticeable that the entries under the experiences of respondents cited only a very limited set of examples, to the extent that some really doubted that this really was addressing enough of a recurring problem to be classified as a pattern.

So, returning to our supplementary research question, we would suggest that if this pattern belongs in the toolbox at all, then it should be kept in a very specialist section. Unfortunately, the relatively simplicity of *Singleton* makes it a rather convenient example to use when teaching about patterns, but here there is a clear message that we should avoid using it as an example. Indeed, as one respondent observed: "the globals debate is never-ending", and maybe that debate is one that is best left to the experts.

6.3.3 Facade

While Table 1 indicates that this is the lest controversial of the three patterns considered in our study, it still attracted a significant 'negative' vote in the first survey. (But, as noted earlier, the 'negative' votes were all from teachers and researchers.) The quantitative responses showed general agreement with the original comments that dealt with 'role', but less so with those that dealt with consequences (there was particularly little support for F5). From the qualitative comments, as FC1 indicates, there was some feeling in this survey that the negative perceptions expressed in the first survey might arise at least partly from misuse, as well as a clear indication that it was particularly valuable for integrating legacy elements (FC2, FE1).

The individual comments about this pattern show rather less emotion than was engendered by the other two, and a number of respondents suggested that the negative views in the original list probably stemmed largely from a lack of understanding. Examples of positive views were:

- "Easy to misuse, I use it as a high level API, nothing else".
- "Reading the negative comments I get the feeling that the 'Facades' discussed are poorly designed."
- "I don't have bad experiences. However it is easy to use the patten inappropriately, but that is not an issue of the pattern, but of the designers using it".

The experiences quoted tended to be split between its use for legacy code ("when dealing with legacy systems...very useful to extract a cleaner interface" and "if it provides a coherent interface to a sub-system whose structure is then hidden, it has great value"), and its value in actual design ("simplified API for complex back-end systems to hide their details", "front controllers that serve as facades from the presentation to the application layer") — while accepting a possible cost in terms of maintenance ("gets hard to maintain after a while"). Again, if we consult the 'consequences' listed in the *GoF* text, the emphasis is placed upon the implementation benefits, and while users seem to agree about these, they also have some reservations about longer-term effects.

There are clearly occasions when Facade is appropriate, as indicated above (integrating legacy code, providing a good abstraction from complex structures). Indeed, the main concerns expressed are about the ease of misuse and the possible complications for maintenance. Again, while misuse can occur for any pattern, the consequences for Facade may be more significant than for many other patterns—although this may also reflect its relatively widespread use.

6.4 Use of a Follow-on Survey

To our knowledge, multi-method research has so far not formed one of the tools for empirical software engineering. Indeed, as indicated by Mingers in [8], it is relatively uncommon to find it used in IS research too. Although the use of a follow-on survey is not strictly a multi-method form, because we were using the follow-on survey as surrogate for interviews, our study could be considered to have been an approximation to that form.

If we return to our supplementary research question, this asks whether the use of a follow-on survey has successfully enhanced our knowledge about these three patterns, and helped answer the question posed in the title of the paper. We would argue that it has done so in two ways.

1. By helping identify those responses from the first survey that did not address a general issue. (A good example is statement S5 from Table 2. The respondents to the second survey did not consider temporal coupling to be of general concern.)
2. By providing reinforcement for issues that were considered important, and in some cases, clarification of what was important about them (such as for *Facade*).

Overall therefore, we would argue that this study has demonstrated the value of using a follow-on study to probe into the underlying rationale for any outcomes. Equally, we do recognise that in this case it was only possible because we were fortunate enough to have a relatively large number of responses

to the first survey, without which, a more resource-intensive form such as semi-structured interviews would probably be the only option. Also, we cannot readily assess the effectiveness of using a follow-on survey as a surrogate for interviews when we lack any comparison with a similar study making use of (say) semi-structured interviews. Indeed, one of the limitations of using a follow-on survey was that it was not possible to probe as effectively as would be possible with an interview, as was demonstrated by the proportion of respondents who failed to provide any qualitative responses.

So while our answer to the supplementary question "is a follow-on survey a useful way to investigate these characteristics?" is positive, we do have to acknowledge that conducting a second survey is probably not the most effective way to follow up on a survey—although as here, it may well be the only pragmatic choice.

7. CONCLUSIONS

Design patterns form a useful element in the software designer's toolbox, by providing schemas that allow the exchange of design experience and knowledge. However, as demonstrated in our first survey [13], not all patterns from the *GoF* are considered to be of equal usefulness, and a designer needs to be aware of the consequences of using specific patterns. It is also noticeable that, whereas the *GoF* text focuses largely on implementation consequences, our respondents tended to be more concerned about consequential issues for such activities as maintenance and testing.

As discussed above, our study has demonstrated the value of performing a qualitative follow-on study to help explain the outcomes from a more quantitative study. This is particularly so for a topic such as this, since any form of knowledge schema is inevitably going to be difficult to assess and evaluate.

The title of our paper asks why the use of these three patterns is controversial, and the outcomes from this second study have provided some further insight into this.

- In the case of *Visitor*, there is a clear message about the consequences of using it outside of a limited context, with inappropriate use leading to negative effects for implementation, testing and maintenance for the reasons provided in our analyses of comments and experiences.
- For *Singleton*, the messages about this pattern are rather polarised as far as the acceptability of global structures is concerned, but taking the issues as a whole, it seems reasonable to recommend that, at the least, the use of this pattern as a teaching example should be avoided.
- While the use of *Facade* is less controversial, and although it can perform a valuable role for such purposes as integrating legacy code, there are still significant negative issues about its use, such as its effect upon maintenance, that the designer needs to consider.

What is important about this use of follow-on survey is that, through its use, we are able to provide a stronger rationale for making the above summaries, based upon the refinements to the evidence provided by the wider set of experts that we surveyed. So, while we cannot make definitive recommendations about when or when not to use one of these patterns, something that is probably not possible for any context, our survey does provide some further clarification about the situations where they could or should not be employed.

Acknowledgment

The authors would like to thank those respondents from the original survey who were kind enough to give further help by participating in this second survey. We also thank Gordon Rugg for suggesting the use of a multi-method approach.

8. REFERENCES

[1] B. Baudry, Y. L. Traon, G. Sunyé, and J.-M. Jézéquel. Measuring and improving design patterns testability. In *Proceedings of 9th International Software Metrics Symposium (METRICS'03)*, pages 50–59, 2003.

[2] F. Détienne. *Software Design – Cognitive Aspects*. Springer Practitioner Series, 2002.

[3] E. Freeman and E. Freeman. *Head First Design Patterns*. O'Reilly, 2004.

[4] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, 1995.

[5] F. Khomh and Y.-G. Guéhéneuc. Do design patterns impact software quality positively? In *Proceedings of CSMR 2008*, pages 274–278, 2008.

[6] B. A. Kitchenham, D. Budgen, and O. P. Brereton. Using mapping studies as the basis for further research—a participant-observer case study. *Information & Software Technology*, 53(4):638–651, 2011. Special section from EASE 2010.

[7] B. A. Kitchenham and S. L. Pfleeger. Principles of survey research part 4: Questionnaire evaluation. *ACM Software Engineering Notes*, 27(3):20–23, May 2002.

[8] J. Mingers. The paucity of multimethod research: a review of the information systems literature. *Information Systems Journal*, 13(3):233–249, 2003.

[9] B. Oates. *Researching Information Systems and Computing*. SAGE, 2006.

[10] C. B. Seaman. Qualitative methods in empirical studies of software engineering. *IEEE Transactions on Software Engineering*, 25(4):557–572, 1999.

[11] P. Wendorff. Assessment of design patterns during software reengineering: Lessons learned from a large commercial project. In *Proceedings of 5th European Conference on Software Maintenance and Reengineering (CSMR'01)*, pages 77–84. IEEE Computer Society Press, 2001.

[12] L. Williams. What agile teams think of agile principles. *Communications of the ACM*, 55(4):71–76, 2012.

[13] C. Zhang and D. Budgen. A survey of experienced user perceptions about design patterns. Submitted for publication., 2011.

[14] C. Zhang and D. Budgen. What do we know about the effectiveness of software design patterns? Accepted for publication in IEEE Transactions on Software Engineering, 2011.