Editor: **Tore Dybå**
SINTEF
tore.dyba@sintef.no

Editor: **Helen Sharp**
The Open University, London
h.c.sharp@open.ac.uk

Design Patterns: Magic or Myth?

David Budgen

My name is John Wellington Wells,
I'm a dealer in magic and spells,
In blessings and curses,
And ever-fill'd purses,
In prophecies, witches and knells!

—The Sorcerer, *Gilbert and Sullivan, 1877*

AN IMPORTANT ELEMENT of this early Gilbert and Sullivan comic opera was the portrayal of the sorcerer as a domestic salesman rather than some exotic and mysterious being. It was the wares that were exotic, not the dispenser.

Unfortunately, this is just what so many managers and software developers continue to seek—a set of "spells" that will magically ensure success. Design patterns can be seen as such a set of spells—apt to be promoted as a way to ensure that software systems will work and that someone who understands the concepts can expect to be able to maintain the systems effectively.

The Basis for Design Patterns

Of couse, there's a sound basis for this idea. It's normal for designers in any discipline to later reuse positive experiences, adapting designs to meet new goals as appropriate. The software design patterns community tends to cite Christopher Alexander's ideas about patterns in architecture as their touchstone,[1] but we can see similar use of this concept in the design of motor vehicles, clothing, public transportation systems, libraries, and so on. Some uses involve static forms, but others are processes, rather like software.

In their early work, Beth Adelson and Elliot Soloway observed experienced software designers identifying their intention to reuse partial designs by assigning labels to them in new design plans.[2] Francoise Détienne characterized this behavior from a cognitive perspective as using schemas—that is, mental models that encapsulate different aspects of a designer's experiences.[3]

So it's maybe not surprising that software developers readily embraced this idea when the Gang of Four (GoF) disseminated it through their book, which cataloged 23 software design patterns.[4] The book has become widely (and probably wrongly) seen as definitive, although the patterns community has also gone on to create and catalog many more patterns.

Indeed, this enthusiasm for pattern mining has tended to obscure some key questions that remain largely unanswered, such as

- *the design process question*: Do patterns form an effective way of exchanging design knowledge? and
- *the design product question*: Do patterns help us to create designs that are easier to understand and maintain?

Neither question seems to have received much consideration from design pattern enthusiasts. However, as often happens with software design, when we look at the available evidence, the answers to both questions are rather mixed. So, as with Gilbert and Sullivan's sorcerer, applying the magic doesn't always produce the intended results.

0740-7459/13/$31.00 © 2013 IEEE

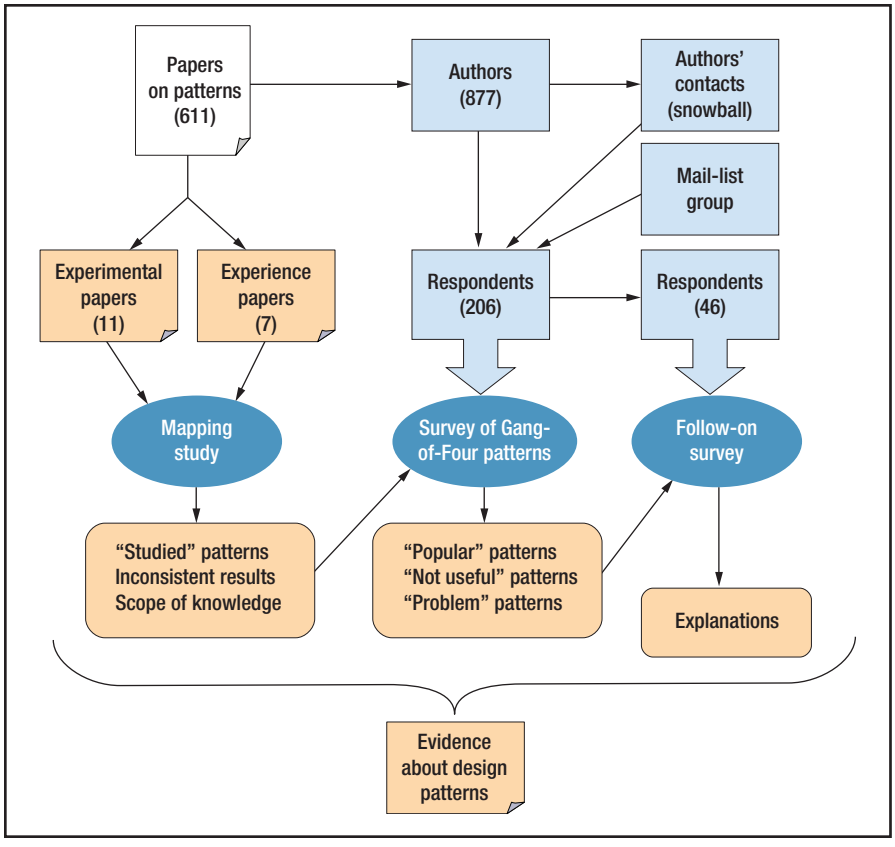

FIGURE 1. Mapping study and two follow-on surveys. The initial mapping study identified 611 papers about software design patterns through the end of 2009. The two follow-on surveys both extended and helped interpret the mapping study results.

A Set of Empirical Studies

What I describe here is largely derived from studies conducted over the past few years with one of my students at Durham University. Figure 1 summarizes the relationships between these studies, which consist of a mapping study, conducted to identify relevant empirical studies,[5] and two surveys that followed.[6,7]

In conducting the mapping study, we identified 611 papers about software design patterns in the period ending in 2009. From these papers, we identified and analyzed 11 experimental studies (the only ones that had been conducted in over a decade of pattern use) and seven "observational" studies to help explain and interpret the findings. All of the studies used GoF patterns.

Next, we organized a survey of the 877 authors of the mapping study papers to find out more about their views of the GoF patterns. Of course, some author addresses were no longer valid, leaving 681 authors, but we augmented these by asking respondents to pass our invitation on to appropriate colleagues and also by sending it to members of a mail-list for people interested in patterns. This eventually produced 206 usable responses (more than we had expected), enabling us to create a profile of "usefulness" for the patterns. Finally, we conducted a follow-on survey with those 206 respondents to see if we could better understand the three patterns that produced particularly varied reactions. Of 46 respondents, 27 provided comments and experiences.

An obvious question is how solid a body of evidence this all forms? The mapping study involved a comprehensive search for primary studies, so we can conclude that its outcomes are fairly complete and unbiased, even if the empirical data isn't very extensive. The surveys sampled three different respondent groups, although our analysis showed them to have generally similar education and experience profiles. In addition, because the largest group included people who had written about patterns, we might consider them to be not only knowledgeable about patterns but also more likely to favor the concept. This should add weight to any reservations they expressed.

What Did We Learn?

The experimental studies we analyzed for the mapping study were diverse in many ways, although two did claim to replicate others. In particular, only three patterns were studied in more than half the experiments we examined—namely, Composite, Observer, and Visitor. The different studies used a mix of participants, including advanced undergraduates, postgraduates, and practitioners—all with some degree of pattern experience, although calibrating the extent of this experience is difficult.

When interpreted with the aid of the observational studies, we concluded the following:

- The use of Composite seemed to create few problems, although we noted that users needed to understand recursion.
- Likewise, the use of Observer seemed to create few problems beyond the risk of producing overly complicated designs.

- In contrast, the outcomes from studies using Visitor were more ambivalent. On the one hand, its complexity was seen as a barrier to effective use; on the other hand, when used well, Visitor might actually aid system maintenance.

We conducted our surveys as a way to more clearly indicate which patterns were considered useful under what circumstances (something difficult to explore with an experiment). We asked respondents to assess the usefulness of all 23 of the GoF patterns (with the alternative of stating that a pattern wasn't one familiar to them). Then we asked them to identify up to three patterns that they considered really useful and, in contrast, up to three that they considered of no value.

With few reservations, the Observer and Composite patterns were the top two choices in this ranking process, reinforcing the mapping study results. Also as with the mapping study, results related to the Visitor pattern were mixed. There was also a group of patterns considered of little real use: Flyweight, Interpreter, Prototype, and Memento, with the last receiving no positive "votes" at all. A few patterns, such as Chain of Responsibility, were noted as being useful only for specialized purposes.

Like Visitor, the Façade and Singleton pattern votes were ambivalent. Neither pattern was studied in other than one of the inputs to the mapping study. We conducted the second survey to clarify what characteristics of these three patterns caused such divergent views. The reasons turned out to be rather different for each one:

- Visitor was seen as useful only for limited purposes, and there was concern that its use could easily lead to implementation constraints and problems.
- Singleton was also seen as having only specialized purposes—but the expressed concerns tended to focus on its alternative role as a means of storing global variables within object-oriented systems. Some respondents viewed this as a useful facility; others felt (strongly) that it conflicted with the object-oriented model.
- Façade proved to be much less controversial than the other two, but some respondents were concerned about the way it could constrain system maintenance.

So, even though many respondents valued all three patterns, there was a strong sense that this approval was quite strongly conditional in nature.

So, How Far Does the "Magic" Work?

Perhaps inevitably, the evidence we can offer about the use of design patterns shows them to be part magic with a good stiffening of myth. Table 1 summarizes the seven patterns for which one or both forms of empirical study provide a useful contribution to the evidence.

With some caveats, the table shows that the Observer and Composite patterns are generally well-supported by experimental data and experience. Similarly, it suggests that the Visitor pattern is supported well enough for use with some caution. Beyond that, experimental evidence is thin, although our survey data of experience gives generally good support to Abstract Factory, Façade, Factory Method, and Iterator. The evidence suggested practicing something more than caution in the use of Flyweight, Interpreter, Memento, and Prototype. Otherwise (and avoiding the issues with Singleton), we have only very limited systematic data.

So, how well can we answer the two general questions regarding the effectiveness of design patterns posed earlier?

- In terms of supporting the exchange of design knowledge, patterns probably are effective in the hands of experienced users, but our combined studies suggest that the learning curve involves potential hazards (for an illustration of this point, see the report from Peter Wendorff[8]).
- In terms of helping to create products whose designs are easier to understand and maintain, the question remains unanswered by the data available to us. Most of the experimental studies analyzed in our mapping study involved participants who performed tasks related to maintenance, but they provided little, if any, support for the benefits of using patterns.

> The evidence we can offer about the use of design patterns shows them to be part magic with a good stiffening of myth.

Many users believe in the effectiveness of patterns and offer some examples to support their views. However,

TABLE 1

Summary of evidence about seven patterns as derived from the mapping study and two surveys.*

Pattern	Support from mapping study**	Support from surveys	
		Survey 1†	Survey 2
Observer	Used in seven experiments that indicated only minor issues of comprehension and overly complicated solutions; three experience reports were generally supportive but also noted the risk of overly complicated solutions	Largest count of positive "votes" (57) with only 3 negative ones	n/a
Composite	Used in seven experiments that noted only a need for a knowledge of recursion; the single experience report noted no issues	Second highest count of positive votes (46) and only 1 negative one	n/a
Abstract Factory	Used in only two (quite diverse) experiments; discussed in two experience reports	Received 31 positive votes and 2 negative ones	n/a
Façade	Used in only one experiment; discussed in two experience reports	26 positive votes but 7 negative ones	Considered easy to misuse with risk of the structure being lost, but considered useful for integrating legacy code
Factory Method	Used in three very diverse experiments; no experience reports	Received only positive votes (20)	n/a
Iterator	Only one experience report	Mostly positive votes (23) with 2 negative ones	n/a
Visitor	Used in seven experiments with mixed outcomes, some suggesting that it enabled faster modification, although the opposite was found in a replication study; the one observational report noted that it "may be hard to understand"	A mixed profile with 26 positive votes but 11 negative ones	Considered to be complicated and likely to impose constraints, with negative effect upon implementation

** Table elements shaded in green indicate a lack of material that could be aggregated.*
***Eleven experimental studies and seven experience reports.*
† Total positive votes is 389; total negative votes is 113.

like the sorcerer's magic, the use of design patterns can generate unwanted effects, especially during a designer's apprenticeship (a particular problem Wendorff identified[8]), and clearly, the effective use of patterns requires much more than a catalog, however good it might be.

Acknowledgments

Much of this work was conducted in collaboration with Cheng Zhang as part of his PhD study. I am also grateful to the School of Engineering and Computing Sciences, Durham University, for providing support for the third follow-on study as well as to Sarah Drummond for her help with the qualitative analysis.

References

1. C Alexander et al., *A Pattern Language*, Oxford Univ. Press, 1977.
2. B. Adelson and E. Soloway, "The Role of Domain Experience in Software Design," *IEEE Trans. Software Eng.*, vol. 11, no. 11, 1985, pp. 1351–1360.
3. F. Détienne, *Software Design—Cognitive Aspects*, Springer, 2002.
4. E. Gamma et al., *Design Patterns: Elements of Reusable Object-Oriented Software*, Addison-Wesley, 1995.
5. C. Zhang and D. Budgen, "What Do We Know about the Effectiveness of Software Design Patterns?" *IEEE Trans. Software Eng.*, vol. 38, no. 5, 2012, pp. 1213–1231.
6. C. Zhang and D. Budgen, "A Survey of Experienced User Perceptions about Software Design Patterns," *Information & Software Technology*, 2012, in press; http://dx.doi.org/10.1016/j.infsof.2012.11.003.
7. C. Zhang, D. Budgen, and S. Drummond, "Using a Follow-on Survey to Investigate Why Use of the Visitor, Singleton, and Façade Patterns Is Controversial," *Proc. 6th Int'l Symp. Empirical Software Eng. and Measurement* (ESEM 12), ACM, 2012, pp. 79–88.
8. P. Wendorff, "Assessment of Design Patterns during Software Reengineering: Lessons Learned from a Large Commercial Project," *Proc. 5th European Conf. Software Maintenance and Reengineering* (CSMR 01), IEEE CS, 2001, pp. 77–84.

DAVID BUDGEN is chair of software engineering in Durham University's School of Engineering and Computing. His research interests include software design and evidence-based software engineering. Budgen received his PhD in theoretical physics from Durham University. Contact him at david.budgen@durham.ac.uk.

Selected CS articles and columns are also available for free at http://ComputingNow.computer.org.