Empirical Study of Design Pattern Usage in Peer-to-Peer Systems

Markus Aleksy[1], Martin Schader[1], Christian Seifried[1], and Makoto Takizawa[2]

[1] University of Mannheim
Department of Information Systems
[2] Tokyo Denki University
Department of Computers and System Engineering

Abstract. In this paper, we present the results of our analysis in which we investigated the different existing peer-to-peer systems with regard to design pattern usage. In the course of our analysis, we mainly concentrated on patterns concerned with the classification of distributed systems. In the following, the design patterns investigated are examined in more detail in their usage context and possible alternative solutions are discussed.

1 Introduction

The peer-to-peer (P2P) concept has experienced a revival over the last few years. The approach does not describe the exchange of data or how the communication between two participants takes place; rather, it is much more an architectural form that can be applied to all levels of a system. Put simply, a peer-to-peer system makes it possible for two or more equal participants to spontaneously form a network and to collaborate over it. In this way, it is not dependant on any central coordination entity.

During the development of a complex system, such as a peer-to-peer system, there are many different problems that appear again and again. At this point, the application of design patterns may be called for. Design patterns are simple and concise solutions for frequently occurring programmer tasks. A design pattern documents a comprehensive and viable solution, won through experience, to frequently recurring problems during program design.

The primary benefit of a design pattern lies in the delineation of a solution for a particular class of problems. Additional advantages are derived from the fact that each pattern has a name. This simplifies discussion amongst software developers by allowing for abstract dialog about software structure. Design patterns are, in this way, basically language independent. In the development of object-oriented software, they serve as an accepted tool for design structuring. The usage of tested and proven design patterns can lead to shorter development times, minimization of mistakes, and a higher application quality.

The design patterns found to be particularly useful for the development of P2P systems according to our analysis can be grouped as follows:

T. Enokido, L. Barolli, and M. Takizawa (Eds.): NBiS 2007, LNCS 4658, pp. 335–344, 2007.

- Communication
  One of the most crucial aspects for the proper functioning of a P2P system is communication between peers. If the developer fails to install a well-designed communication infrastructure that can be used to exchange all the required information such as routing messages, lookup/location queries and results etc., the backbone of the P2P system will be missing. For the design and implementation of such a foundational system layer it is therefore essential to apply reliable and well-suited procedures. In this context, design patterns such as *Acceptor-Connector*, *Forwarder-Receiver*, *Non-blocking Buffered I/O*, *Fire and Forget*, and *Asynchronous Completion Token* should be considered.
- Location of data, peers and services
  In order for a P2P system to operate successfully it must be possible to locate other peers participating in the network and the data and services they offer. Design patterns such as *Lookup*, *Location*, and *Service Locator* put forward valuable approaches to the solution of this problem.
- Resource Management
  In this category, we subsume design patterns that describe solutions to the problem of controlling and managing access to and availability of peers and the resources they own. Typical representatives are, for example, the *Leasing* pattern and the *Heartbeat* pattern.
- Caching
  Caching and *Evictor* are to be mentioned as examples of design patterns that specify mechanisms to care for an optimized placement of data within the P2P network.
- Security
  This category comprises design patterns like *Security Policy* or *XML Firewall*, which are concerned with security aspects and are applicable to P2P systems.

2 Empirical Study

Among the goals of our study was the examination of the following questions:

- Which design patterns appear to be especially well suited for designing a P2P system?
- Are design patterns typically used to the same extent in all of the different categories introduced above?

We defined a set of design patterns to be looked for in our analysis. This set was mainly drawn from standard literature in the field, such as [4], [9] and [13]. [13], for example, served as a source for selecting existing design patterns for the area of "Communication", and [9] provided design patterns for the categories "Location", "Resource Management", and "Caching". This selection was extended with several other design patterns drawn from papers published at specific technical conferences such as "Pattern Languages of Programs (PLoP)"