Non-Software Examples of Software Design Patterns

Michael Duell
AG Communication Systems
duellm@agcs.com

John Goodsen
Saguaro Software
jgoodsen@radsoft.com

Linda Rising
AG Communication Systems
risingl@agcs.com

Abstract

Many people are overwhelmed by their initial exposure to design patterns. Requests for more examples are common feedback from Design Pattern training courses. Unfortunately, meaningful examples are not always readily available.

The patterns presented by Gamma, Helm, Johnson, and Vlissides (Gang Of Four) [1] are, by their own admission, elegant solutions to problems rather than the designs that people initially develop. The proprietary nature of software further limits the access to meaningful examples. Although access to examples of patterns may be limited, examples can be found elsewhere. Software Design Patterns have roots in both the work of architect Christopher Alexander, and in the Object movement. According to Alexander, patterns repeat themselves, since they are a generic solution to a given set of forces. The object movement looks to the real world for insights into modeling software relationships.

With these dual roots, it is a reasonable hypothesis that the software design patterns should be repeated in real world objects. This workshop presented an opportunity to develop real world examples of the Gang Of Four patterns and explore the insights gained from the process of developing the examples.

Participants submitted two or more examples of non-software instances of the Gang Of Four software design patterns. Each participant read all submissions before the workshop. The time was spent examining each example, both to determine its appropriateness and to share suggestions for improvement. The process of determining how well an example fits a particular pattern provided new insight on each of the patterns.

1 Participants

In addition to the workshop organizers, the following individuals participated in the workshop:

- Brian Campbell, AG Communication Systems (campbelb@agcs.com)
- Jens Coldewey, Independent consultant (jens_coldewey@acm.org)
- Helen Klein, University of Michigan (hkein@umich.edu)
- James Noble, Macquire University (kjx@mpce.mq.edu.au)
- Michael Richmond, Macquire University (rar@krakatoa.mq.edu.au)
- Bobby Woolf, Knowledge Systems Corporation (bwoolf@ksccary.com)

2 Pattern Examples

A short summary of some of the pattern examples presented follows:

2.1 *Visitor* Example

The *Visitor* example involves an outside consultant coming into a department to meet with employees. While the visitor is escorted to each cubicle, she does nothing. Once she arrives at a cubicle, she introduces herself, and the employee introduces himself (double dispatch). Following the introduction, the consultant springs into action interviewing the employee (sending messages and obtaining results). The basic idea is that the consultant is not required to have knowledge of the organization's

structure to do her job. The escort supplies this knowledge.

2.2 *Proxy* Examples

The first *Proxy* example was a shareholder's proxy ballot as a *Proxy* for the shareholder. This example was "intentionally" bad as it shows that name correspondence does not necessarily make a good example. The proxy ballot does not control access to the shareholder, and does not provide an interface to the shareholder.

Another *Proxy* example involves two people who refuse to talk to one another, and each uses a separate third party to communicate. In other words, person A and person D are not speaking, so A talks to B, who passes the message to C, who then relays the message to D. As this example uses a "double" proxy, most participants felt that the added complexity detracted from the explanatory power of the example.

A better example involves asking technical questions of a Sales Representative. In all likelihood, the Sales Representative will forward the technical request to a Technical Representative and relay the answer to the customer.

Other *Proxy* examples developed in the workshop included:

- 1-8xx numbers in the United States and Canada, which translate to a routing number.
- An ambassador, who is a *Proxy* for his or her government.

2.3 *Iterator* Examples

The properties tool in VisualWorks uses the Prev and Next buttons to act as an *Iterator* for the canvas tool. While this is not a non-software example, the example shows the *Iterator* behavior, and there is no need to examine source code.

Other *Iterator* examples included the receptionist in a doctor's waiting room. The receptionist iterates the aggregate of patients who are seated randomly

around the room. The receptionist will send the "next" patient to the doctor.

The channel selector on modern day television sets is another example of an *Iterator.* Rather than a dial containing all possible channels or one button per tuned channel, today's television sets have a "Next" and "Previous" button for tuning channels. The "Channel Surfer" doesn't need to know if Channel 3 or Channel 4 comes after Channel 2.

All three of these examples demonstrate the intent of the *Iterator*, since the *Iterator* knows how the objects are ordered.

2.4 *Mediator* Examples

The first *Mediator* example was a set of (Australian) traffic lights in an intersection. The traffic light encapsulates the rules of right of way, rather than distributing them among the drivers (Give way to the left; through traffic has the right of way over turning traffic). This example was interesting, because it is readily acceptable as a *Mediator* at first glance, but upon deeper examination, the actions of the traffic light (mediator) are independent of the actions of the cars (colleagues). Even with no traffic, some traffic lights will change according to a set schedule.

The traffic light example raises the question of how closely an example must match a pattern to be useful in communicating the concept.

Two other examples of *Mediator* that match the pattern more closely are the Emergency Services Dispatcher and Air Traffic Control at a controlled airport [2].

In the Emergency Services example, individual ambulances do not track the location of other ambulances. Instead, the dispatcher tracks all vehicles in the area. The communications channel (usually radio) does not get congested with low priority messages among the vehicles but is prioritized by the dispatcher.

In the Air Traffic Control example, the control tower at an airport provides a central point of communication for aircraft in the terminal area. Constraints on terminal area airspace are maintained by the tower. With the centralized communication and constraint maintenance, the tower behaves as a mediator.

2.5 *Observer* Examples

The CNN news agency was presented as an example of the *Observer* pattern. The news agency acts as the subject, and the viewing public are the observers. When something in the world changes, everyone is notified. The discussion of this example centered on whether or not the subject knows the observers and whether or not registration is essential to the pattern. Turning on a television set could be considered to be registering. This broadcast method of distributed communication is common in data communications.

An alternate example of warranty registration was given for *Observer*. Consumers (observers) who send in warranty registration cards to a company (subject) will be notified in the event of a recall, new version release or any other news related to the product. This example uses explicit registration, and the collaborations may be easier to see.

2.6 *Chain of Responsibility* Example

The chain of command in the military illustrates the *Chain of Responsibility* pattern. When a request is made of a private, the private can either grant the request or forward the request to his or her superior. The request will be forwarded until someone with the appropriate authority is able to act upon it.

2.7 *Factory Method* Example

A bread machine is an example of a *Factory Method*. The machine provides an interface for creating bread, but the exact type of bread is deferred until a recipe is followed. The same machine can be used to make white bread, wheat bread, cheese bread or cinnamon bread.

2.8 *Builder* Example

The use of a bread machine can be considered to be a *Builder*. If a wife asks her husband to make some cinnamon walnut bread, she does not care what process he uses. Whatever process is used, the wife expects a loaf of bread, containing cinnamon and walnuts.

2.9 Template Method/Strategy Example

Recipes can often be the basis for a *Template Method*. Once a basic recipe for bread, muffins, soup stock, etc. is developed, the addition of different ingredients can be used to differentiate the final dish.

Some participants felt that recipes were a better example of *Strategy*, since a collection of bread recipes represented different options for making bread.

The discussion of recipes as an example of *Strategy* shed some light on *Strategy* examples. The difficulty in representing a software design with objects in the real world is finding something that represents behavior. For example, in considering different alternatives for getting to work, e.g. bicycling, taking the bus, car-pooling, driving alone, etc., what objects would represent the different strategies? Does a bicycle represent the act of bicycling? One good way of solving this is to capture action or behavior in something like a recipe. The recipe clearly represents a series of actions: do this, then this, and finally this. The concrete object, the recipe, represents the behavior in the pattern.

2.10 *Memento* Examples

Using the brakes on one side of a car as a guide to reassembling the brakes on the other side [2] was presented as an example of a *Memento*. This example relies on the assumption that brakes on both sides of car are substantially similar. The assumption that the brakes on one side of a car are similar to the brakes on the other, was not considered to be an encapsulation of the state of the brakes.

An example of a photograph and insurance appraisal for jewelry was also given for *Memento*. In this example, although the jewelry can be replaced if lost or damaged, the identity of the object changes. Even if a diamond of exact cut, color, clarity and carat weight replaces a lost diamond, it is not the same diamond.

An example of audio mixing equipment was then proposed. Since there are several switches set in different positions, it is common to photograph the equipment, so that the switches can be returned to their original positions if perturbed.

Yet another example was proposed at the poster session. When a restaurant patron enters an establishment wearing a coat, it is common for the coat to be left in the coat room while the patron dines. The patron is given a claim check, so that upon leaving the restaurant, he or she is returned to the same state: HAS_COAT. In this example, the claim check is used to ensure that the coat the patron has upon leaving the restaurant is the same coat that he or she had upon entering.

3 Insights

Even the best examples began to break down when the granularity of analysis became too fine. For example, in the *Visitor* example, a software visitor never asks to do things unrelated to the visit tasks (such as go to lunch), but the consultant *Visitor* might. When human behavior is involved, the potential for deviation from the pattern collaborations is higher, since humans are more complex than software systems. Workshop participants did not believe this deviation to be a problem, since the examples were meant to be illustrative, rather than strictly equivalent. If non-software examples are to be used as an aid to understanding patterns, the limitations of the example should be described, however.

3.1 What Makes A Good Example?

After discussing specific examples, workshop participants began to abstract the examples in order to determine the types of non-software examples that are effective in explaining a given pattern. In general, a good example should have identifiable participants which correspond with the participants in the pattern. The participants in the example should also demonstrate behavior comparable to the participants in the pattern. Consider the receptionist in the doctor's office. The receptionist corresponds to the *Iterator* and the patient sign-in list corresponds to the aggregate. The behavior of the receptionist and sign-in list are also comparable to the behavior of the *Iterator* and aggregate.

An example is also better when the consequences map to the consequences of the pattern. Consider the air traffic control example. Like the *Mediator,* the colleagues (airplanes) are decoupled; many-to-many interactions are replaced by many-to-one interactions; mediation is encapsulated in a separate object (the ATC tower); and control is centralized.

There was no agreement on whether people, places, things, or ideas made better participants for pattern examples. Based on opinions expressed by individual workshop participants, the preference for examples employing people, place, or thing participants is dependent upon the individual studying the example. Since the effectiveness of an example is largely dependent upon individual preference, several examples for each pattern will help communicate the pattern to a larger audience.

3.2 What Does a Lack of Example Imply?

It is easier to find examples for some patterns than it is for others. Originally, workshop participants had thought that the type of pattern (Creational, Structural or Behavioral) might impact the ability to develop good examples. Structural pattern examples were thought to be easier to find than Behavioral examples, but the lack of a good *Bridge* example, and the presence of several good Behavioral examples (such as *Iterator* and *Mediator*) cast some doubt on the validity of this statement.

Generally, workshop participants agreed that it is more difficult to find examples for patterns

relying on inheritance (such as *Bridge*) and specific object identity (such as *Memento*). The lack of a good non-software example was said to imply that the system of forces resolved by the pattern are particular to software. For example, *Bridge* is often used when an implementation must be selected or switched at run time. Since run time binding is primarily a software concept, finding a non-software example is difficult.

4 Poster Session

Several pattern examples were on display during the poster session. Conference attendees were requested to provide feedback on which examples they liked, which they did not like and why they liked or disliked an example. Some examples were liked by some people but disliked by others. In general, the examples that were liked were preferred because an aspect of the pattern was emphasized well in the example. For example, the receptionist example for *Iterator* was liked because the ordering of patients appeared to be random, yet the *Iterator* was able to iterate them sequentially. Likewise, the taxi cab example for *Visitor* [2] was not liked because double dispatch was not evident. Although the taxi example does contain double dispatch (the driver announces his presence by honking the horn, and the passenger enters the cab and tells the driver her desired destination), it was not stated explicitly, and therefore was missed by many studying the example.

5 Conclusions

Good examples of software design patterns can be found in non-software settings. Good examples will have participants that correspond to the participants in a given pattern. They will also have consequences similar to those of the given pattern. These examples can be effective in communicating the intent, participants and consequences of software design patterns, thereby making them easier for those newly introduced to patterns to understand. As the patterns were discussed in the workshop, participants found non-software examples were also a useful tool for people with patterns experience to communicate details of patterns.

Even when aspects of an example deviate from a pattern in terms of structure, object identity, or behavior, the example may still be useful for explaining a pattern, provided the limitations are noted. With or without limitations, the explanatory power of any example is largely dependent upon the individual studying the example. For this reason, several examples for each pattern should be developed to increase the probability that the pattern will be understood. Whenever an aspect of a pattern is important, it should be emphasized in the example rather than leaving it as an exercise for the reader.

Examples in which the participants correspond to the participants in the pattern, consequences are similar, and limitations and important aspects are called out explicitly have great power not only as learning aids but also as metaphors to improve communication between designers.

6 Important Websites

The submissions to this workshop can be found at:

- `http://www.radsoft.com/patterns/oopsla97.htm`

A copy of reference [2] can be found at:

- `http://www.agcs.com/patterns/papers.htm`

References

1. Gamma, E., R. Helm, R. Johnson, J. Vlissides, *Design Patterns-Elements of Resuable Object-Oriented Software,* Addison Wesley, Reading, MA, 1995.
2. Duell, M. Non-Software Examples of Software Design Patterns, *Object,* 7(5), July 1997.