Hierarchical Patterns: A Way to Organize (Analysis) Patterns

Lubor SESERA
Softec, Ltd.
Kutuzovova 23, 831 03 Bratislava, Slovakia

ABSTRACT

The paper addresses the issue of categorization and generalization in software patterns. It focuses on the realm of analysis (conceptual) patterns in which the problem is more noticeable when compared to design patterns. The paper introduces hierarchical analysis patterns as a means for categorization and balancing generality and real-world usefulness. A three level hierarchy of analysis patterns is presented. It is documented using real-world examples. Finally, there is a rational that hierarchization might be useful for other kinds of software patterns as well.

Keywords: Analysis Patterns, Hierarchical Analysis Patterns, Design Patterns, Generalization, Insurance.

1. INTRODUCTION

Software patterns are an established realm of software engineering. Many times software patterns are considered identical to design patterns due to the famous book of Gang of Four [5]. However, design patterns focus on one aspect of software development only, design micro-architecture in particular. There exist other kinds of software patterns such as analysis (conceptual) patterns, architecture patterns, programming patterns (idioms), process patterns, project management patterns, anti-patterns, etc. This paper addresses analysis patterns. Analysis patterns are used in the analysis phase of software development to build a conceptual model of a system.

One of the key issues of software patterns is categorization that would lead to a widely accepted system of patterns. In design patterns this is partially compensated for by the 'bible' book of Gang of Four. In analysis patterns the situation is fairly different. There exists neither an accepted system of patterns nor a 'bible'. Three main analysis patterns books [2], [3], [4] and some papers have been published; however, none of them has gained a position comparable to the Design Patterns book. Apart from [3], they do not even claim to build an extensive system of patterns. Worse, patterns created by distinct authors are distinct in their 'nature', which is why such patterns are difficult to compare and combine. In [8] we needed several dimensions to compare them. In particular, we used abstraction/generalization, flexibility and granularity dimensions. The key difference is in the approach to generalization.

This paper introduces hierarchical analysis patterns as a means to categorize and combine analysis patterns. In the next section a three level hierarchy of analysis patterns is proposed starting with the most general level. It is demonstrated using real-world examples. In Section 3 horizontal relationships are discussed. Finally, there is a rational that hierarchization might not be constrained to analysis patterns only but it can be useful for other kinds of software patterns as well.

2. HIERARCHY OF ANALYSIS PATTERNS

Analysis patterns have to cope with two opposing forces:

1. Generality. When patterns are more general they are also more reusable and high reusability has been declared the main goal of patterns.
2. Usefulness. Real-world usefulness of patterns decreases with generality as patterns are not only less understandable but, especially, they omit a lot of 'details'.

So far analysis patterns and systems of analysis patterns are 'flat'. As there is no general agreement how to balance those forces, distinct authors 'balance' them differently. For instance, Peter Coad's patterns [2], are very general and simple, David Hay's patterns [3], are constrained to the realm of traditional enterprise systems, and Wolfgang Keller's patterns [6] are restricted just to the insurance industry. Alongside, there are Martin Fowler's patterns [4] as pearls of abstraction process that are difficult to combine even among themselves.

Our point of view is that this tension is inherent in analysis patterns and there is little chance it can be balanced by the proper level in a flat system. Rather, the issue should be solved by introducing levels of analysis patterns where each level provides the proper generality and abstraction[1] with regard to its objective. A more general level gives a framework for a more specialized level. It can be used when the specialized level has not been created yet. On the other hand, the specialized level can provide a rather specific guideline for an analyst building a conceptual model of a real-world system. Our hierarchy of analysis patterns is shown in Figure 1. It consists of three fundamental levels. However, if needed, any level can be decomposed further to its sublevels.

There is a question how to build the most general level. The answer might not be in patterns published but in the more general ISA framework of Zachman [11]. Any complete system should include aspects of *Who, What, How, Why, Where* and *When*. Although (at least so far) analysis patterns are data model patterns only, they should include these aspects on a small scale. In particular, four main packages of general analysis

[1] To be precise, the generality and abstraction concepts are not the same (see e.g. [10]). For simplicity, here we omit subtle discrepancies.

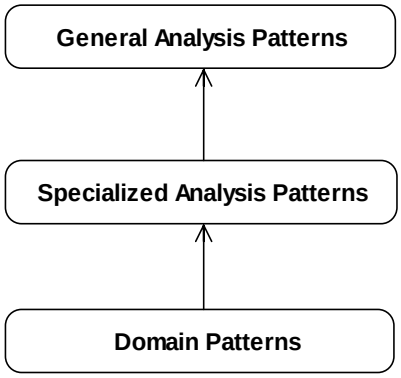

Figure 1 *Hierarchical levels*

patterns should be built: *Parties* (Who), *Objects* (What), *Operations* (How) and *Accountabilities* (Why). Aspects of location (Where) and time (When) are inherent parts of the main packages. It is quite interesting that these packages are not far from David Hay's [3] 'anchors for data models'[2] if they are generalized above the enterprise systems realm. (For instance, instead of Hay's concept of *Contract*, Fowler's more general concept of *Accountability* [4] has been used.) In [3] also the basic content of the packages can be found.

The top-level patterns can be specialized in various ways. In Figure 2 our specialization for a class of financial information systems is sketched out. Here, *Accountabilities* are elaborated to *Obligations*[3], *Claims*, *Payments* and other packages. Similarly, *Objects* are specialized to *Accounts* and *Operations* are specialized to *Accounting Transactions*. Analysis patterns of this level can be utilized for a wide range of domains associated with financial information systems.

The diagram in Figure 2 also shows two examples of the third level that are specialization of the previous level. Both are from the insurance industry. The first example addresses companies with self-employed brokers selling insurance products while the second example addresses insurance companies. For instance, in the insurance brokerage *Claims* are refined to *Brokers' Provisions* while in insurance *Claims* are refined to *Insurance Claims*. A simple example how the specialized model of the insurance contract (policy) with its premium prescriptions fits the more general model of obligation and its claims can be found in [7].

There are other examples of analysis patterns of the third level known to us and not shown in Figure 2. They include domains of both traditional private companies and public/government institutions. Examples of the former are bank products (e.g. bank cards) and factoring; examples of the latter are social insurance, health insurance and state benefits. Some of them have been published[4] in [8], [7] and [9]. Although, many times public and government institutions use other types of obligations rather than contracts (e.g. applications or registrations) the 'nature of business' is similar to private companies and the same second level concepts can be utilized. For instance, in the realm of state benefits *Obligations* are specialized to *Applications* and *Claims* to claimable state *Benefits*.

There are types of the second and the third levels other than for financial systems. For instance, one can consider Hay's patterns for *Work Orders*. General *Work Orders* from [3] are patterns of the second level of hierarchy. They can be seen as specialized *Accountabilities*[5] associated with *Parties* and *Operations*. These patterns can be specialized for example to maintenance work orders for road maintenance or gas pipelines maintenance [8], emergency work orders in gas networks or work orders in manufacturing [9]. As it was shown in [9] road maintenance and gas pipelines maintenance are so similar in their nature that a sublevel (maintenance of a service network) of the second level of the hierarchy can be created. Analogously, other sublevels of the specialized analysis patterns can be built.

3. HORIZONTAL RELATIONSHIPS

There exist horizontal relationships among packages and analysis patterns on each of the levels described in the previous section.

In Figure 3 relationships among pattern packages of the first hierarchical level are shown. The relationships represent UML dependency relationships. It is clearly visible that *Parties* and *Objects* are fundamental packages while *Accountabilities* is the supplementary package.

The *Operations* package can play an alternative role[6] which may be either:

1. Subjects of *Accountabilities* (e.g. a service to a party, work order activities, etc.) or
2. Operations performed with *Accountabilities* (e.g. change of a contract, credit card transactions, etc.).

Figure 4 shows an example of the third level relationships, brokers' provisions in particular. These relationships follow the relationships of the first and the second levels: *Provisions* (i.e. *Claims*) are dependent on *Contracts* (i.e. *Obligations*), while *Contracts* (as specialized *Accountabilities*) are dependent on *Brokers* (specialized *Parties*). The system, such as those we have developed for some brokerage companies, is based on accounting transactions. That is why *Payments* are dependent on *Brokers' Accounting* based on calculated *Brokers' Provisions*. Alternatively, if accounting were not used, *Brokers' Payments* would be the supplementary package of *Brokers' Provisions*.

[2] *Parties, Things of the Enterprise, Procedures and Activities, Contracts*.

[3] *Obligation* is our concept from [7] that is more general than the concept of *Contract*. It includes also other types of obligations than traditional business contracts.

[4] Although they are not in the form of hierarchical patterns.

[5] Originally in [3] there are no associations between *Contracts* and *Work Orders* and *Work Orders* are part of the *Procedures and Activities* anchor.

[6] An alternative might be to include the first option to the *Object* package. In this way 'objects' would mean 'roles', i.e. not only tangible and conceptual objects but activities and parties as well. This is, however, quite confusing for practitioners.

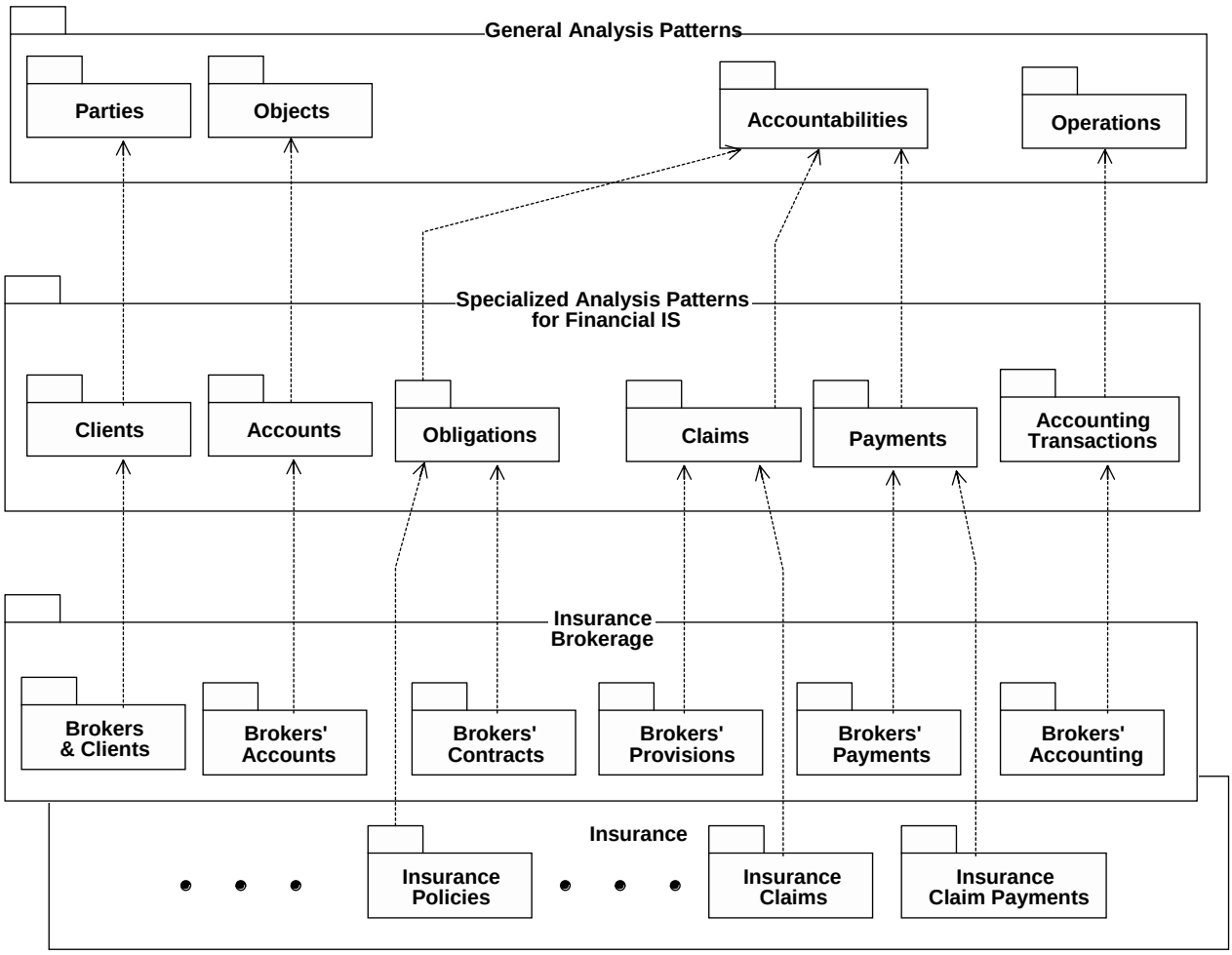

Figure 2 *Examples of hierarchical patterns*

4. RATIONAL

In this paper hierarchical analysis patterns have been introduced. We believe, however, that hierarchization of patterns is not restricted to analysis patterns only. For example, some of the J2EE patterns [1] might be seen as specialization of Gang of Four's [5] and other 'general' design patterns. For instance, *Composite View* is the special case of the *Composite* pattern; *Session Façade* is the special case of *Façade*. This is, however, a challenge for future research.

REFERENCES

[1] Alur, D., J. Crupi, D. Malks. **Core J2EE Patterns: Best Practices and Design Strategies**, Prentice Hall, 2001.

[2] Coad, P. **Object Models: Strategies, Patterns and Applications**. Yourdon Press, 1997.

[3] Hay, D. **Data Model Patterns: Conventions of Thought**, New-York: Dorset House, 1996.

[4] Fowler, M. **Analysis Patterns: Reusable Object Models**, Reading, MA: Addison-Wesley, 1997.

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

[6] Keller, W. **Some Patterns for Insurance Systems**, PLoP'98, also at: http://ourworld.compuserve.com/homepages/WofgangWKeller/

[7] Sesera, L. **A Recurring Fulfillment Analysis Pattern**, Pattern Languages of Programs Conference, *2000*. http://jerry.cs.uiuc.edu/~plop/plop2k/proceedings/proceedings.html

[8] Sesera, L. **Analysis Patterns**, (invited talk.) in: SOFSEM'2000, Lecture Notes in Computer Science series, Vol. 1963. Springer Verlag, 2000.

[9] Sesera, L., A. Micovsky, J. Cerven, J. **Data Modeling in Examples**, Grada, 2001 (in Czech).

[10] Smolarova, M., P. Navrat, M.Bielikova. **Abstracting and Generalising with Design Patterns**, Advances in Computer and Information Sciences '98, IOS Press, 1998.

[11] Sowa, J.F., J.A. Zachman. **Extending and Formalizing the Framework for Information Systems Architecture**, IBM Systems Journal, 1992.

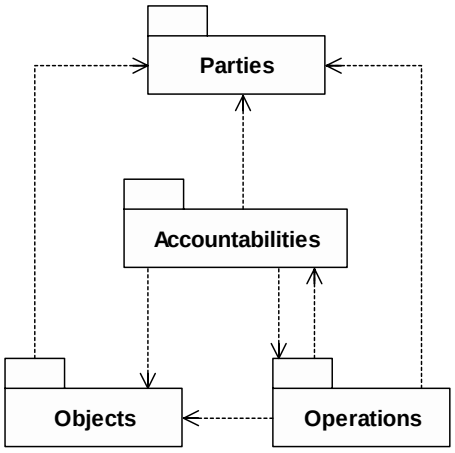

Figure 3 *Relationships among pattern packages of the first level*

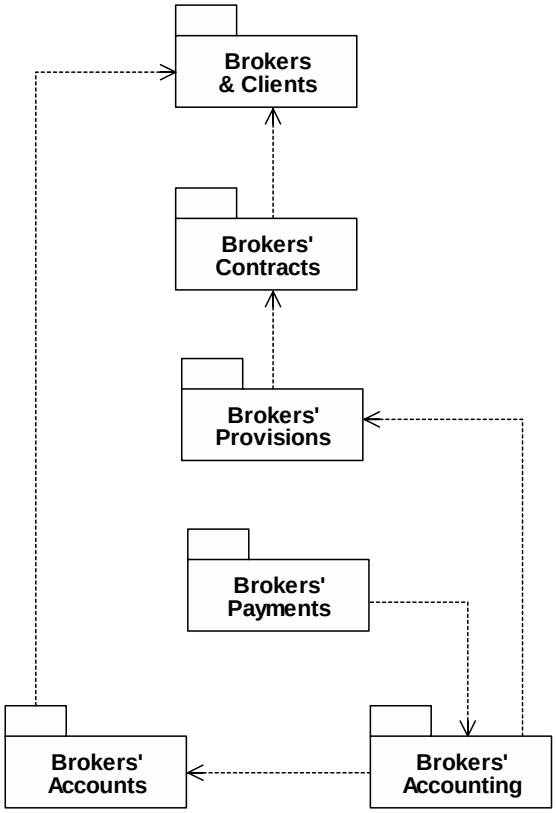

Figure 4 *Relationships among Brokers' provisions packages*