Evaluating Software Design Patterns

— the "Gang of Four" patterns implemented in Java 6

Thesis Specification

(5 pages)

August 2007

Gunni Rode

http://www.rode.dk/thesis

Supervisor: Professor Eric Jul

Table of Contents

Introduction

Designing and developing complex software systems is not, and has never been, an easy task. On the contrary, the process is often very time consuming; it requires interaction between many different people, skills, and roles, internally and externally. The entire process, therefore, worst of all, tends to be error prone, not forgetting costly. Even more so, the larger and more complex the system is, the worse these factors seem to become. However, even the most complex systems are built by using smaller *parts*; a part can be anything from an entire subsystem, native to the language or otherwise, to a specific file or class. Such parts may in turn be built using even smaller parts, and so forth, and therefore need to *communicate* to function. If we view a part as a *problem* to be solved, one approach to identify re—occurring problems and their well—proven solutions is to use *Software Design Patterns*.

A pattern is based on empirical knowledge and communicates when the pattern is applicable as well as the solution to the problem at hand (see for example [PPR; Hillside; Lea00]). Patterns are named and written using a consistent format, thereby allowing both developers and others to communicate using a common *vocabulary*, or *language*, thus facilitating the entire design and development process. The notion of patterns was originally coined by Christopher Alexander [Alexander77] within the field of architecture, but has been shown to be applicable in many other areas as well, but perhaps most notably within the field of computer science. Arguably some of the most utilised software design patterns are the patterns authored by Gamma et al., dubbed the “Gang of Four” patterns [Gamma95]. They pertain to Object—Oriented (OO) design, and describe communicating objects and classes that are customized to solve a general design problem in a particular OO context [Gamma95, p.3]. The pattern descriptions provide canonical implementations in C++ and/or Smalltalk, and discuss numerous language issues to be aware of during implementation. We wish to implement the “Gang of Four” patterns in Java 6 to evaluate how features found in Java can be utilised in the application of the patterns, for example (runtime) features such as annotations, closures, reflection, and dynamic proxies. The objective is to provide a subjective evaluation, not a definitive conclusion as this goes against the very idea of design patterns, which may be useful in disclosing how the “Gang of Four” patterns and Java 6 can cooperate.

Goals

The main goal of this Master Thesis is to obtain a Master Degree in Computer Science at the University of Copenhagen. The specific goals for the content of the thesis are:

I. Present an introduction to and a discussion about the theory deemed necessary to understand topics covered by the evaluation. This will include OO; patterns in general with focus on software design patterns, especially the “Gang of Four” design patterns; and a discussion on selected related works and topics.

II. Implement and evaluate the “Gang of Four” design patterns in Java 6.

III. Present the evaluation outcome and comment on the findings separately for each evaluated pattern and by juxtaposing the individual evaluations.

Motivation

The motivation for undertaking this project is for the undersigned to get a better understanding of design patterns related to OO, in particular the “Gang of Four” patterns and how they relate to Java 6. This is relevant as the “Gang of Four” patterns are frequently used in real—life systems, and so is Java, but Java 6 furthermore offers a range of versatile features that will be interesting to apply in the pattern implementations.

Participant, Supervisor, Credit, and Time Frame

The work is carried out by Gunni Rode under the supervision of Professor Eric Jul at the department of Computer Science, University of Copenhagen. The workload is 30 ECTS, and the thesis will be handed in for grading no later than the 1st of September 2007.

Project Phases

The work on the thesis will be partitioned into several phases:

1. Understanding and describing the history and the basic theory behind patterns in general, including Alexander’s work [Alexander77; Appleton00; Lea93], but focusing on software patterns;
2. Describing the relevant OO theory, in particular the OO development life—cycle with special attention on how patterns can be aid the design (and implementation) process. Focus will be on the “Gang of Four” patterns, which will also be introduced in full;
3. Establish and explain the overall evaluation set—up required for the evaluation, including describing the overall approach to the implementation. This is required as the evaluation will be subjective, yielding subjective conclusions;

4. Implement the “Gang of Four” design patterns in Java 6 in accordance with the context established in item 3, while recording particular useful features to be used in the evaluation;
5. Evaluation of the implementations authored in item 4. Each pattern will be evaluated individually as well as collectively compared with the other patterns, to identify possible common traits and issues;
6. Completing the thesis. The thesis will be concluded with conclusion on the work performed, as well as an outlook and suggestions on further work.

The end result will be a thesis and a CD containing the developed code. A **proposed** outline of the major chapters is listed below; each chapter is assigned an **informal** weight to indicate its contribution to the overall work put into the thesis as well as its **suggested** number of pages (assuming 100 pages in total):

1. **Introduction** (1%, 3p) – *Abstract, Foreword, Formalities, Goals, Thesis Summary;*
2. **Object—Oriented Development** (12%, 14p) – *Concepts, Analysis, Design, Programming, Connection to Design Patterns, “Gang of Four”, etc;*
3. **Pattern Theory** (13%, 14p) – *Origin, Christopher Alexander, Design Patterns, Descriptions and Properties, Software Design Patterns, “Gang of Four”, etc;*
4. **Related Work** (8%, 7p) – *Related Studies;*
5. **Comparative Evaluation** (28%, 26p) – *Identifying Common Traits, Language Features, Pattern Relationships, etc;*
6. **Individual Evaluation** (33%, 30p) – *Pattern Description, Implementation, UML, Features used;*
7. **Evaluation Conclusions** (4%, 4p) – *General Subjective Conclusions based on 5 & 6;*
8. **Conclusion** (1%, 2p) – *Final Remarks, Conclusion, Follow—up on Goals, Further Work.*

Chapters 1 & 8 will supply the introduction and conclusion (~2%, ~5p); chapters 2-4 will explain and discuss the main theory (~33%, ~35p); while chapters 5-7 will contain the main work, i.e. implementation and evaluation (~65%, 60p).

Bibliography

Below is an **initial** list of references used.

[Alexander77] **A Pattern Language** - Towns, Buildings, Construction
Christopher Alexander
1977; Oxford University Press; ISBN 0195019199

[Appleton00] **Patterns and Software** - Essential Concepts and Terminology
Brad Appleton
2000; http://www.cmcrossroads.com/bradapp/docs/patterns-intro.html

[Gamma95] **Design Patterns** - Elements of Reusable Object—Oriented Software
Erich Gamma, *Richard Helm*, *Ralph Johnson*, and *John Vlissides*
1995; Addison Wesley Longman, Inc.; ISBN 0201633612

[Hillside] **Hillside.net** - Online Pattern Catalog
http://hillside.net/patterns/onlinepatterncatalog.htm

[Lea93] **Christopher Alexander: An Introduction for Object—Oriented Designers**
Doug Lea
1993; http://g.oswego.edu/dl/ca/ca/ca.html

[Lea00] **Patterns—Discussion FAQ**
Doug Lea (maintained by)
2000; http://g.oswego.edu/dl/pd-FAQ/pd-FAQ.html

[PPR] **Portland Pattern Repository**
http://c2.com/cgi/wiki