Patterns in Object-Oriented Analysis

Nicolas Blaimer, Andreas Bortfeldt, Giselher Pankratz

Diskussionsbeitrag Nr. 451

Diskussionsbeiträge der Fakultät für Wirtschaftswissenschaft
der FernUniversität in Hagen

Herausgegeben von dem Dekan der Fakultät

Patterns in Object-Oriented Analysis

Nicolas Blaimer, Andreas Bortfeldt, Giselher Pankratz

Abstract:

Analysis patterns have recently shown their great potential to enhance the set-up of models in object-oriented analysis. This report attempts to summarise the present state of literature on the subject of analysis patterns. It is organised in two main parts. In the first part, we present a comprehensive overview of the existing literature considering about one hundred mostly English written contributions. In the second part, we address some fundamental methodological issues related to both the creation and the application of analysis patterns. We study pattern documentation and organisation – e.g. in pattern catalogues – as important factors which the successful re-use of analysis patterns depends on. Furthermore, we consider and generalise numerous existing approaches to the application of analysis patterns. We show that all these procedures can be subsumed under two methods that are described in a formal fashion. Within the first method, patterns are integrated in the analysis model from the very first, whereas in the second method patterns are integrated only after a complete model has been established. Since the first method is of particular interest, it is illustrated by means of a substantial and nontrivial example. We also give special attention to the issue of traceability of patterns incorporated in a model, and we describe a promising approach from recent literature in detail.

Keywords:

Object-oriented analysis, analysis pattern, pattern template, pattern catalogue, pattern application.

Fakultät für Wirtschaftswissenschaft, FernUniversität in Hagen
Profilstr. 8, D-58084 Hagen, BRD

Tel.: 02331/987-4433
Fax: 02331/987-4447
E-Mail: andreas.bortfeldt@fernuni-hagen.de

Please cite as: Blaimer, N.; Bortfeldt, A. and Pankratz, G.: Patterns in Object-Oriented Analysis, Working Paper No. 451, Faculty of Business Administration and Economics, University of Hagen (Germany), 2010.

Patterns in Object-Oriented Analysis

Nicolas Blaimer, Andreas Bortfeldt, Giselher Pankratz

1 Introduction

Object-oriented software development has been state of the art for some years (BALZERT 2005, p.V). Commercial software development is now inconceivable without the purely object-oriented programming languages Java and C#, not least because of the comprehensive frameworks available for these languages. Unified Modelling Language (UML) has established itself as the standard notation for object-oriented modelling. Unlike in the past, when authors used their own notation in literature, which then needed to be explained, UML is used in all publications nowadays; it has become an indispensable tool for object modelling: that is, object-oriented analysis and object-oriented design.

In order to remain competitive, today's companies depend on IT as active support for their operational and strategic targets. This means that software projects are becoming larger and more comprehensive. Because the market is changing and developing with ever-increasing haste, IT has to adapt itself to the new conditions more quickly and more flexibly in order for the company to develop or hold its market position.

Against this background, software developers face the challenge of developing ever more quickly ever more complex applications – fault-free ones, naturally. A tool that can help with this challenge is the concept of re-use. Re-use is possible in all phases of software development, allowing in principle the re-use of parts of the analysis and design model as well as the code of former projects in new projects. This method, however, can be prone to faults, as in most cases the code taken over has to be adapted. The use of patterns has turned out to be an important concept, one that is practised partly as an alternative, partly as a supplement.

Various patterns have been developed for each phase of software development that involves the concept of re-use: analysis patterns for the analysis phase, design patterns for the design phase, and idioms for the implementation phase. The distinguishing feature of patterns is that they provide expert knowledge for certain problems that has been formatted for re-use, allowing the faster development of more flexible applications which, through the use of tried and tested solutions, contain fewer faults. The analysis patterns in particular have an enormous cost-cutting potential when it comes to avoiding faults, as in software development it is the conceptual faults that are the most expensive.

Analysis patterns are the subject of this study. On the one hand it will give an overview of literature on analysis patterns, while, on the other hand, providing a detailed investigation of some fundamental aspects of a successful application of analysis patterns.

It seems that the development of design patterns is more advanced than that of analysis patterns, as there is more literature available on the subject of design patterns, at least in volume.

The extensive literature overview in chapter two clarifies the current stage of the development of analysis patterns.

Pattern documentation and organisation were identified as important factors on which the application and re-use of analysis patterns depend. Or, put somewhat bluntly, I cannot apply what I have not found, or apply what I have found but cannot understand. The problem at the moment is not the direct application of an analysis pattern, but its accessibility. Patterns can be applied if there

are a sufficient number of fully documented patterns in an organised structure available. The documentation and organisation of analysis patterns will be discussed in depth in chapter three in order to clarify this problem.

Finally, chapter four focuses on the methodology of the application of analysis patterns. As an extract from the literature two application methods are presented and an application example is provided for the development of an analysis model with the help of analysis patterns. The study finishes with a summary in chapter five of the results gained.

This study requires basic knowledge of object-oriented software development and UML. For this purpose, the reader is referred to BALZERT (2005) and UML (2008).

1.1 The Term 'Pattern'

The usage of the term 'pattern' in software development was inspired by the notion of architectural patterns introduced by Christopher Alexander (ALEXANDER et al. 1977). Patterns for the design phase and the analysis phase, and idioms for the implementation phase were developed.

RIEHLE and ZÜLLIGHOVEN (1996) give a general definition of patterns: 'A pattern is the abstraction from a concrete form which keeps recurring in specific non-arbitrary contexts.'

Martin Fowler (FOWLER 1997) defines in reference to analysis patterns: 'A pattern is an idea that has been useful in one practical context and will be probably useful in others'.

Another definition of analysis patterns comes from Kai Bender: 'Therefore, analysis patterns are those formalised triples of problem, solution and context, which solve domain-specific problems by encapsulating specialist knowledge' (translated after BENDER 1999, p. 19).

At first glance, the three definitions mentioned above characterise patterns somewhat differently. What they have in common, however, is the aspect of re-use. Initially, patterns should store knowledge and then make it available for re-use. From Fowler's and Bender's definitions it becomes clear that the knowledge is tried and tested expert knowledge, available for others to use in the form of a pattern.

The purpose of the pattern is, then, the re-use of the knowledge encapsulated in it in order to solve a particular problem. A pattern that is not re-used can be compared to a book that is not read.

The different types of patterns are more closely described below.

1.2 Types of Patterns in Object-Oriented Software Development

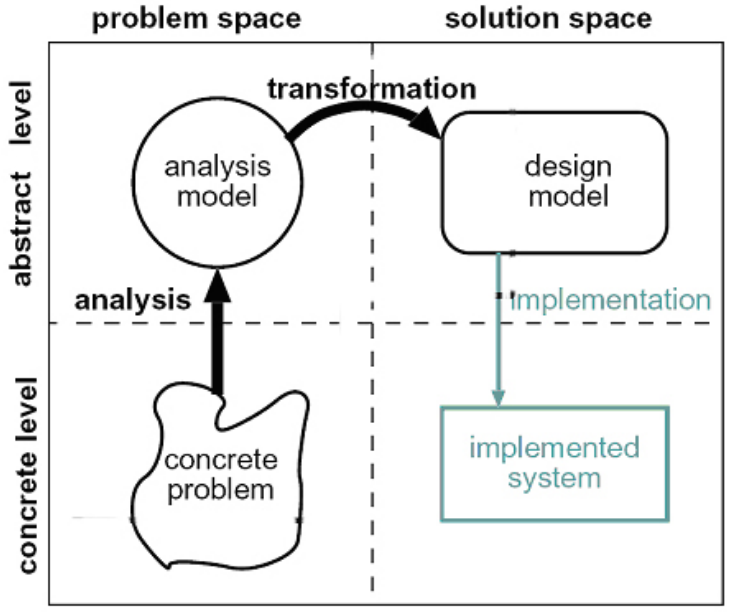
The basic phases of object-oriented software development are analysis, design, implementation, testing and maintenance (BORTFELDT 2001).

The analysis phase comprises two parts. First of all, what is required of the software system is defined and recorded in the so-called requirements specification. When the object-oriented paradigm is used, the analysis model consists of two sub-models, a static and a dynamic.

The software-technical realisation of the software system is specified in the subsequent design phase. To do this, the application architecture and other aspects of the realisation (target platform, programming language, data storage etc.) are determined. Afterwards, the design model is constructed; this is already detailed enough for the implementation to be begun straight away.

Fig. 1.1 shows the step from analysis to design. The analysis model is a result of the abstraction of the concrete problem that is to be solved through the software system; both belong to the so-called problem space. The design model is the abstraction of the implemented system; they both belong to

the solution space. The transition from analysis model to design model requires a transformation from the problem space to the solution space.



Source: HAHSLER 2001, p.18

Fig. 1.1. Classification of analysis and design

In the implementation phase the design is converted into executable programme code. The result is the implemented system.

For each of the phases mentioned above, different types of patterns have emerged. Fig. 1.2 assigns the various types of patterns to the basic phases of object-oriented software development and shows the respective degree of abstraction.

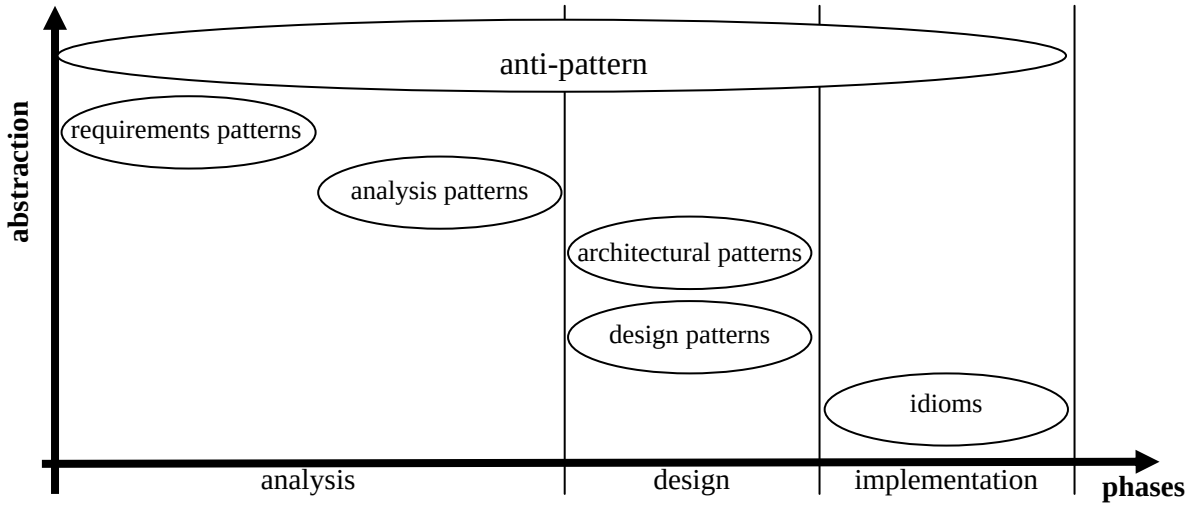


Fig. 1.2. Patterns in the phases of software development

In the following, the different types of patterns are described in some detail.

1.2.1 Anti-Patterns

Unlike other patterns that offer well-proven solutions to recurrent problems, anti-patterns describe negative examples of solutions: they describe behaviour that has been repeatedly observed in practice and its negative effects and point the way to better solutions. Awareness of anti-patterns helps to avoid their negative effects. Anti-patterns can be found in the whole process of software development: in project management, analysis, architecture, design and programming. Anti-patterns will not be discussed any further in the following (the reader is referred to BROWN et al. 1998).

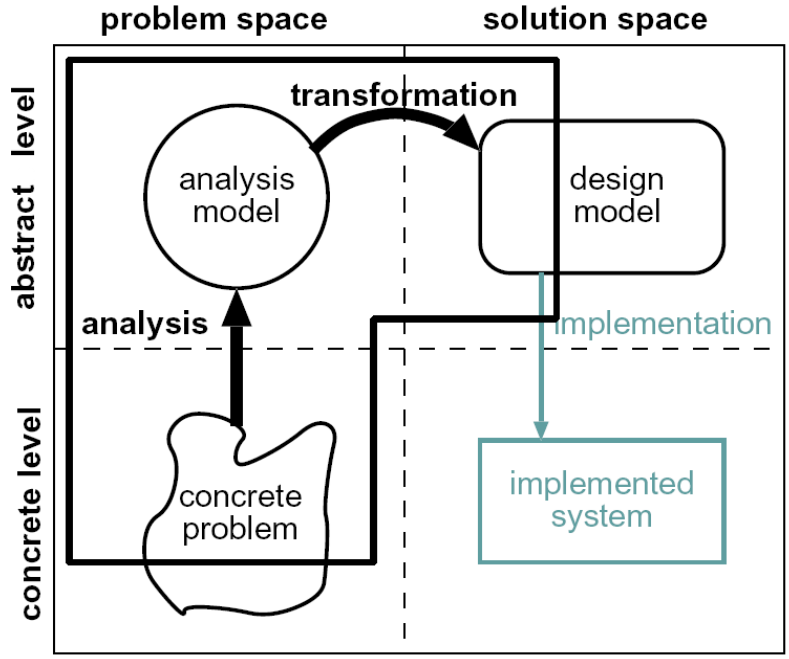
1.2.2 Requirements Patterns

Requirements patterns support the creation of the requirements specifications. They specify generic behaviour of systems on a high level of abstraction and thereby help to reach precise and better requirements descriptions for a system more quickly. Requirement patterns are not the subject of this report to any great extent.

1.2.3 Analysis Patterns

Analysis patterns support the creation of the conceptual model of an application. An analysis pattern provides a re-usable, well-proven model representation of a recurrent subproblem. Thus, analysis patterns facilitate the proper transformation of a concrete problem into an analysis model (Fig. 1.3). Usually, an analysis pattern needs some minor modifications before being transferred into the analysis model at hand (pattern instantiation). An analysis pattern at least specifies the static components of the corresponding model fragment; however, in an ideal case an analysis pattern also contains a description of the dynamic aspects of the respective subproblem.

An analysis pattern consists of at least two classes, which together form a semantic unit for representig a particular (sub-)problem. It is a component above class level.



Source: HAHSLER 2001, p. 45

Fig.1.3. Application scope of analysis patterns

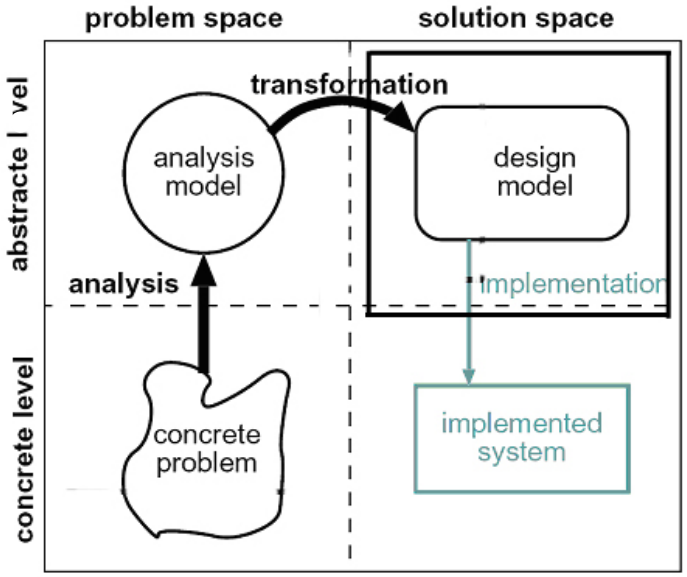
Analysis patterns can be classified into domain-specific and domain-independent patterns. Domain-specific patterns provide model fragments of subproblems that occur in a specific domain, for example, insurance companies, energy suppliers etc. Domain-independent (or domainless) analysis patterns are not limited to one domain; instead, they model problems that are are cross-domain.

1.2.4 Architecture Patterns

Software architecture patterns describe basic organisational principles for designing the structure of a software system. They support the choice of an appropriate software architecture by providing tried and tested solutions for structural and organisational issues. A software architecture pattern contains best practise solutions for decomposing a software system into sub-systems and describes the responsibilities and relationships between the sub-systems (AKAEM 2008).

1.2.5 Design Patterns

Design patterns support the creation of the design model. They provide solutions for recurring problems that occur during the transition from the analysis model to the design model (Fig. 1.4). As distinct from analysis patterns which are closely related to a concrete subproblem, design patterns contain more generic model fragments which are dedicated to rather technical or implementation-related issues. Similar to analysis patterns, design patterns require some modification when being integrated into the design model at hand.



Source: HAHSLER 2001, p. 28

Fig. 1.4. Application scope of design patterns

Design patterns describe the respective solution in terms of classes and their relationships i.e. they are used to specify components and sub-systems in detail. Thus design patterns are clearly distinguished from architecture patterns, which describe solutions from a high-level perspective, i.e. in terms of components and sub-systems.

1.2.6 Idioms

Idioms are patterns on the lowest level of abstraction. They show for a given programming language how certain recurring programming tasks can be solved by the means of that language. Idioms will not be dealt with further in this report.

1.3 Terminology

In the remainder, some technical terms relating to object-oriented patterns are used. These are explained below.

- Pattern Language – a pattern language is a collection of interrelated patterns for solving a particular problem. The patterns are organised through their relationships within the pattern language.
- Pattern Template – a pattern template is a template for the description of patterns. A template determines a list of characteristics that have to be more closely defined in order to describe a pattern.
- Pattern Instantiation – pattern instantiation means the adaptation of a pattern to the concrete problem that is to be solved. Class names, class attributes and class operations are adapted and parts of the pattern that might no longer be needed are omitted.
- Ad Hoc Modelling – ad hoc modelling or manual modelling means the building of a model without pattern support.
- Framework – the term ‘framework’ is used in the sense of conceptual framework. Conceptual frameworks are used in science to describe possible approaches to complex problems.

2 Analysis Patterns in Literature

The following should give the widest possible overview of literature dealing with analysis patterns. The purpose was to make the overview as comprehensive as possible; nevertheless, it is possible that not all the relevant sources were found. Only sources in German and English have been taken into account.

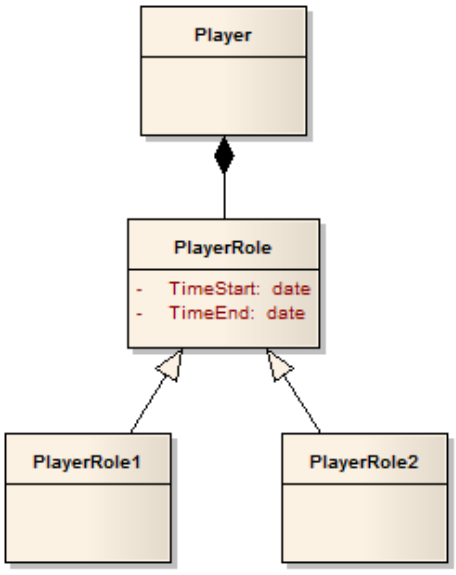
Each of the following chapters gives a summary of the work of an author or a group of authors. They are arranged chronologically according to the year of the first work by the author or the group of authors.

The summaries of the works are limited to the most important aspects of the parts relevant to analysis patterns. In connection with the literature catalogue, they offer the reader a starting-point for his or her own studies.

2.1 Peter Coad, David North, Mark Mayfield

In his article 'Object-Oriented Patterns' (COAD 1992), Peter Coad was the first person to deal with patterns for object-oriented analysis (OOA). He presents six patterns in the article, differentiated at this juncture not into patterns for the analysis and design, but depicted as higher-level building blocks for OOA and OOD (object-oriented design). Coad does not yet use today's terms 'analysis' and 'design' patterns; he labels his patterns as 'object-oriented patterns' and defines them as follows: 'An object-oriented pattern is an abstraction of a doublet, triplet, or other small grouping of classes that is likely to be helpful again and again in object-oriented development.'

The six patterns presented have a low complexity and can be used across domains. Fig. 2.1 shows the 'Roles Played' pattern (transcribed from Coad's notation into UML).

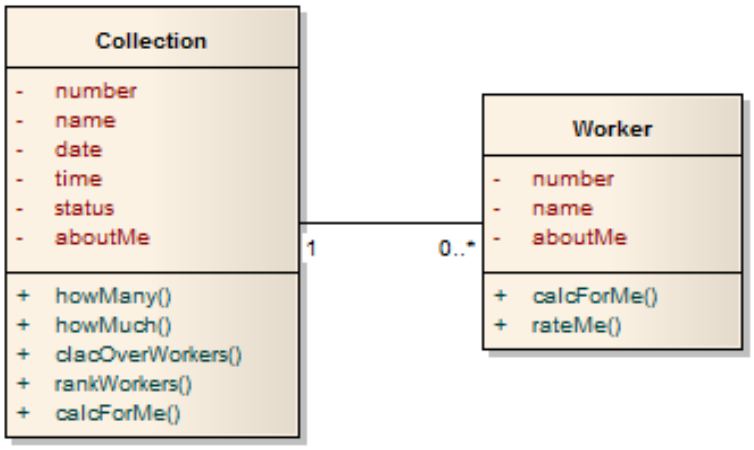


Source: based on COAD 1992, p. 155

Fig. 2.1. 'Roles Played' pattern

Together with David North and Mark Mayfield, Peter Coad continues his work in the book 'Object Models – Strategies, Patterns & Applications' (COAD et al. 1995). Here, with the aid of extensive examples, the authors introduce their method for the development of object-oriented analysis models. In doing so, they deal with the application of patterns and introduce the first catalogue with

patterns (‘pattern handbook’). The catalogue comprises 31 patterns, the first pattern of the catalogue introducing the so-called ‘fundamental pattern’ (Fig. 2.2).



Source: based on COAD et al. 1995, p. 423

Fig. 2.2. ‘Collection-Worker’ pattern – The fundamental pattern

Coad et al. describe this pattern as fundamental, as all their other patterns are constructed following this as an elementary model. All patterns consist of two classes, one of which (the ‘collection’) always is linked to several objects of the other class (the ‘worker’).

The other 30 patterns are divided into the categories ‘transaction patterns’, ‘aggregate patterns’, ‘plan patterns’ and ‘interaction patterns.’ The patterns introduced in COAD (1992) are not included in this catalogue.

The patterns introduced in COAD et al. (1995) distinguish themselves through both their rather low complexity and their domain-independence. Thus, on the one hand, they can easily be identified, but on the other hand, their integration into an analysis model makes high demands on the transfer abilities of the user. Due to their low complexity, several patterns must be combined in order to achieve a more complex model.

Coad et al. already use a structured template with six attributes for the description of their patterns (see chapter 3.1.1).

2.2 Suzanne Robertson, Kenneth Strunch

In their article ‘Reusing the Products of Analysis’ (ROBERSTON and STRUNCH 1993) Robertson and Strunch already use the term analysis pattern, although as a synonym for the term ‘requirements pattern.’ Their definition of an analysis pattern is ‘any part of a requirements specification that originates in one project and can be re-used in one or more other projects’.

With the help of an example from an insurance company, Robertson and Strunch demonstrate how patterns were recognised, documented and then, saving an enormous amount of time, re-used in the analysis phase of other projects.

Robertson and Strunch have identified four abstraction stages for analysis patterns. The fourth and highest abstraction stage comprises patterns that reproduce knowledge about a particular domain (cross-domain patterns are not taken into consideration). In the third stage, the knowledge of a special field is added; in the second the knowledge of a particular organisation and in the first stage the knowledge of a special project. They made the patterns available to their analysts for their work in an analysis pattern book.

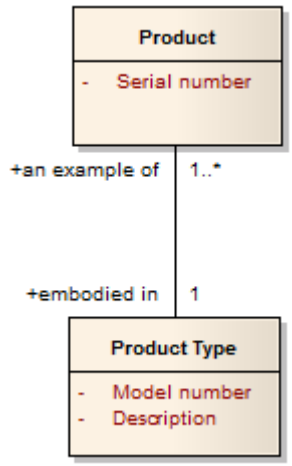
According to Robertson and Strunch, the concept of re-use in analysis must develop itself into a culture. This applies in particular to companies and organisations which do not yet accept that making patterns available or exchanging them is not detrimental to their business, but a benefit for everyone.

For their example patterns Robertson and Strunch do not use object-oriented notation but the notation of structured analysis, Entity Relationship Models (ERM) and Data Flow Diagrams (DFD). They only use their example patterns for illustration and do not explain them further.

2.3 David C. Hay

In his book 'Data Model Patterns – Conventions of Thoughts' (HAY 1995) David C. Hay presents data-model patterns for the modelling of business problems. His data-model patterns are not ready-to-use solutions but the starting point for analysis: Hay sees his book as a model-building kit that provides the analyst with the necessary tools.

To describe his patterns, Hay uses a non-formalised textual description and the Oracle CASE* method. Although this notation can be transferred easily into a UML class diagram (Fig. 2.3), the lack of a dynamic component becomes clear through the absence of operations in Hay's data-centric pattern models.



Source: based on HAY 1995, p. 48

Fig. 2.3. 'Products and Product Types' pattern

Hay starts with developing cross-domain patterns of low complexity that can be used anywhere in an enterprise. Partly building on these, he develops more complex patterns that describe special problem areas. Unlike in COAD et al. (1995), Hay's patterns offer in each case a larger 'ready-made' model segment for the instantiation in the analysis model.

2.4 Dirk Riehle, Heinz Züllinghoven

'Understanding and Using Patterns in Software Development' by Riehle and Züllinghoven (RIEHLE and ZÜLLINGHOVEN 1996) deals with patterns in software development. Not only do they examine analysis patterns, which they call conceptual patterns, but also design patterns and idioms. Consequently, their definition of the term pattern is a general one: 'A pattern is the abstraction from a concrete form which keeps recurring in specific non-arbitrary contexts.'

One of the most important aspects for Riehle and Züllinghoven is the (re-)use of patterns. Patterns, they claim, can only be exploited to their full potential if they are accepted and re-used by others.

According to Riehle and Züllinghoven, an analysis pattern should preferably be limited to one domain, as analysis patterns that are too general are of limited use because they are harder to transfer onto the concrete problem. On the other hand, if a pattern is too specific it will be limited to one or a few projects and cannot be used on a wider scale. Therefore, a balance between abstraction and specialisation must be found.

Riehle and Züllinghoven do not yet use object-oriented techniques for the analysis phase; as a consequence they propose a purely textual form for the description of patterns.

To arrange the patterns in accessible form, they suggest organising them into sets of related patterns. As each pattern is related to other patterns, not only to ones from the same phase of software development but also to those of subsequent phases, a directed graph is proposed for organising them. The patterns are represented by the nodes of the graph, with the conceptual patterns at the beginning and the idioms at the end.

2.5 Martin Fowler

In his book 'Analysis Patterns – Reusable Object Models' (FOWLER 1997) Martin Fowler presents a pattern catalogue of 63 analysis patterns. His definition of the term pattern reads as follows: 'A pattern is an idea that has been useful in one practical context and will be probably useful in others'. Fowler's definition makes clear that he puts great emphasis on the tried-and-tested quality of his analysis patterns. For him, the key element of a pattern is that it was discovered during the everyday development process and is not an academic invention. All patterns presented by him were developed in one or several projects that he supervised.

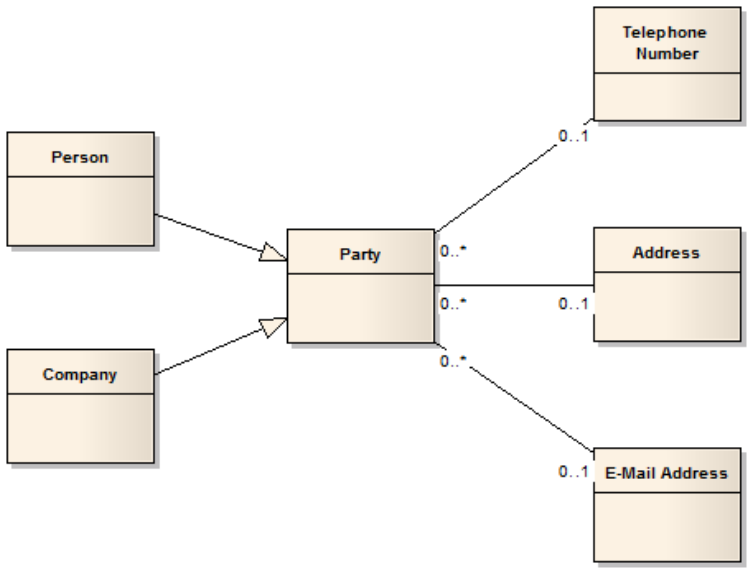
Fowler's book is the most cited in analysis pattern literature; the majority of the subsequent publications contain references to FOWLER (1997).

Most of Fowler's patterns can be used across domains, although he groups them into the domains in which he 'discovered' them (Accountability, Observation and Measurements, Observations for Corporate Finance, Referring to Objects, Inventory and Accounting, Using the Accounting Models, Planning, Trading, Derivate Contracts, Trading Packages).

Fowler does not describe his patterns with a structured template; he prefers, instead, a purely textual description, along with the graphical depiction in his own object-oriented notation. For Fowler a structured template such as in GAMMA et al. (1995) contains too many disadvantages affecting its descriptive potential (FOWLER 1997, p. 6). This could be one of the reasons why analysis patterns are not nearly as well documented, and in the same homogenous manner, as design patterns. In their standard work GAMMA et al. (1995) provide a structured template for design patterns which was followed by other authors. Fowler has meanwhile recognised the advantages of a structured template (FOWLER 2007).

The graphic notation Fowler uses to describe his analysis patterns supports the object-oriented paradigm. Unlike Hay's, Fowler's patterns contain a dynamic part. For example, Fowler adds sequence diagrams to some of his patterns. Compared to Coad's patterns, the complexity of Fowler's analysis patterns is rather high; this is also governed through the somewhat lower degree of abstraction, as Fowler describes his patterns all in the context of their original domains.

The following diagram (Fig. 2.4) shows one of Fowler's most well known analysis patterns, the so-called party pattern.



Source: based on FOWLER 1997, p. 18

Fig. 2.4 ‘Party’ pattern

2.6 Eyðun Eli Jacobsen, Bent Bruun Kristensen, Palle Nowack

In their article ‘Patterns in the Analysis, Design and Implementation of Frameworks’, (JACOBSEN et al. 1997) Eyðun Eli Jacobsen, Brent Bruun Kristensen and Palle Nowack characterise the use of patterns at the different stages of object-oriented software development. They state that patterns can be used in all phases of the software-development process and they mainly differentiate themselves according to the task that is to be solved in the respective phase.

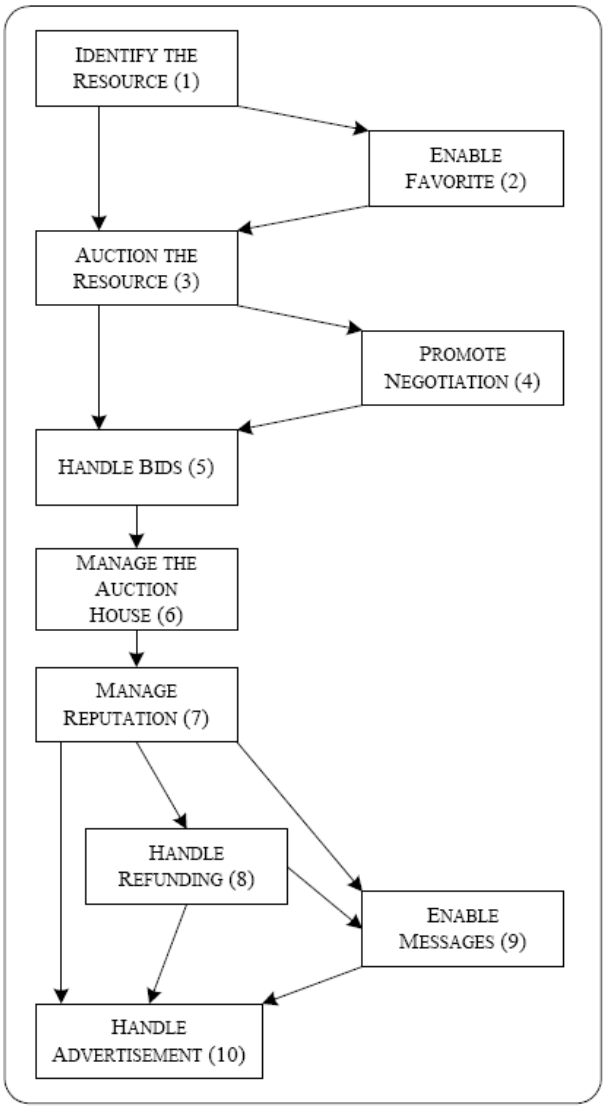
According to Jacobsen et al., the application of analysis patterns can be generally characterised as direct since unlike in the design phase, the technical implementation still plays no role, and the structure of the solution that is provided in the pattern can be used directly. Accordingly, the application of design patterns is determined by technical aspects.

2.7 Rosanna T. V. Braga, Fernao S. R. Germano, Paulo Cesar Masiero, Reginaldo Ré

In their publication ‘A Confederation of Patterns for Resource Management’ (BRAGA et al. 1998) Rosanna T.V. Braga, Fernao S.R. Germano, Paulo Cesar Masiero and Reginaldo Ré introduce three analysis patterns for resource management. The patterns are described in a structured template and the graphical models are generated with UML.

With ‘A Pattern Language for Business Resource Management’ (BRAGA et al. 1999) Braga et al. expand the three patterns from BRAGA et al. (1998) to a pattern language with 15 analysis patterns.

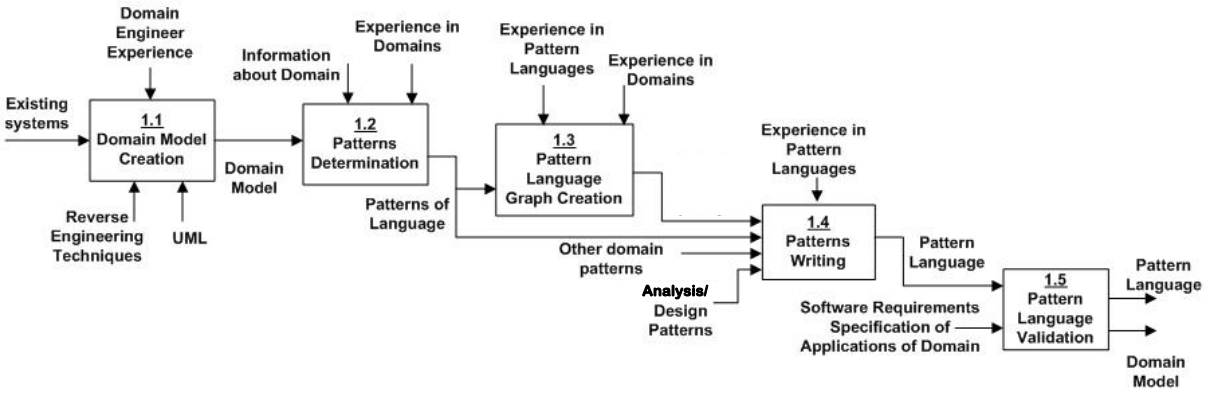
In ‘A Pattern Language for Online Auctions Management’ (RÉ et al. 2001) Reginaldo Ré, Rosanna T.V. Braga, Paulo and Cesar Masiero publish a pattern language for online auctions. The pattern language comprises ten analysis patterns (Fig. 2.5); these, as in BRAGA et al. (1998), are likewise documented in a structured template using UML as graphic notation.



Source: RE et al. 2001

Fig. 2.5. Pattern language for online auctions

In 'A Process to Create Analysis Pattern Languages for Specific Domains' (BRAGA et al. 2007) Braga et al. condense their experience with the generation of pattern languages and introduce their process for the creation of a pattern language.



Source: BRAGA et al. 2007

Fig. 2.6. Process for creating a pattern language

With this they answer the question of how a pattern language is developed and how patterns are defined, so that they can be easily used for the modelling of systems in any domain. For the application of their process, Braga et al. assume that the user has extensive knowledge of the respective domain.

Fig. 2.6 shows the process suggested by Braga et al. First, a detailed model of the domain is generated, capturing the main functionalities of applications in the domain observed. In the next step, a list of patterns is generated on the basis of the domain model. The identified patterns are then arranged in a pattern graph. The graph specifies the relationships of the patterns to each other, which pattern should be applied and when, and which patterns are optional. Finally, the patterns are written and afterwards the pattern language is validated.

2.8 Eduardo B. Fernandez

In his publication 'Building Systems using Analysis Patterns' (FERNANDEZ 1998) Eduardo B. Fernandez introduces his perspectives on the application of analysis patterns.

Fernandez's interest is concentrated on larger patterns. He is the first to suggest working with larger patterns, categorising the patterns of Coad and Fowler as smaller patterns. The larger, more complex patterns are more domain-specific and are, according to Fernandez, best applied through analogy; that is, they are transferred and adapted from an application model onto a new model. For this, Fernandez gives an example of a pattern that was gained from a model for a computer workshop and then transferred to a patient-administration for a hospital. Fernandez does not provide a general description of this process. Thus a (good) application can be used to create new complex analysis patterns.

In 'Stock Manager: An Analysis Pattern for Inventories' (FERNANDEZ 2000a) Fernandez presents a complex pattern for stock-keeping. He uses a structured template for the description of the object-oriented pattern and uses UML for the description of the model.

Eduardo Fernandez together with Xiaohong Yuan published further articles on the theme of analysis patterns (see 2.14).

2.9 Paul Johannesson and Petia Wohed

In their work, Paul Johannesson and Petia Wohed introduce the so-called 'Deontic Analysis Patterns' (JOHANNESSON and WOHED 1998). Here the so-called basic deontic pattern is the basis for all deontic analysis patterns. It describes abstractly the construction of a deontic pattern.

The central idea of the deontic pattern (from the Greek *deon*, meaning duty) is to support the correct modelling of object-responsibilities i.e. which object is to fulfil which task.

Along with the basic deontic pattern, Johannesson and Wohed present three more deontic patterns derived from it. For the depiction they use their own notation, similar to that of HAY (1995). The deontic patterns do not contain any dynamic elements; Johannesson and Wohed concede, however, that the patterns introduced are only cursorily described and must be extensively developed for use in a pattern catalogue. They see the basic deontic pattern as the starting-point for the creation of a pattern catalogue of deontic patterns as it should serve as a template for the creation of new patterns and help when patterns are checked to see that they are sound.

2.10 Jugurta Lisboa Filho, Cirano Iochpe, Kate Beard, Karla A. V. Borges

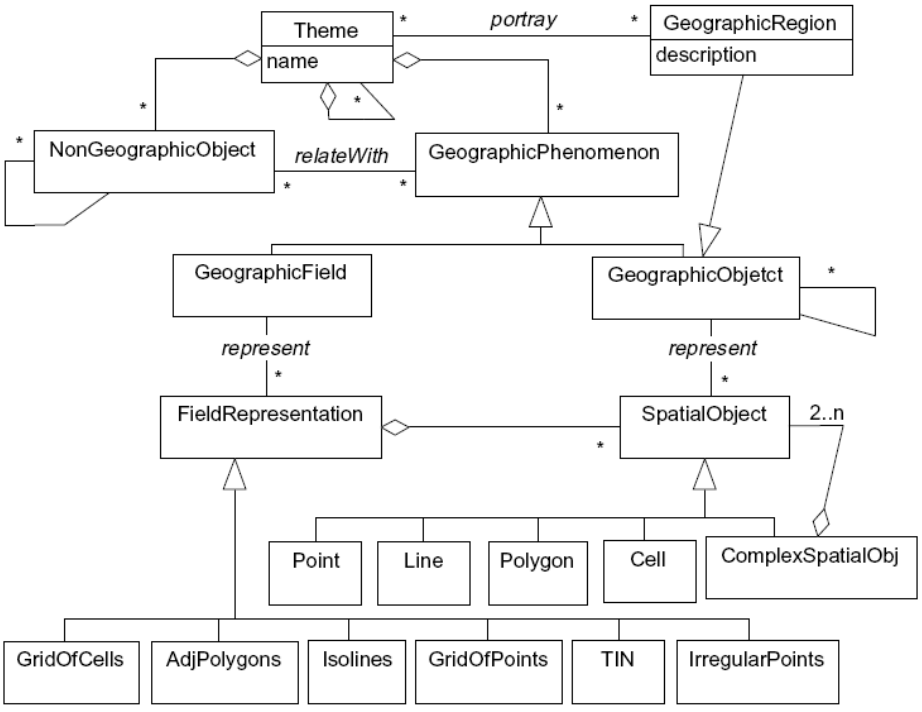
In their work 'Applying Analysis Patterns in the GIS Domain' (LISBOA et al. 1998) Jugurta Lisboa Filho, Cirano Iochpe and Kate Beard examine the applicability of analysis patterns in the domain of geographical information systems (GIS). According to their findings, previous analysis patterns (Hay, Coad, Fowler) are addressed to the developers of 'conventional' applications; appropriate analysis patterns, however, could also accelerate and improve the application development of GI systems.

Lisboa et al. lay stress upon the necessity of documenting recognised patterns in a structured template; since the patterns should correspond to the object-oriented paradigm, they use UML for graphic description.

The GIS analysis patterns are to be made accessible to all developers via a pattern catalogue in the internet. Lisboa et al. see the existence of a pattern catalogue, in which the patterns are well organised and easy to find, as the basic requirement for the dissemination and effective deployment of the analysis patterns in the GIS domain.

With the help of the pattern catalogue and after analysis of the requirements, appropriate patterns can then be sought and instantiated for the model.

In their subsequent work 'Specifying analysis patterns for geographic databases of a conceptual framework' (LISBOA and IOCHPE 1999) Jugurta Lisboa Filho and Cirano Iochpe introduce GeoFrame, a conceptual framework for GI systems. GeoFrame serves developers as an initial conceptual model by providing a diagram with base classes. Analysis patterns are then extracted and documented in their work from applications modelled with GeoFrame.



Source: LISBOA et al. 1999

Fig. 2.7. GeoFrame class diagram

In 'Analysis Patterns for GIS Data Schema Re-use on Urban Management Applications' (LISBOA et al. 2002) Jugurta Lisboa Filho, Cirano Iochpe and Karla A.V. Borges emphasise once more the significance of analysis patterns for the development of geographical information systems. Since a

GI system should not be developed any differently to any other information system, a precise analysis is granted a key role here also.

With the help of analysis patterns, less-experienced developers should be able to model better systems. For this purpose, the documentation of the patterns must meet certain standards of comprehensibility; to this end Lisboa et al. introduce the structured template they used (see chapter 4.1.3.4) and, on the basis of the UML stereotypes, define a UML enhancement for GIS modelling.

Lisboa et al. present three analysis patterns from the field of local-government administration, documented according to their template. Hand in hand with this, they call for each to publish their patterns and improve the patterns of others, since this is the only way that a complete pattern catalogue can be achieved.

2.11 Kai Bender

In his dissertation 'Analysemuster in der Architektur kommerzieller Informationssysteme' (BENDER 1999, in German) Kai Bender examines analysis patterns in commercial information systems architecture. Initially, he observes design patterns, and building upon this examines analysis patterns with the aid of the books by Hay, Coad and Fowler. An important conclusion of his investigation is that analysis patterns, as distinguished from design patterns, represent reusable specialist knowledge in model form and do not constitute the solution to a design-paradigm-specific construction problem (BENDER 1999, p. 149).

Bender's clear differentiation between analysis and design patterns also becomes apparent in his definition of analysis patterns: 'According to this, analysis patterns are those formalised triples of problem, solution and context that solve domain-specific problems by encapsulating specialist knowledge' (translated after BENDER 1999 p. 19).

Bender brings in the so-called analysis frameworks as a supplement to the previous analysis patterns. These distinguish themselves in comparison to the analysis patterns through a higher level of abstraction and describe basic connections that often only apply to a few fields of application (e.g. an analysis framework can contain organisation structures of a particular company and therefore only be deployable there). At the same time, however, he sees a problem, namely the possible restricted re-usability of solutions that are too specialised. Before the design phase can be started with an analysis framework, analysis patterns must first be applied.

Bender is the first author to suggest an additional phase in the analysis process in which patterns should be sought and instantiated.

In order to illustrate his examples Bender uses UML notation and conventional data-flow diagrams for the static and the dynamic view, respectively. Data-flow diagrams appear to him to be more practical than UML sequence diagrams, as at the moment of the analysis the problem is usually still not understood well enough for sequence diagrams to be applied.

2.12 Nathalie Gaertner, Bernard Thirion

In their work 'Grafcet: An Analysis Pattern for Event Driven Real-time Systems' GAERTNER and THIRION 1999) Nathalie Gaertner and Bernard Thirion describe an analysis pattern for modelling 'discrete event process controllers' Although they use the term analysis pattern based on Fowler, they regard the term 'business pattern' (meaning the same) as a more apposite one.

To describe their pattern Gaertner and Thirion use a structured template based on GAMMA et al. (1995). The graphical illustration is carried out in the UML notation.

2.13 Heide Balzert

In her book ‘Lehrbuch der Objektmodellierung – Analyse und Entwurf’ (BALZERT 1999), Heide Balzert deals with the phases of analysis and design in object-oriented software development; in the field of analysis she also introduces a catalogue of ten analysis patterns.

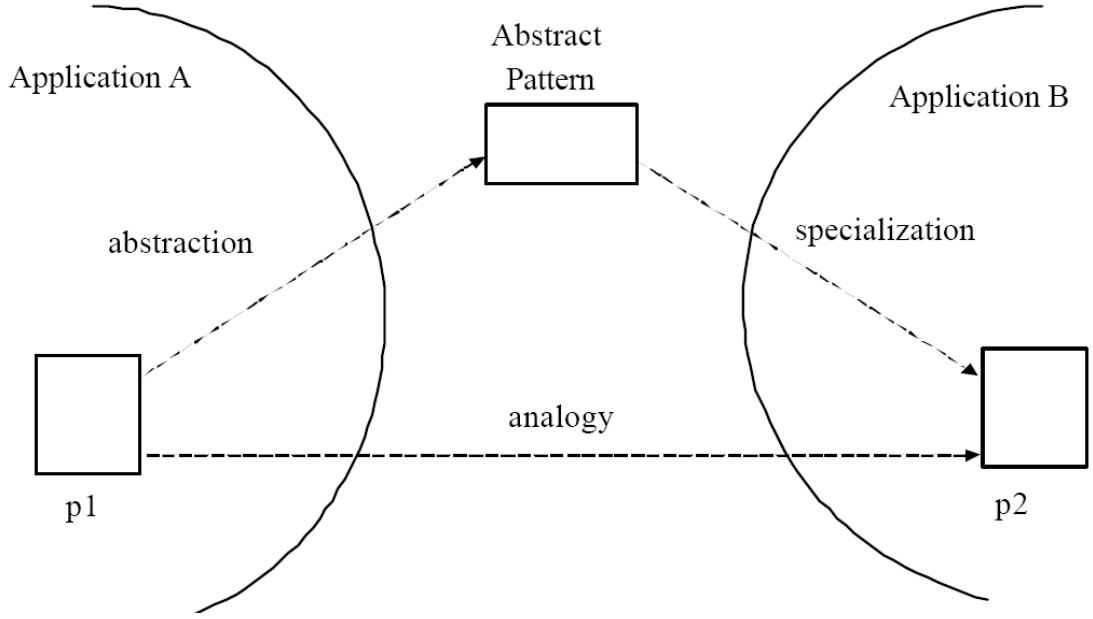
The patterns are of a general nature and not domain-specific; their complexity can be classified as rather low. According to Balzert, experienced developers apply this type of patterns to a certain extent automatically. The patterns are described using a textual description and a concrete example, presented as an UML model.

2.14 Eduardo B. Fernandez, Xiaohong Yuan

In their work of the same name (FERNANDEZ and YUAN 2000), Eduardo B. Fernandez and Xiahong Yuan introduce the new analysis-pattern type they developed, ‘Semantic Analysis Patterns’ (SAP). They define a semantic analysis pattern as ‘a pattern that describes a small set of coherent Use Cases that together describe a basic generic application. The Use Cases are selected in such a way that the application can fit a variety of situations.’

The analysis patterns of Fernandez and Yuan correspond to mini applications or connected parts of applications. Their complexity is greater than that of the patterns of Coad and Fowler.

According to Fernandez and Yuan, semantic analysis patterns can be instantiated in two ways: either the concrete original pattern is applied directly to a new situation through analogy, or an abstract pattern is specialised. Abstract patterns can be obtained from concrete patterns through abstracting the components. In Fig. 2.8 this methodology is summed up in a diagram once more. Fig. 2.9 and 2.10 show the instantiation of the abstract ‘admissions pattern.’



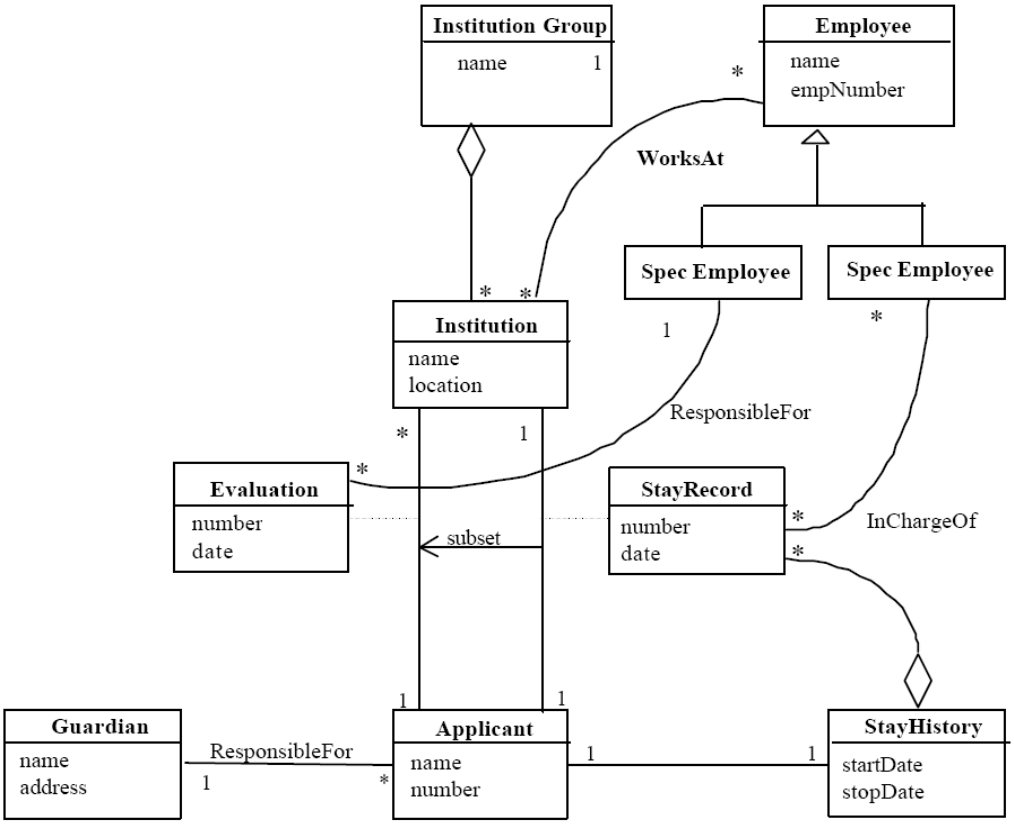
Source: FERNANDEZ and YUAN 2000b

Fig. 2.8. Pattern generation (abstraction) and pattern application (analogy or specialisation)

A concrete pattern is understood to be a model segment that on the one hand corresponds to the abstract pattern, but on the other hand also contains details (e.g. concrete attributes and operations) of an application.

In addition, Fernandez and Yuan introduce a modelling method on the basis of semantic analysis patterns (see 4.1.3.5).

Fernandez and Yuan use the object-oriented paradigm for defining their patterns and UML notation for depicting them. The complexity, but also the specialisation of their patterns, can be classified as high.



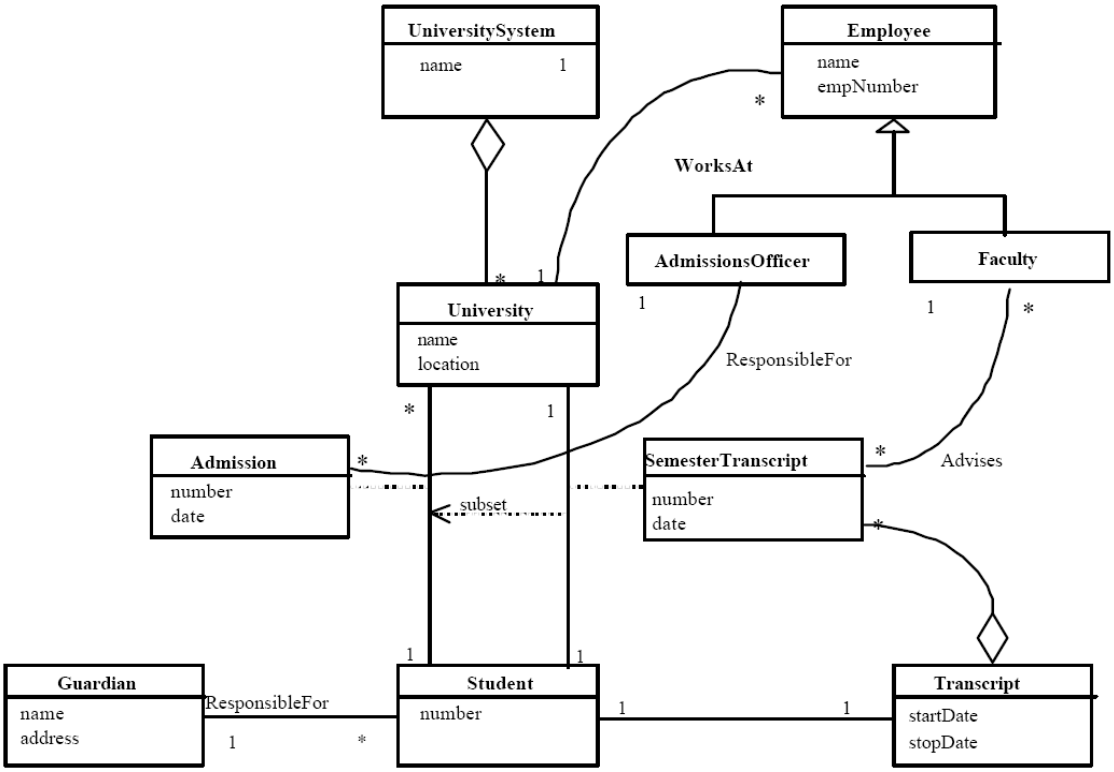
Source: FERNANDEZ and YUAN 2000b

Fig. 2.9. Abstract pattern: admissions pattern

In their subsequent work ‘Securing Analysis Patterns’ (FERNANDEZ and YUAN 2007), Fernandez et al. introduce a simple method for integrating application security into the conceptual analysis model. They do this in compliance with the security requirements of an application, by adding ‘security patterns’ (patterns that offer a solution for a specific security problem) to the semantic analysis patterns (see FERNANDEZ and YUAN 2000b). This way, security aspects of an application are already integrated before the actual generation of the analysis model, which has considerable advantages in comparison to the retrospective integration of security patterns into a finished conceptual model.

In addition to the works mentioned above several analysis patterns of the type ‘semantic analysis patterns’ were published by Fernandez et al., or with contributions from Eduardo B. Fernandez. All patterns are described in a structured template, and UML was used for graphical depiction of the models: ‘An Analysis Pattern for Reservation and Use of Reusable Entities’ (FERNANDEZ and YUAN 1999), ‘Analysis Patterns for the Order and Shipment of a Product’ (FERNANDEZ et al. 2000c), ‘An Analysis Pattern for the Repair of an Entity’ (FERNANDEZ and YUAN 2001a), ‘Patterns for Internet Shops’ (FERNANDEZ et al. 2001b), ‘The Sports Manager Pattern’ (FERNANDEZ and PAN 2002a), ‘The Account Analysis Pattern’ (FERNANDEZ and LIU 2002b), ‘An Analysis Pattern for the Request and Allocation of Limited Resources’ (FERNANDEZ and DAI 2002c), ‘An Analysis Pattern

for Course Management' (YUAN and FERNANDEZ 2003), 'Analysis Patterns for Patient Treatment' (SORGENTE et al. 2004), 'The SOAP Pattern for Medical Charts' (SORGENTE et al. 2005).



Source: FERNANDEZ and YUAN 2000b

Fig. 2.10. Instantiation of the admissions pattern: student admissions

2.15 Simon Pickin, Angeles Manjarrés, Gerson Sunyé, Damien Pollet, Jean Marc Jézéquel, Manuel Arias, Francisco Javier Díez

In 'Describing AI Analysis patterns with UML' (PICKIN et al. 2000) Simon Pickin et al. examine how object-oriented analysis patterns, and therefore the object-oriented paradigm, can be deployed in the domain of artificial intelligence (AI).

Previously, according to Pickin et al., mainly procedural programming and structured analysis were used in the AI domain. For this there exist analysis libraries consisting of 'generic tasks'. Pickin et al state that the generic tasks should be transferred into object-oriented analysis patterns, thereby helping the better-suited object-oriented paradigm to prevail, with the UML being the exclusive notation for the patterns. For illustration purposes, they convert the generic decision task into the generic-decision analysis-pattern in exemplary fashion.

In 'AI Analysis Patterns as UML Meta-Model Constructs' (MANJARRÉS et al. 2002) Angeles Manjarrés et al. show how the generic AI knowledge models (task templates) can be converted into object-oriented analysis patterns. They show this using the example of the assessment task template, which is converted into the assessment pattern.

Manuel Arias et al. continue to transfer knowledge and models of structured analysis in the AI domain to analysis patterns in 'Construction of a Development Environment for GPMs Based on OO Analysis Patterns' (ARIAS et al. 2003). 'GPM' stands for graphical probabilistic model.

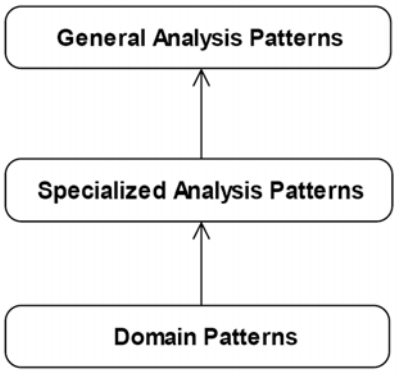
2.16 Lubor Sesera

In his work ‘Analysis Patterns’ (SESERA 2000a) Lubor Sesera characterises existing analysis patterns using the criteria of abstraction, flexibility and granularity. In the second part of his work he presents several analysis patterns that are based on the patterns by Hay and Fowler.

Sesera finds that in the literature available to him (mainly Coad, Hay and Fowler) the notion of analysis patterns differs somewhat from author to author. Therefore, he establishes the three criteria he uses to characterise analysis patterns, abstraction, flexibility and granularity.

Sesera’s patterns are not object-oriented; they are data-model patterns following the example of Hay or taken from his book (HAY 1995). For the depiction of his pattern examples Sesera uses an unstructured textual description, and UML for the depiction of the Entity Relationship Diagrams.

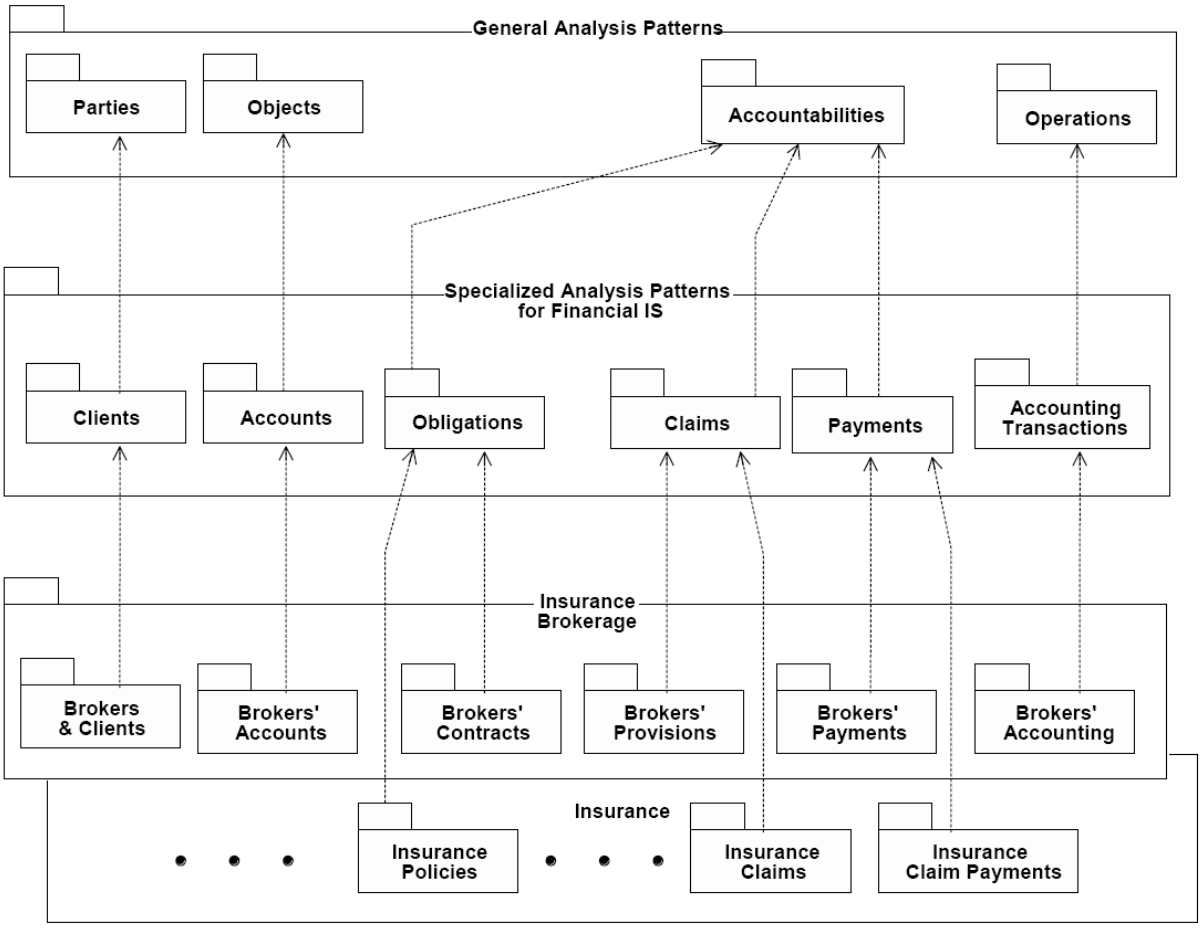
In Sesera‘s next work, ‘Hierarchical Patterns: A Way to Organise (Analysis) Patterns’ (SESERA 2004), he establishes, building on his attempt at characterising analysis patterns (SESERA 2000a), that there is still no taxonomy of analysis patterns. For design patterns there is the ‘Bible’ of the Gang of Four, that established a systematic approach. For analysis patterns there is, according to his view, nothing comparable: each author uses his or her own level of abstraction for their analysis patterns. He criticises Fowler’s patterns, among others, as ‘pearls of abstraction process that are difficult to combine even among themselves’ (SESERA 2004 p. 37). Therefore, he arranges analysis patterns into three hierarchical levels (Fig. 2.11).



Source: SESERA 2004

Fig. 2.11. Hierarchical levels of analysis patterns

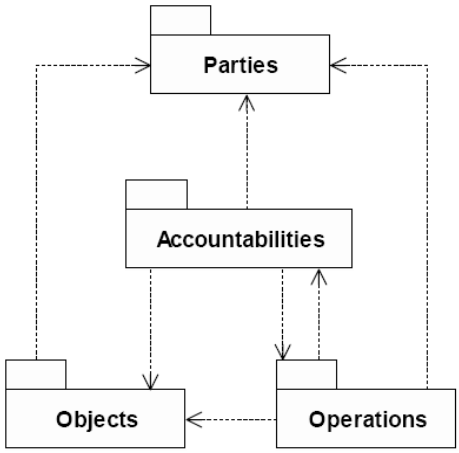
Here, the more general levels form a framework for the more specialised levels lying below them (Fig. 2.12).



Source: SESERA 2004

Fig. 2.12. Example for Sesera's pattern hierarchy

Between the pattern packages and the analysis patterns of each level there are horizontal relationships which are import relationships between different pattern packages. Fig. 2.13 depicts the relationships of the pattern packages on the first level.



Source: SESERA 2005a

Fig. 2.13. Relationships of the pattern packages on the first level (General Analysis Patterns)

With 'A Recurring Fulfilments Analysis Pattern' (SESERA 2000b) Sesera introduces a pattern for the modelling of 'recurring (contract) fulfilments.' In 'Obligation-Fulfillment: A Pattern Language

for Some Financial Information Systems' (SESERA 2005), Sesera presents a pattern language, consisting of eight patterns, for financial information systems. He categorizes these patterns as belonging to the second level of the specialised analysis patterns of his hierarchy. Each pattern is introduced in a structured template; diagrams are modelled in UML. For reasons of space, Sesera foregoes the depiction of the class diagrams for four of the eight patterns introduced, and for all eight patterns he omits the depiction of the dynamic part of the patterns.

2.17 Petia Wohed

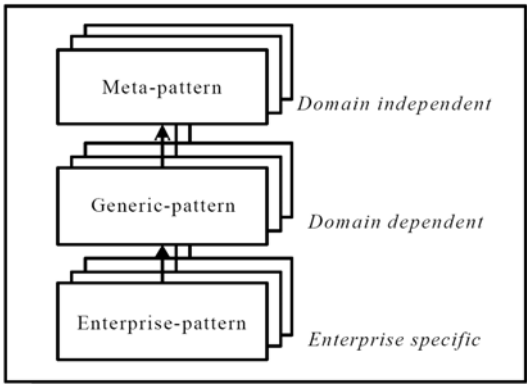
In her publications (WOHED 2000a, WOHED 2000c), Petia Wohed pursues the approach of automated pattern-finding and pattern-adaptation for the generation of an initial analysis model. The pattern-finding and pattern-adaptation is supported by a tool, the so-called 'Modelling Wizard Tool', and is based on a pattern library. The user is led through a dialogue, and from his answers to the questions an initial model is generated step-by-step. The tool prototype is restricted to the booking domain; the patterns used originate from Hay and Fowler.

With the help of Coad's transaction patterns, Wohed (WOHED 2000b) attempts to abstract the implementation of the booking domain in the Modelling Wizard Tool onto the transaction domain. This is done to demonstrate that the tool, when adapted to the transaction domain, can be deployed in every domain. This, however, goes hand in hand with a loss of information; therefore, she suggests that for deployment in a particular domain the tool must be specialised once more beforehand away from the transaction domain. For this reason the Modelling Wizard Tool cannot be used in a new domain without time and effort as it has to be specialized from the abstract transaction domain into the concrete new domain (e.g. booking) before it can be used effectivly.

2.18 Maria João Ferreira, Pericles Loucopoulos

With their publication 'Organisation of analysis patterns for effective re-use' (FERREIRA and LOUCOPOULOS 2001), Maria João Ferreira and Pericles Loucopoulos made their target to support the automatic pattern selection. To this end, they propose a classification scheme for analysis patterns

Ferreira and Loucopoulus assume that the re-use of a pattern is considerably influenced by its retrievability; therefore, the patterns must be provided with attributes for automatic evaluation. For this purpose, they initially arrange the analysis patterns into three levels of abstraction, as shown in Fig. 2.14.



Source: FERREIRA 2001

Fig. 2.14. Pattern abstraction levels

In the second step, they generate a classification scheme that is based on facets as characterisation features. Each facet is represented by a set of terms, the so-called term space. Table 2.1 shows the five facets identified by Ferreira and Loucopoulus Table 2.2 depicts a section of the defined term space.

Table 2.1. Pattern Classification Scheme

Concept	**Facet Concept**
Context (the environment in which analysis pattern operates)	Domain (noun) – F1 DomainArea (noun) – F2
Concept (abstraction captured in an analysis pattern)	Process (verb) – F3 ProcessOn (noun) – F4
Content (the implementation of the abstraction)	Type (noun) – F5

Source: FERREIRA 2001

Table 2.2. Example for a term space

Facet					
Domain (noun)	**DomainArea** (noun)	**Process** (verb)	**ProcessOn** (noun)	**Type** (noun)	**Term Space**
independent	human_resource	define	Competence	text	
electricity	distribution	plan	Competency	rule	
…	…	measure	Business	dfd	
		estimate	Planning	concept	
		…	Activity	…	
			…		

Source: FERREIRA 2001

During the manual classification of a pattern, each facet is assigned at least one term from the defined batch of terms. By means of the terms assigned to the facets the pattern that has come into question can then be selected from a pattern catalogue on an automated search and presented to the enquirer. In order to make the selected patterns comparable, Ferreira and Loucopoulus suggest a structured template for categorising and saving the patterns in the pattern catalogue.

In continuing their work (FERREIRA and LOUCOPOULOS 2001), Ferreira and Loucopoulus present in their publication ‘Business analysis patterns: methodological and support environment aspects’ (FERREIRA and LOUCOPOULOS 2003) the prototype of the tool ‘Re-use Infrastructure Based on Analysis Patterns’ (RIBAP). With the development of RIBAP Ferreira and Loucopoulus rigorously implement their ideas for automatic pattern-selection.

The basis for filing a pattern in RIBAP is the pattern template from FERREIRA and LOUCOPOULOS 2001. The template is to ensure that each pattern is described in such a manner that allows it to be re-used effectively. In order to describe the relationships between the patterns, Ferreira and Loucopoulus developed a classification scheme (Table 2.3). Since a pattern usually only offers a solution for a part of a problem, it is important to know its relationships to other patterns, e.g. ones that it could be combined with.

Table 2.3. Classification scheme of pattern relationships

Relationships	Meaning
Primary	
X Specialises Y	Pattern X is a sub pattern of Y
X Uses Y	Pattern X includes a sub pattern Y. It solves a sub problem of a larger problem
X (is) Similar Y	Pattern X and pattern Y are alternate solutions to the same problem and may be used alternatively (OR)
X (can be) combined (with) Y	Pattern X and pattern Y are alternate solutions to the same problem and may be used simultaneously (AND)
X Alternates Y	Pattern X and pattern Y are alternate solutions to the same problem and may be used either in isolation or simultaneously (AND/OR)
Secondary	
X Related Y	Pattern X is associated with pattern Y
X RelatedOPT Y	A pattern has a related pattern of another type: • Change process pattern X includes related product pattern Y • Product pattern X includes related change process pattern Y

Source: FERREIRA and LOUCOPOULOS 2003

The patterns in RIBAP are classified with the help of the pattern-classification scheme from FERREIRA and LOUCOPOULOS (2001). Thus a developer can find a suitable analysis pattern for his problem with a search enquiry via the terms provided in the classification scheme.

In addition, the patterns in RIBAP can be organised in pattern catalogues and pattern languages.

2.19 Michael Hahsler, Andreas Geyer-Schulz

In his dissertation ‘Analyse Patterns im Softwareentwicklungsprozeß’ (Analysis patterns in the software development process, HAHSLER 2001), Michael Hahsler examines several aspects of analysis patterns not previously dealt with in literature. A result of his research of literature (e.g. Coad, Fowler) is that for analysis patterns, in contrast to design patterns, there still exists no adequately described template for documentation (FERREIRA and LOUCOPOULOS (2001) published their template in the same year as Hahsler). Hahsler introduces a detailed template that is based on existing templates for design patterns (see chapter 3.1.2). The most important departure from previous templates for design patterns is found in the section ‘Consequences’. With analysis patterns, according to Hahsler, this aspect is used mainly for the description of economic, organisational and social aspects.

According to Hahsler, the application scope of analysis patterns is wider than previously described, as analysis patterns can support software development from the recognition of the real problem, via the actual analysis, up to the transfer into the design model. Through the information encapsulated in them, analysis patterns support the abstraction of the real problem into the analysis model and the later transformation of the finished analysis model into the design model.

Using the suggested template, Hahsler introduces four patterns from the field of information management.

In the second part of his dissertation, Hahsler introduces the Virtual University case study, in which the above-named patterns were put into practice. In doing so, he demonstrates the potential for cutting costs through the application of the analysis patterns.

Together with Andreas Geyer-Schulz, Hahsler presents the most important results of his dissertation in the English article ‘Software Engineering with Analysis Patterns’ (GEYER-SCHULZ and HAHSLER 2001).

2.20 Chris Rupp, Jürgen Dallner

In their article ‘Mustergültige Anforderungen’ (‘Standard Requirements’, RUPP and DALLNER 2001) Chris Rupp and Jürgen Dallner introduce their method for the description of requirements with the help of a requirement template. Here, the requirement template defines the syntactical structure of a textual requirement.

The requirement template plays a central role in Rupp and Dallner’s definition of requirement patterns: ‘A requirement pattern describes application-specific problems by making available a set of template-based descriptions of requirements in natural language, comprising semantically defined elements along with related acceptance criteria and model elements.’ Thus, according to Rupp and Dallner’s definition, an analysis patterns should also contain textual requirements. In this, they differ from other authors’ understanding of an analysis pattern.

2.21 Martin Auer

In his work ‘A Software Metric Pattern Dialect’ (AUER 2002), Martin Auer introduces a pattern language for the domain of software metrics. As the origin for generating the pattern language, he uses the patterns from Fowler’s ‘measurement and observation.’ He translates Fowler’s patterns from their respective domain into the software-metric domain and adapts them if necessary. This procedure, in comparison to the creation of patterns from scratch, has the advantage that the pattern language is more complete from the beginning.

To round off, Auer introduces three new domain-specific patterns. He describes the new patterns textually, without templates, supplemented by a UML model.

2.22 Haitham S. Hamza, Mohamed E. Fayad

In his Master’s thesis ‘A Foundation for Building Stable Analysis Patterns’ (HAMZA 2002a), Haitham Hamza presents the new concept of “stable analysis patterns”. He describes the structure and the characteristics of a stable analysis pattern, defines a template for analysis patterns that is supposed to be better suited for description than the ones used previously, and introduces a pattern language that serves as a manual for generating stable analysis patterns.

Hamza uses two types of pattern-classification. On the one hand, he subdivides the patterns into domain-specific and cross-domain or domainless, and on the other hand he arranges them into three categories according to how they have been created. He distinguishes between patterns that (1.) originate through experience or (2.) application of analogies and (3.) others that were generated in accordance with the concept of stable analysis patterns.

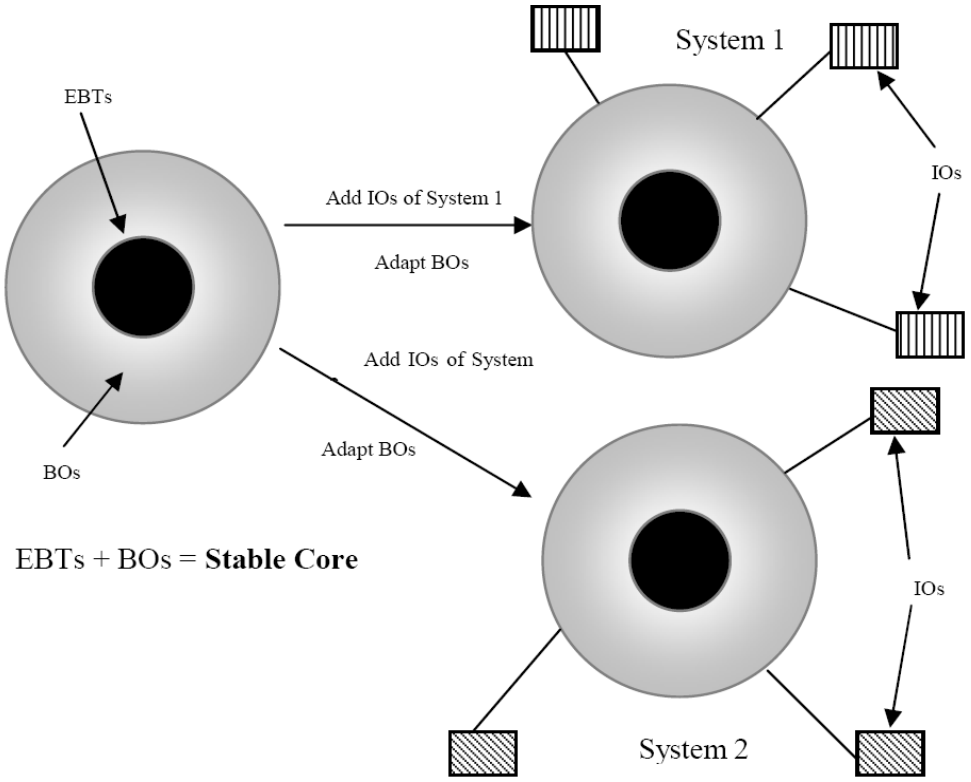
Furthermore, he identifies three main factors that influence the re-usability of analysis patterns: the stability, the degree of abstraction and the documentation of a pattern.

Stability means that the pattern merely models the core of a problem. Since the core of a problem no longer depends on the context of the original concrete problem, the pattern can be re-used independently of the context. According to Hamza, one of the main causes of the instability of patterns is the intermingling of analysis elements and design elements.

A pattern’s level of abstraction determines how widely it can be used; there must be a balance between flexibility and applicability on the one hand (a high abstraction level makes the pattern useable across domains) and the comprehensibility and complexity of the re-use on the other (a low abstraction level makes the pattern more comprehensible and reduces the complexity of the application). A level of abstraction that is too high can result in models that are too abstract and

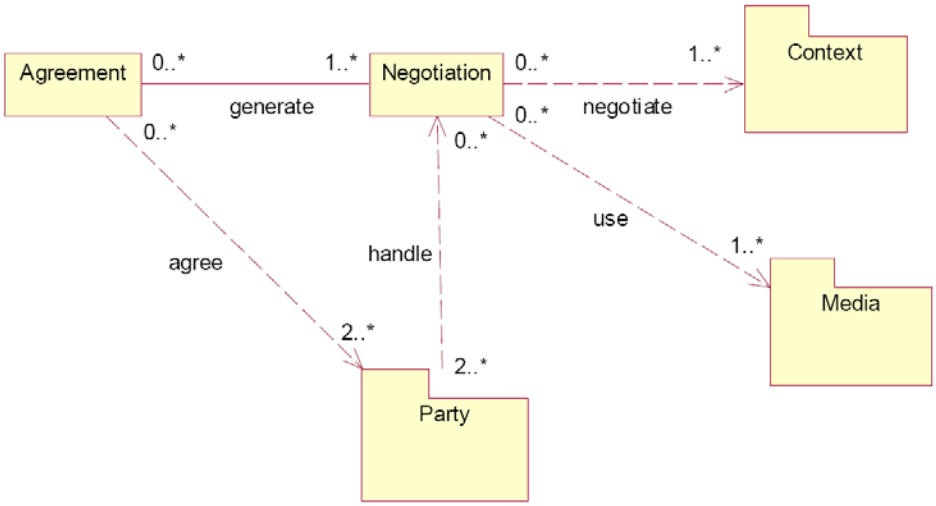
therefore too difficult to understand and re-use. A low level of abstraction can result in domain-specific patterns for problems that are actually cross-domain.

According to Hamza, the application of analysis patterns is for the above reasons not as simple as it should be. Thus in order to obtain effective and re-useable analysis patterns, Hamza developed the concept of the stable analysis pattern, which is based on the concept of software stability (FAYAD 2001). Fig. 2.15 shows the model of the concept. This combines patterns from three components, 'Enduring Business Themes' (EBT) and 'Business Objects' (BO), which remain stable when the application or the context changes. The third components are the 'Industrial Objects' (IO), which are always adapted to the context in which the pattern is applied. A crucial feature of a stable analysis pattern is the fact that it strictly addresses only one core problem. Fig. 2.16 shows the AnyNegotiation pattern and Fig. 2.17 shows an application example.



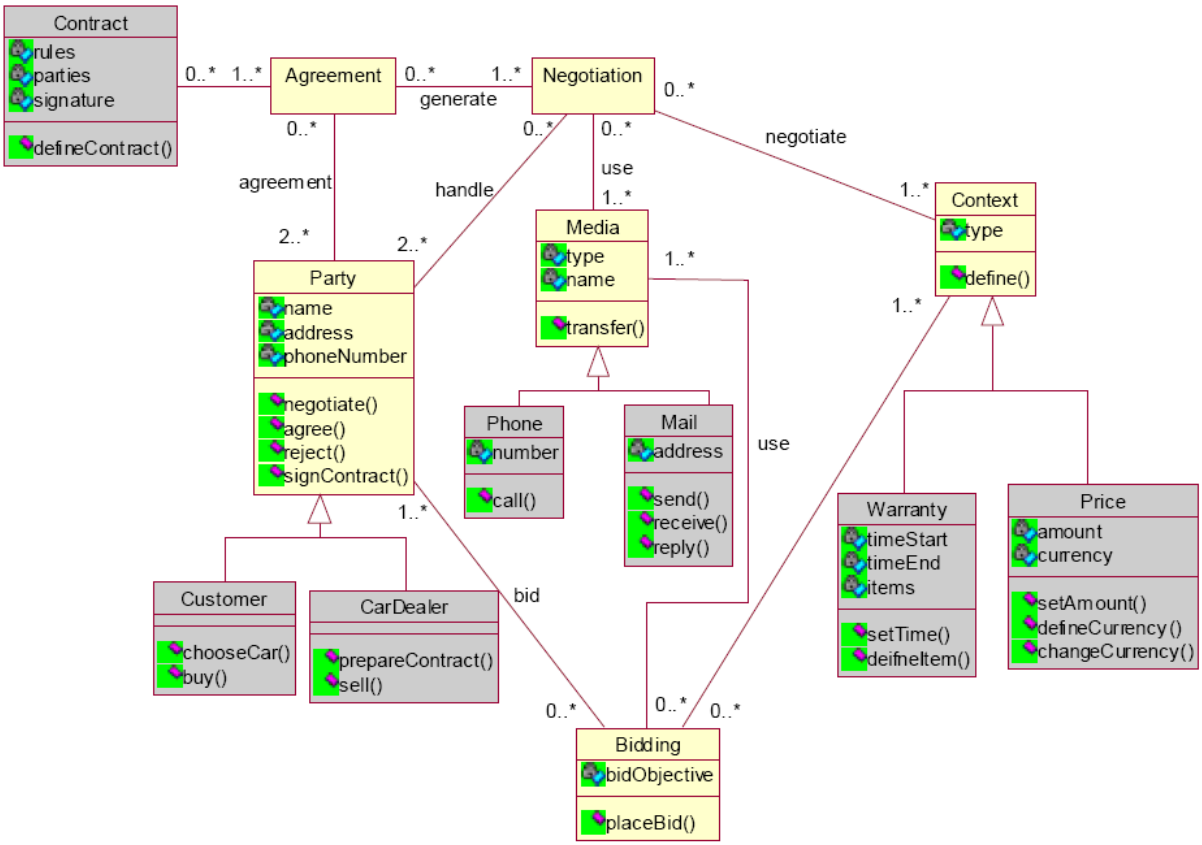
Source: HAMZA 2004b

Fig. 2.15. Software stability-concept model



Source: HAMZA 2002a

Fig. 2.16. AnyNegotiation pattern



Source: HAMZA 2002a

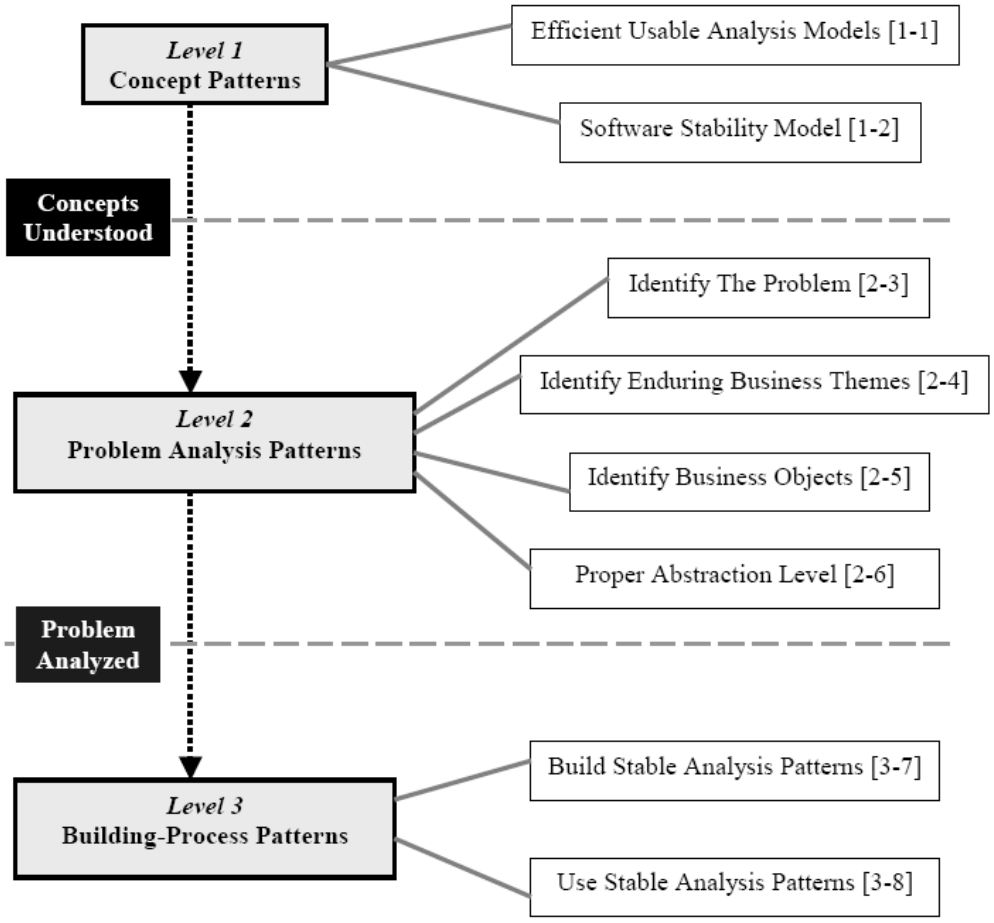
Fig. 2.17. Application of the AnyNegotiation pattern on the model of a car purchase

Hamza defines eight characteristics for the appraisal of analysis patterns. They should be ‘simple, complete and most probably accurate, testable, stable, graphical or visual, easy to understand, general, easy to use and re-use.’ If a pattern is missing one of these properties, its effectiveness and re-usability is limited.

Hamza examines the three categories, classified by him according to the procedure for pattern-generation, and comes to the conclusion that only the stable analysis patterns can demonstrate all the eight aforementioned characteristics.

In a set of instructions he calls a pattern language, Hamza describes the creation of stable analysis patterns. Each step for the generation of a stable analysis pattern is documented in the form of one of eight patterns which help with the generation of a new pattern.

The concept of the stable analysis pattern and the pattern language for the generation of stable analysis patterns (Fig. 2.18) are published in HAMZA and FAYAD (2002b) and HAMZA and FAYAD (2002c) respectively.



Source: HAMZA 2002c

Fig. 2.18. Pattern language for the creation of stable analysis patterns: overview

In 'Towards A Framework For Analysis Patterns Evaluation' (HAMZA et al. 2003a), the authors deal with the problem of pattern selection. This problem arises when the developer has a choice of several patterns for his analysis problem, in which case he has to be able to select the correct pattern. The benefits of applying patterns are reduced if the developer has thoroughly to analyse the problem beforehand in order to be able to select the appropriate pattern. To help the developer with his decision, Hamza et al. propose a framework for the evaluation of analysis patterns.

In 'Extracting Domain-Specific and Domain-Neutral Patterns Using Software Stability Concepts' (HAMZA et al. 2003b) the authors deal with the question of extracting domain-specific and domain-independent patterns from an existing system that was designed according to the concept of software stability.

In the article 'Applying Analysis Patterns through Analogy: Problems and Solutions' (HAMZA and FAYAD 2004a) the authors Hamza and Fayad demonstrate the problems of the application of analysis patterns through analogy and show how they can be solved through the application of stable analysis patterns.

To do this, Hamza and Fayad concentrate on two quality aspects of analysis patterns, traceability and generality. According to Hamza and Fayad patterns that are applied through analogy (e.g. the patterns of Fernandez et al.) possess a sufficient degree of abstraction. However, for solving a problem, these patterns are used like a template i.e. as soon as they are instantiated the pattern disappears in the analysis model (for example, the abstract class-names are replaced by names that are related to the problem.) That means the pattern cannot be re-traced in the model, which has a negative effect on the maintenance of the system.

According to Hamza and Fayad stable analysis patterns were developed to cater to both aspects. They remain always recognisable in the analysis model, as the EBTs and the BOs do not change (see Fig. 2.16 and 2.17), regardless of the context in which the pattern is used.

In 'On the integration of stable analysis patterns with traditional patterns' (HAMZA 2004b) the author describes the problem of different structures of existing analysis patterns. Analysis patterns differ in their structure depending on the approach and methods with which they were developed. According to Hamza, the lack of consistency in the structure of the available analysis patterns causes problems when using patterns in an analysis model. He introduces a multiple-phase process for the integration of a stable analysis pattern and a pattern of a different structure.

In two further papers, the authors treat software analysis and design in an overall fashion. In 'PAD: A Pattern-Driven Analysis and Design Method' (HAMZA and CHEN 2005) Haitham Hamza and Yi Chen introduce their method of the same name for software development (see 4.1.3.9). With 'A Framework for Developing Design Models with Analysis and Design Patterns' (CHEN et al. 2005) Yi Chen, Haitham Hamza and Mohamed Fayad establish in detail the path from the analysis model (consisting of stable analysis patterns) to the design model.

In addition to the above-mentioned works, Haitham Hamza and Mohamed Fayad published or were involved in the publication of eleven stable analysis patterns. All patterns are described in a structured template; the graphical depiction was carried out in UML: 'The Sampling Analysis Pattern' (SÁNCHEZ et al. 2003), 'The AnyAccount Pattern' (FAYAD and HAMZA 2003a), 'The Automation Analysis Pattern' (FAYAD et al. 2003b), 'The Accessibility Analysis Pattern' (FAYAD et al. 2003c), 'Evaluation Analysis Pattern' (FAYAD and PRADEEP 2003d), 'The Searching Analysis Pattern' (FAYAD et al. 2004a), 'The Trust Analysis Pattern' (FAYAD et al. 2004b), 'The Negotiation Analysis Pattern' (HAMZA and FAYAD 2004), 'The Learning Stable Analysis Pattern' (FAYAD and TELU 2005), 'The Visualization Stable Analysis Pattern' (FAYAD and DAS 2007a), 'The Classification Stable Analysis Pattern' (FAYAD and DAS 2007b).

2.23 Andreas Harrer, Vladan Devedzic

In 'Design and Analysis Patterns in ITS Architectures' (HARRER and DEVEDZIC 2002) and 'Software Patterns in ITS Architectures' (DEVEDZIC and HARRER 2005) the authors deal with the use of patterns in the domain of intelligent teaching systems.

They do not focus on the application of patterns but on the recognition and extraction of patterns from existing systems; they study analysis patterns as well as design patterns with the purpose of making a collection of patterns available in the domain of intelligent teaching systems (ITS). In DEVEDZIC and HARRER (2005) they present three analysis patterns and two design patterns in a structured template.

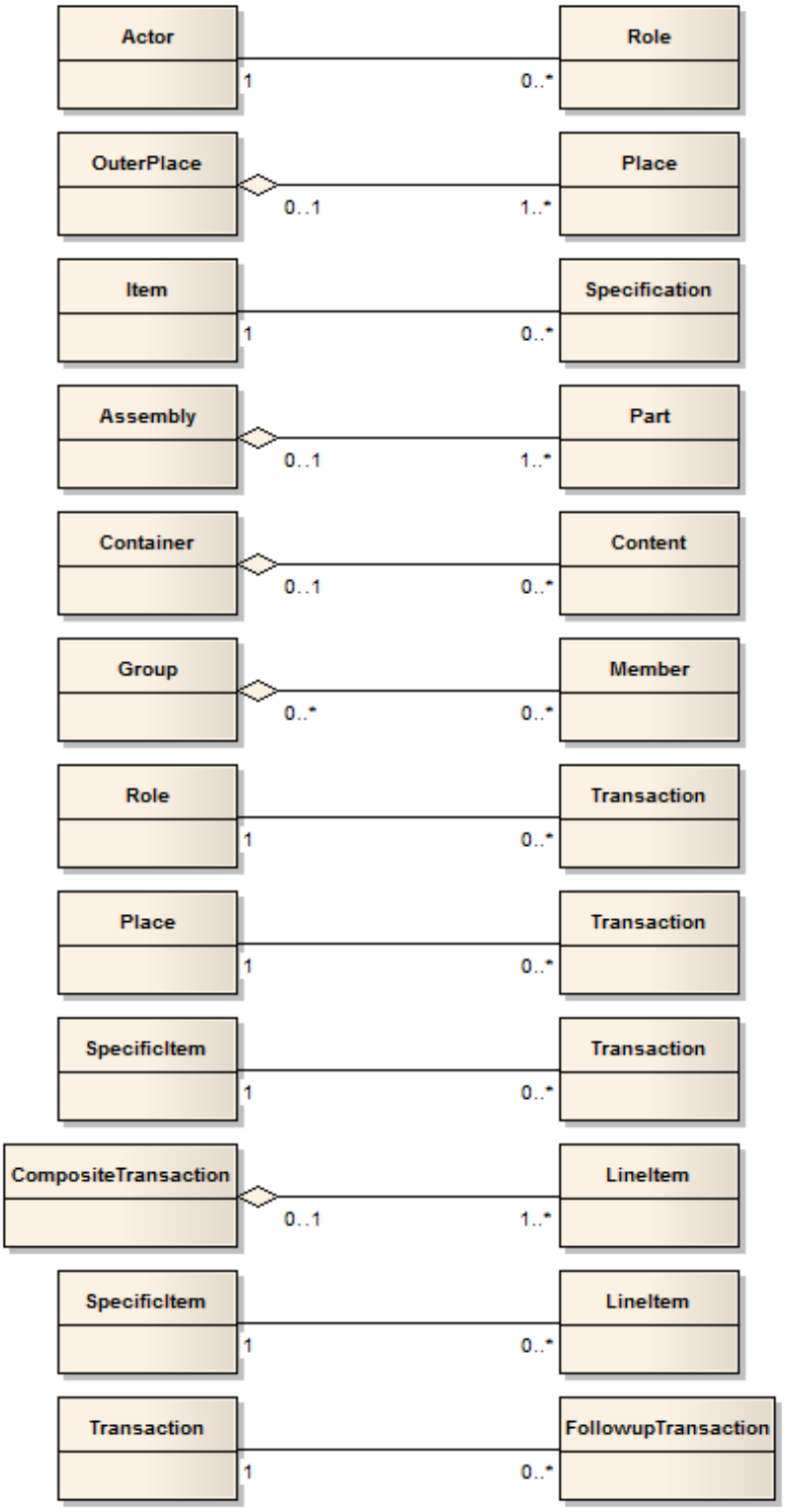
2.24 Francis G. Mossé

In his article ‘Modeling Roles – A Practical Series of Analysis Patterns’ (MOSSÉ 2002) Francis G. Mossé describes analysis patterns for the modelling of roles in object-oriented analysis. For example, a company can take on the role of both a customer and a vendor. The customer buys products, the vendor sells them. Mossé documents the patterns without a structured template; the class diagrams of the patterns are modelled in UML.

2.25 Jill Nicola, Mark Mayfield, Mike Abney

In their book ‘Streamlined Object Modelling – Patterns, Rules and Implementation’ (NICOLA et al. 2002) Jill Nicola, Mark Mayfield and Mike Abney present ‘Streamlined Object Modelling,’ an object-modelling method for analysis, the core of which consists of twelve collaboration patterns, five types of business rules and three types of business services.

The collaboration patterns of Nicola et al. are the equivalent of analysis patterns. They are a further development of the patterns of COAD et al. (1995). The twelve collaboration patterns (see Fig. 2.19) cannot be reduced any further; that means they cannot be broken down into smaller patterns. Each pattern consists of a collaboration of two so-called ‘pattern players’ (objects of the following types: people, places, things, events); thus with these twelve patterns every conceivable relationship between the pattern players in the real world can be modelled. According to the authors every domain can be broken down into collaborations between ‘people’ objects, ‘place’ objects, ‘thing’ objects and ‘event’ objects, and therefore can be modelled with the help of the twelve collaboration patterns.



Source: based on NICOLA et al. 2002, p. 30

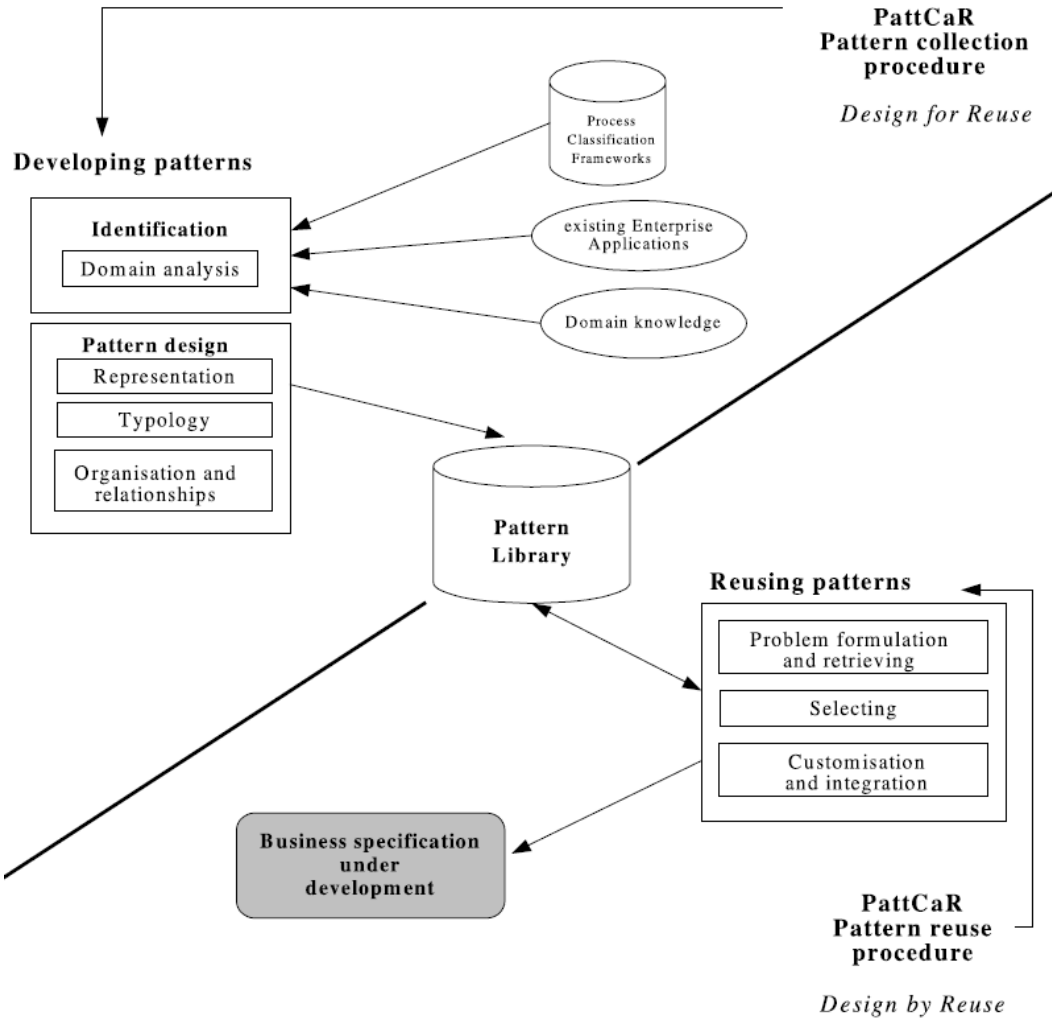
Fig.2.19. The 12 collaboration patterns

The collaboration patterns are of low complexity, as they consist always of only two objects; therefore, several patterns have to be combined in order to generate an analysis model. Their low complexity and high level of abstraction make them domain-independent.

Although Nicola et al. provide a detailed description of each pattern, they do not use an explicit template for documentation.

2.26 Isabel Secura, Pericles Loucopoulos

In their article 'Towards a systematic approach to the capture of patterns within a business domain' (SECURA and LOUCOPOULOS 2002) the authors present the PattCaR (Pattern Capture and Re-use) method. PattCaR is a systematic approach for the capture and subsequent re-use of patterns in existing applications of a domain. Fig 2.20 gives an overview of PattCaR



Source: SECURA and LOUCOPOULOS 2002

Fig.2.20. PattCaR - overview

Secura and Loucopoulos present two analysis patterns developed using the PattCaR pattern collection procedure, documented in a structured template.

2.27 Lei Zhen, Guangzhen Shao

In 'Analysis patterns for oil refineries' (ZHEN and SHAO 2002) Lei Zhen and Guangzhen Shao present four analysis patterns for modelling information systems for oil refineries. In 'Analysis Patterns for Materials and Products of Refineries' (ZHEN 2003) Zhen introduces another four patterns for the description of materials and products of a refinery. The patterns were built along the lines of the semantic analysis pattern (FERNANDEZ and YUAN 2000b). They are documented in a structured template; the graphical model uses UML.

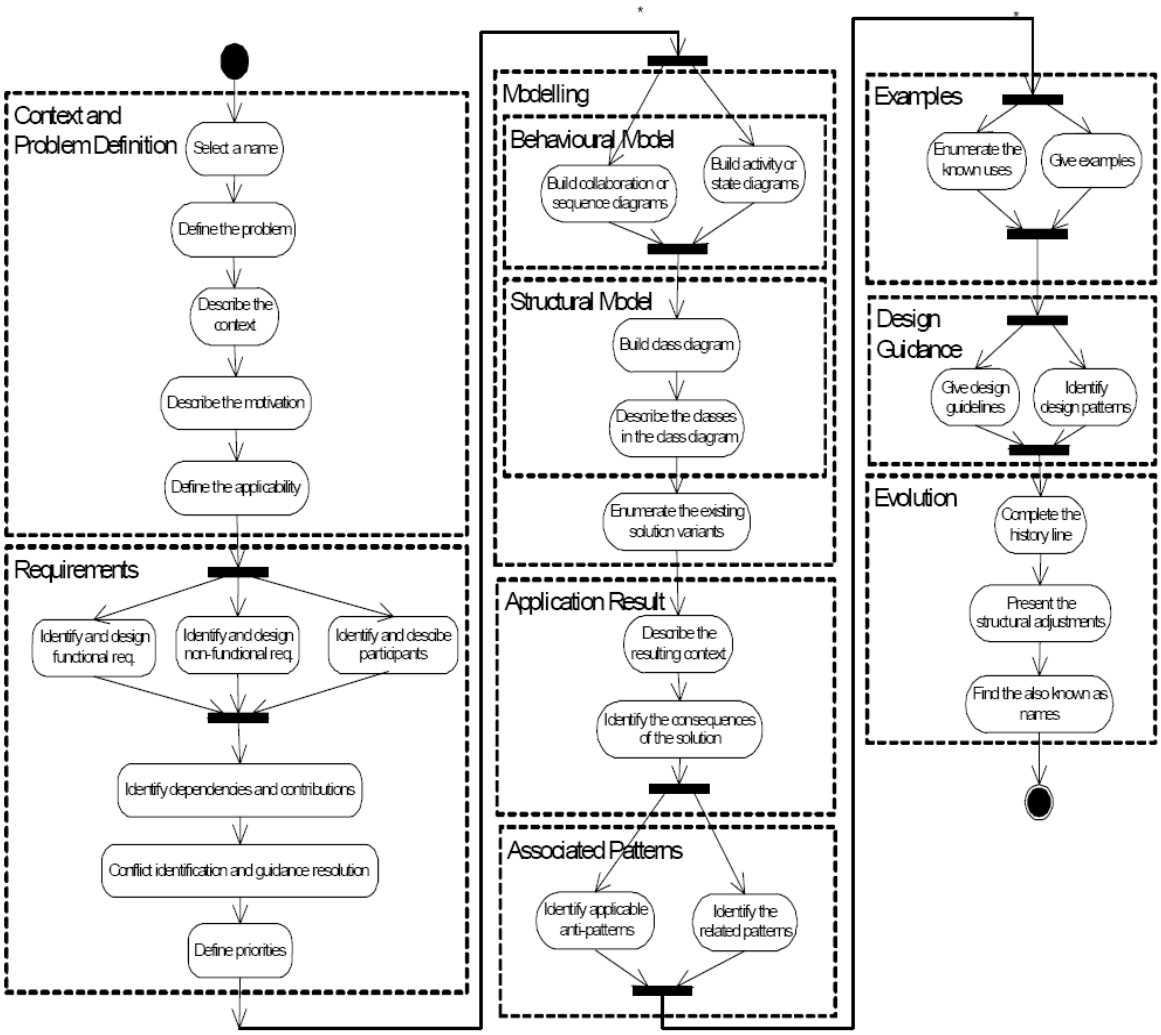
2.28 Marta Pantoquilho, Ricardo Raminhos, João Araujo

In their publication 'Analysis Patterns Specifications: Filling the Gaps' (PANTOQUILHO et al. 2003) Pantoquilho et al. introduce their template for the documentation of analysis patterns.

For Pantoquilho et al. the previous approaches to the documentation of analysis patterns are not comprehensive enough to enable the developers to apply them in practice. A more detailed description would help the developers to deploy the correct patterns in a successful and efficient manner. To solve this problem they suggest a template for analysis patterns which contains the elements of the approaches up till now, as well as new elements for the completion of the pattern description (e.g. the development that a pattern has already undergone; structured description of the solution in detail).

Pantoquilho et al. demonstrate their template by applying it to Fowler's analysis pattern *Party*.

In 'A Systematic Analysis Pattern Specification' (PANTOQUILHO et al. 2006) the authors once more present their template along with a process model for the systematic creation and documentation of analysis patterns (Fig. 2.21).



Source: PANTOQUILHO et al. 2006

Fig. 2.21. Process model for the creation of an analysis pattern

2.29 Sandeep Purao, Veda C. Storey, Taedong Han

In their article 'Improving Analysis Pattern Re-use in Conceptual Design: Augmenting Automated Processes with Supervised Learning' PURAO et al. (2003) present their method for tool-supported application of analysis patterns, using calibratable learning-algorithms in a semi-automatic process in order to emulate the approach of an experienced developer in the application of analysis patterns.

What they call the naive approach to tool support of the analysis pattern application is limited to the generic re-use tasks of retrieval, adaptation and integration. With the help of intelligent trainable algorithms their improved approach tries to emulate a human developer, making analogies and putting together the whole analysis model from individual components. Guided by the intelligent tool-assistant an initial analysis model can be generated in a quick and automated process, and is then adapted and refined by the developer.

Purao et al. have developed and tested a tool prototype, with the patterns from COAD et al. (1995) serving exclusively as the pattern-repository, as they are domain-independent. Domain-independence is a prerequisite for the use of the patterns, as the tool is intended to be useable for all domains without further adaptation. Empirical tests with the prototype have shown that the improved approach of Purao et al. produces far better analysis models than the naive approach. The results were deemed to be very good – even in domains for which the learning algorithms were not trained. The analysis models that were generated with the improved approach came closer to the reference models created by Purao et al. than analysis models that were generated with the naive approach.

2.30 Jim Arlow, Ila Neustadt

In their book 'Enterprise Patterns and MDA' (ARLOW and NEUSTADT 2004) Jim Arlow and Ila Neustadt present the concept of 'Business Archetypes' and 'Business Archetype Patterns'. (MDA stands for model driven architecture.) The concept of the archetype, on which business archetypes and business archetype patterns are built, is defined by Arlow and Neustadt as 'a primordial thing or circumstance that recurs consistently and is thought to be a universal concept or situation' (ARLOW and NEUSTADT, p. 4). For the relevant field of business and software development they define the business archetype as 'a primordial thing that occurs consistently and universally in business domains and business software systems' (ARLOW and NEUSTADT 2004 p. 5). This results in the definition for the business archetype patterns as 'a collaboration between business archetypes that occurs consistently and universally in business environment and software systems' (ARLOW and NEUSTADT 2004 p. 6).

The concept of the archetype pattern differs from conventional analysis patterns. They have a higher level of abstraction than analysis patterns, since even a business archetype has a higher level of abstraction than a class in analyis. Business archetypes still explicitly support their variation (e.g. omission of optional components) in order to allow themselves to be adapted to specific domains. They are generated and optimised with regard to their use in MDA.

Arlow and Neustadt introduce a catalogue of nine archetype patterns and several variations of each of those nine patterns. They describe these as essential patterns for the business domain. Each pattern contains a solution for modelling a special part of a business system. The patterns are documented using UML notation and the technology of literate modelling, which is presented here the first time by Arlow and Neustadt. Literate modelling is used to render UML models (here, archetype patterns) comprehensible and accessible for users outside the field of software development.

Arlow and Neustadt describe in brief how archetype patterns can be identified and captured in other domains. They also go into the application of archetype patterns (see 4.1.3.7).

2.31 Narasimha Bolloju

In his article 'Improving the quality of business object models using collaboration patterns' (BOLLOJU 2004) Narasimha Bolloju presents the results of his study of the application of analysis patterns, using exclusively the twelve collaboration patterns as defined by NICOLA et al. (2002) (see 2.25).

For the study thirteen teams of students at first solved an analysis modelling task without patterns. Afterwards they had to improve their models by using the twelve collaboration patterns. The outcome of the study shows that with the help of the patterns the models could almost always be improved. In particular, the improved models contained fewer errors and were more complete.

Furthermore, Bolloju applies the results of the study to draw conclusions regarding the use of analysis patterns right at the beginning of the modelling. He defines four guiding principles for the effective use of collaboration patterns at the beginning of analysis.

2.32 Jordi Cabot, Ruth Raventós

In 'Roles as Entity Types: A Conceptual Modelling Pattern' (CABOT and RAVENTÓS 2004) Jordi Cabot and Ruth Raventós introduce an analysis pattern for modelling role concepts. According to Cabot and Raventós, roles can be used to describe dynamic and temporary aspects of real-world objects (for example, someone takes on the role of a student for a certain time; after graduation, the student role finishes, and the person assumes the role of an employee). Their pattern contains the static as well as the dynamic aspects of roles.

For documentation Cabot and Raventós use the template suggested by HAHSLER (2001) and UML notation for graphical depiction.

2.33 Jugurta Lisboa Filho, Victor de Freitas Sodré, Jaudete Daltio, Maurício Fidelis Rodrigues Júnior, Valério Vilela, Marcus Vinícius Alvim Andrade

In 'A CASE Tool for Geographic Database Design Supporting Analysis Patterns' (LISBOA et al. 2004) and 'Improving productivity and quality of GIS databases design using an analysis pattern catalog' (DE FREITAS SODRÉ et al. 2005) Jugurta Lisboa Filho and Victor de Freitas Sodré describe the CASE tool ArgoCASEGEO developed by them. ArgoCASEGEO was developed as an open source CASE tool for the domain of Geographical Information Systems (GIS). It fully supports the UML-GeoFrame introduced by LISBOA et al. (1999).

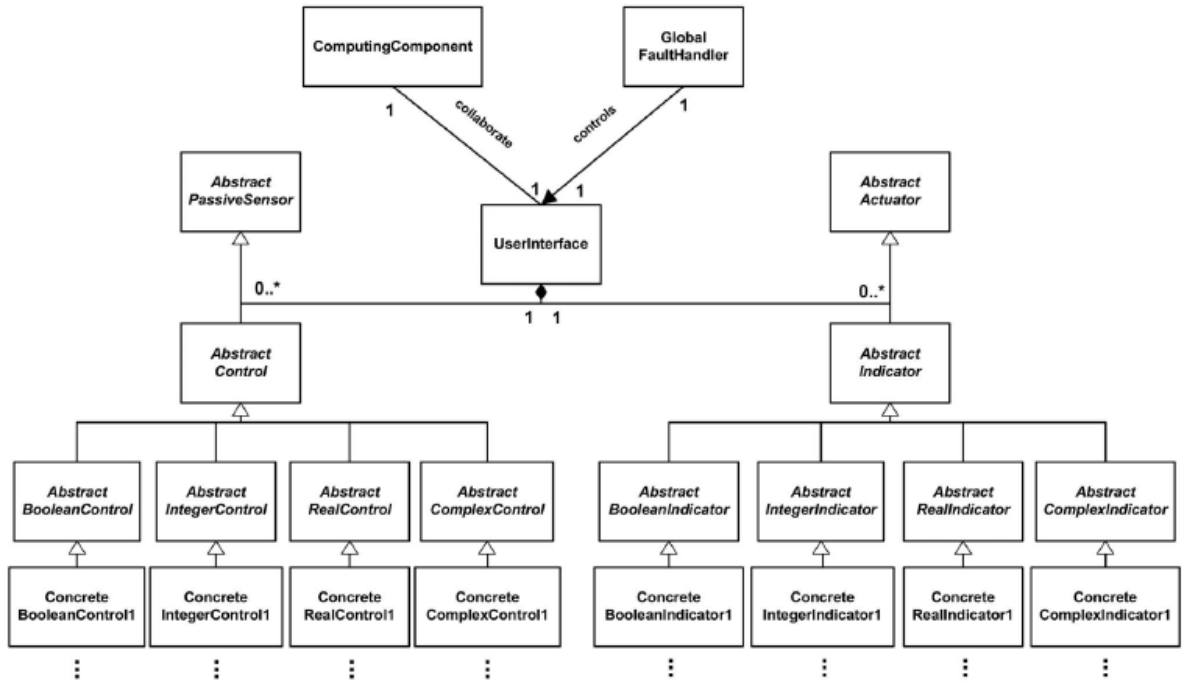
Since according to Lisboa et al. the developers of GI systems are usually inexperienced, the integrated catalogue of analysis systems is the key element of ArgoCASEGEO. It enables the developer to attain an analysis model more quickly, more efficiently and with fewer errors. For this purpose the pattern catalogue is filed as a module of the CASE tool in a database. For each pattern the UML model is stored in XMI format and the documentation in XML format. In order to find a suitable analysis pattern for the problem to be solved, the user can search the database by theme, or search through the pattern documentation using an arbitrary key phrase. The storage in XML/XMI format allows easy exchange of patterns.

2.34 Sascha Konrad, Betty H.C. Cheng, Laura A. Campbell

In ‘Object Analysis Patterns for Embedded Systems’ (KONRAD et al. 2004) Konrad et al. present the results of their study on the application of analysis patterns in the development of embedded systems. The study provides a short description of seven patterns they have so far identified, and their application in the analysis phase of software development for embedded systems, as well as a presentation of the authors’ template for pattern documentation.

Konrad et al. established that in the field of embedded systems mainly ad hoc development approaches are used. Design and implementation are emphasised over analysis, and conceptual errors have to be corrected at great expense at a later stage of development. Their analysis patterns were mainly identified and documented by means of examples (depicted in UML) from the automobile industry in order to provide developers with a catalogue of key elements for the modelling of embedded systems. Feedback from partners in the industry showed that the analysis patterns improve communication between developers with varying backgrounds and levels of experience by providing a common basis.

The object analysis patterns fall into two classes, the ‘structural’ and ‘behavioural’ object analysis patterns. The classification is carried out according to the sub-phases of the analysis phase, depending on whether a pattern describes more behavioural or more structural aspects. An example pattern is presented in Fig. 2.22.



Source: KONRAD et al. 2004

Fig.2.22. UML class diagram of user interface pattern

The pattern template developed by Konrad et al. is made along the lines of that of GAMMA et al. (1995) with some analysis-specific elements added and certain design-specific elements taken away.

2.35 Jayadev Gyani, P. R. K. Murthi

In ‘A Pattern Language for Online Share Trading’ (GYANI and MURTHI 2005) the authors publish eight analysis patterns for the domain of online share trading. The patterns are compiled into a pattern language and are documented as a structured template, although only textually.

2.36 Diego Calvo de Nó

In his Master’s thesis ‘Muster in der objektorientierten Analyse’ (Patterns in Object-Oriented Analysis, DE NÓ 2005) Diego Calvo de Nó describes processes for the application of analysis patterns, as well as the possible compilation of an analysis-pattern catalogue.

In the first part of his work, de Nó introduces two process models, which he calls integration approach and ‘Baukastenmethode’ (building-set approach) (see 4.1.3.10).

In the second part of his work, de Nó presents the draft and the introduction of a freely accessible web pattern catalogue in the form of a Wiki to create a pattern catalogue that grows with the contributions of its users. De Nó intended to start the catalogue with patterns by, for example, Fowler, Arlow and Neustadt. Although the patterns were to be stored in a structured form, they were not to be modelled on a comprehensive and complete template (e.g. PANTOQUILHO et al. 2003). UML models are provided for all patterns.

2.37 Lotte De Rore, Monique Snoeck

In ‘An analysis pattern: Three-party pattern’ (DE RORE and SNOECK 2005) the authors publish a domain-independent analysis pattern which reproduces three concrete or abstract objects from the real world which are related to each other. For documentation they use a structured template and a UML model.

2.38 Enrique Andreu, Rubén Béjar, Miguel Á. Latre, Sergio Martínez, Pedro R. Muro-Medrano

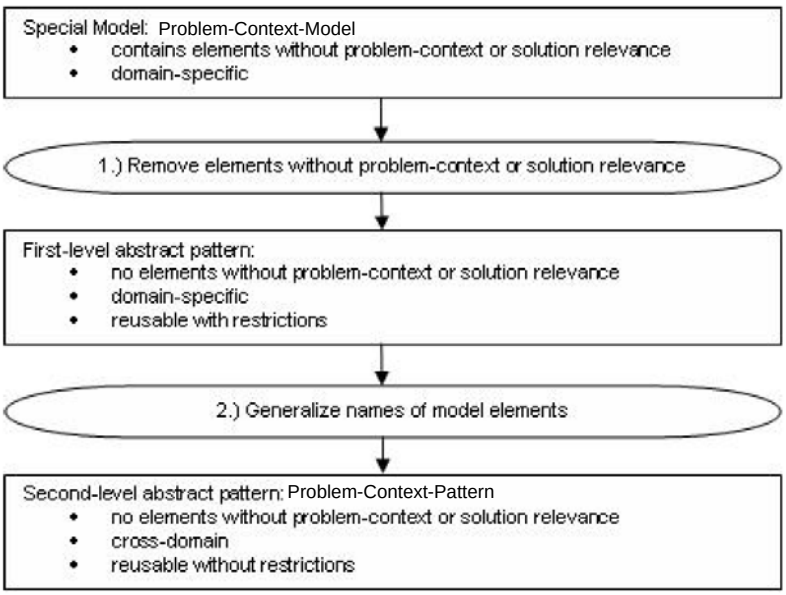
In ‘Pattern-Based Approach to Support Automatic Homogenous Map Labeling with Texts, Charts and Other Elements in a WMS’ (ANDREU et al. 2006) the authors present the generic-label analysis pattern in the domain of geographical information systems. The pattern provides functionality for the automatic labelling of maps in Web Map Services.

For pattern documentation, Andreu et al. use a structured template and UML.

2.39 Alexander Fülleborn, Maritta Heisel

In ‘Methods to Create and Use Cross-Domain Analysis Patterns’ (FÜLLEBORN and HEISEL 2006) Alexander Fülleborn and Maritta Heisel present a method for the derivation of domain-independent analysis patterns from domain-specific ones and their re-use for problems in other domains. The aim is to enable a cross-domain search for a pattern for a concrete domain-specific problem and the instantiation of the abstract pattern retrieved for the concrete problem.

For this purpose, the ‘problem-context-description’ at first needs to be derived from the requirements document. Based on the problem-context-description, the problem-context-model is generated. After this, the problem-context-model, which is still domain-specific, is generalised in two further steps into the problem-context-pattern (Fig. 2.23). During this process elements not relevant to the problem context or the solution are initially removed, and then in the second step the names of the model elements are generalised.



Source: FÜLLEBORN and HEISEL 2006

Fig.2.23. Generalisation process from problem-context-model to problem-context-pattern

With the problem-context pattern as search key a search for suitable problem solutions – in other words, analysis patterns – can now be undertaken across all domains. If a suitable problem-context pattern is found, the relevant solution-pattern can be used for the solution of the problem.

So on the one hand we have the problem-context pattern for the problem to be solved as search key, and on the other hand, a pattern catalogue which maps the problem-context patterns to the corresponding solution patterns.

2.40 Ralph Kemp

In his Master's thesis 'Anwendung von Mustern in der objektorientierten Analyse' (KEMP 2006) Ralph Kemp examines the application of patterns in object-oriented analysis. He presents four process models ('pattern-selection ante', 'pattern-selection intra', pattern-selection ante et intra' and 'pattern-selection post') for the application of analysis patterns (see 4.1.3.11). Afterwards he allocates the implicit and explicit application descriptions of analysis patterns in literature to his process models. Kemp's process-models visualise the application of analysis patterns in the context of all phases of software development, something that only HAMZA and CHEN (2005) have done in such detail.

Furthermore, Kemp puts forward a simple but effective labelling of patterns applied in analysis models, using a UML stereotype (see chapter 4.2).

2.41 Volker Pleß, Giselher Pankratz, Andreas Bortfeldt

In 'The Rate Structure Pattern: An Analysis Pattern for the Flexible Parameterization of Charges, Fees and Prices' (PLEß et al. 2006) Volker Pleß, Giselher Pankratz and Andreas Bortfeldt present a domain-independent analysis pattern for flexible pricing. The pattern is not documented in a structured template; the pattern model is illustrated using UML.

2.42 Luigi Ubezio, Claudia Raibulet, Antonio Carpinato

In their article ‘Novel Analysis Patterns in the Context of the Managed Investments Instruments’ (UBEZIO et al. 2006) Luigi Ubezio, Claudia Raibulet and Antonio Carpinato introduce three analysis patterns from the domain of asset management. The patterns are not documented in a structured template; the graphical models are depicted in UML notation.

2.43 Conclusions from the literature overview

In the recent decade the literature on analysis patterns has undergone an accelerated growth. About 80% of all contributions that are considered for this report were published in 2000 or later. On the one hand a large number of analysis patterns have been created that belong to different specialised business fields or other important domains. Moreover, multiple collections of more abstract and domain-independent patterns were provided. Thus, a rich supply of analysis patterns is available in the mean time in which software analysts can search for inspiration and support in building new analysis models.

Another important thread in the literature deals with methodological aspects related to the creation and application of analysis patterns. Several works have covered the application of analysis patterns in the software development process. The need for better documentation with the help of pattern templates has been recognised; likewise the deficits that result from the lack of pattern organisation have been identified. Some of these methodological questions are discussed in detail in the following sections of this report.

3 Documentation of Analysis Patterns

The documentation of analysis patterns consists of documentation in the narrower sense (form and content of an analysis pattern in a pattern template, for example) and the pattern organisation (management of several patterns).

Good documentation is an important component in every phase of software development, from the analysis to the program code, in order to preserve knowledge and to pass it on.

The same applies to the documentation of analysis patterns in the narrower sense. Its purpose is to preserve all the knowledge regarding the solution of an analysis problem and to pass it on for the re-use of a solution. If a pattern is not adequately documented, the information it contains can not be re-used or can only be re-used restrictedly. The author diminishes the usability of his or her pattern when he or she does not use a template or only uses an incomplete template for documentation.

The second important aspect of the documentation of analysis patterns is their organisation. All methods for the application of analysis patterns discussed in chapter four contain a step ‘select a suitable analysis pattern.’ The finding of a suitable pattern is, then, a basic prerequisite for the application of an analysis pattern. A collection of analysis patterns is required for application, without which an application is not possible – or at least not very effective. If a developer has to first search through the whole of the internet for a possibly suitable existing analysis pattern, he will presumably do his analysis work without pattern-support.

The developer’s main problem, and also that of the acceptance and dissemination of an analysis pattern in everyday use, is not ‘How do I use an analysis pattern?’ but ‘Where do I get a suitable pattern from?’ Until this question is answered – through a central pattern catalogue, for example – analysis patterns cannot develop to their full potential.

In the literature only few works are concerned with the improvement of documentation in the narrower sense through pattern templates (see 3.1.); none of them, however, have become accepted yet. Other work contains thoughts on how analysis patterns can best be organised in catalogues in order to make them accessible to developers efficiently; but it appears that up till now these thoughts still await their implementation.

According to the study by HENNINGER and CORREA (2007), design patterns are, in contrast to general opinion, no better organised than analysis patterns. Their greater number does, on the contrary, make the location of a suitable pattern rather more difficult.

3.1 Pattern templates

The documentation of an analysis pattern is at least just as important as the generation of the pattern in itself. The quality of the documentation contributes considerably to the re-usability of a pattern. For instance, an analysis pattern as a section of a conceptual model is by its nature not easy to understand; the more detailed the description is, however, the easier and quicker the pattern is to comprehend. This means that the better an analysis pattern is documented, the easier it is to be understood and re-used by its potential users.

In the literature two approaches for documenting patterns are used: on the one hand the free description in prose; on the other the description in a structured template.

Perhaps the best-known use of the prose form is the pattern catalogue by FOWLER (1997). This form of pattern description distinguishes itself through the patterns being more difficult to understand and lacking in comparability. It should only be used to supplement documentation in a template.

The description of patterns in structured templates is usually found to vary in characteristics from author to author. Many templates are derived directly from design-pattern templates (e.g. GAMMA et al. 1995) and do not take into account the peculiarities of analysis patterns or only integrate them inadequately. The following sections introduce the most important pattern templates in the literature that were explicitly developed for analysis patterns.

Among these templates, the template from PANTOQUILHO et al. (2003) deserves closer attention. It was especially designed to improve the documentation of analysis patterns. It contains the important 'classic' pattern attributes, as well as new ones that improve its comprehensibility. In particular, the level of detail of the solution description was strongly increased by an itemisation into ten attributes. Through this, the readability and comprehensibility is improved for the user; and the authors of patterns are provided with a detailed guide that shows them how to document their pattern for maximum re-usability. At the same time, the completeness of the documentation of the pattern increases with the level of detail of the pattern template.

To sum up, the use of a pattern template, with regard, in particular, to electronic pattern catalogues and the rising number of published patterns, has turned out to be a must for the successful re-use of analysis patterns.

3.1.1 Peter Coad, David North, Mark Mayfield

COAD et al. (1995) use the pattern template from Table 3.1**.** for all patterns in their pattern handbook. The template is very simple and is missing important attributes, such as a description of the problem to be solved, the context of the application, and the restrictions that the pattern is subject to.

Table 3.1. Pattern template from COAD et al. (1995)

Attributes	**Description**
Pattern Name	Pattern name
Pattern Category	Classification into one of the four pattern catagerories: transaction patterns, aggregate patterns, plan patterns and interaction patterns
Object-Model Template	The pattern depicted as object-model (class diagram)
Typical Object Interactions	Describes the interactions between objects by listing the sequences of possible method requests
Examples	Gives concrete examples for how the pattern classes can be instantiated
Combinations	Names the patterns with which a combination is possible

3.1.2 Michael Hahsler

For his template, HAHSLER (2001) uses primarily attributes that were taken from the design pattern templates from GAMMA et al. (1995) and BUSCHMANN et al. (1996), and thereby meet the standard of the design-pattern documentation. He has newly incorporated the attribute 'Design', which supports the developer on the transition from analysis model to design model.

Table.3.2. Pattern template from HAHSLER (2001)

Attributes	**Description**
Pattern Name	Pattern name, expressing important aspects of the pattern
Also Known As	Further synonyms for this pattern
Intent	Depicts the problem that is to be solved through the pattern
Motivation	An example that shows the effect of a pattern by means of a concrete problem
Forces and Context	Listing and discussion of the forces that are at work in the problem space
Solution	Description of the abstract solution in an analysis model
Consequences	Description of all consequences (positive as well as negative) that result from the pattern application
Design	Shows how the proposed analysis model could be converted into a Design
Known Uses	Examples of systems in which the pattern was discovered or already successfully applied

3.1.3 Marta Pantoquilho, Ricardo Raminhos, João Araujo

PANTOQUILHO et al. (2003) introduce the most comprehensive and most detailed template for analysis patterns. Their template contains all the important aspects of the other templates introduced. It provides the authors of patterns with a detailed structure at decisive points, such as, for example, the illustration of the solution offered by the pattern, and thus helps well to prepare patterns for re-use.

The template already contains attributes for versioning and for tracking the changes of the individual versions. This makes it already suitable for use in a pattern catalogue.

Table 3.3. Pattern template by PANTOQUILHO et al. (2003)

Attributes			**Description**
Name			Names the pattern
Also Known As			Other names by which the pattern is known
Evolution			Catalogue with all previous pattern versions
Structural Adjustments			Description of all extensions and omissions on use of the template
Problem			A short description of the problem that the pattern solves
Motivation			Description of a difficult situation in order to motivate the use of the pattern
Context			Precise description of the context in which the problem recurs
Applicability			Description of the conditions under which the pattern can be used
Requirements	Functional		List of functional requirements in the form of Use-Cases
	Non-functional		List of all non-functional requirements
	Dependencies		Illustration of dependencies between requirements (e.g. graphical)
	Priorities		Definition of the priorities between the requirements (e.g. in a hierarchical structure)
	Conflict Resolution		Explanation of the interaction between requirements and how conflicts can be solved
	Participants		Identification and description of the participants that interact with the system
Modelling	Structure	Class Diagram	Structure of the pattern elements
		Object description	Description of the objects and their accountabilities
	Behaviour	Collaboration or sequence diagrams	Description of the scenario
		Activity or state diagrams	Description of the scenario
Solution Variants			Description of alternative solutions
Resulting Context			System configuration after application of pattern
Consequences			Advantages and disadvantages of pattern application
Anti-Pattern Traps			The most common traps that can result from the application of the pattern
Examples			One or more application examples
Related Patterns			List of related patterns
Design Patterns			Design patterns that can be used to refine the analysis patterns later in the design phase
Design Guidelines			Tips how the solution should be implemented
Known Uses			Description of known pattern applications in existing systems. At least three systems should be mentioned.

3.1.4 More Pattern Templates in the Literature

There are more templates in the literature, the attributes of which offer nothing new in comparison to the templates presented here and which do not achieve the comprehensiveness of the templates from PANTOQUILHO et al. (2003). They are discussed individually in the works of FERREIRA and LOUCOPOULOS (2001), HAMZA (2002), LISBOA et al. (2002) and KONRAD et al. (2004).

3.2 Pattern Organisation

The organisation of analysis patterns is an important prerequisite for their application, as at first a suitable analysis pattern must be located and selected before it can be applied. This shows that the aim is to guarantee the retrievability of patterns. Only the pattern that can be located by its potential user can be re-used. This means that even the best documented pattern can fail in its purpose – to pass on the knowledge of known problems' solutions – if it is badly organised and the user has no idea of its existence.

HENNINGER and CORREA (2007) have researched the present state of the organisation of design patterns. They come to the conclusion that the abundance of various sources that are difficult to find and can hardly be searched through – or cannot be searched through at all – is utterly counter-productive for the purpose of patterns, which is their re-use. Henninger and Correa's research did not cater to the quality of documentation of patterns (e.g. documentation in a template).

According to the literature overview (see chapter 2) carried out for this report, the present organisational state of analysis patterns is similar. Even if the number of the published analysis patterns is not anything like as large as that of the design patterns, their retrievability is very poor. The patterns are available either in books or in the proceedings of conferences. Patterns in book form are not freely available, as the user has to first own the book in order to know about the patterns and the problems they are supposed to solve. Furthermore, this form of organisation does not allow a tool-based electronic search for a suitable pattern; the same limited accessibility applies to the conference proceedings. Some conferences (e.g. Pattern Languages of Programs (PLoP)) offer free access to the articles of the proceedings on their conference web-sites; other conferences (e.g. IEEE International Conf. on IRI) only allow use of the articles for a fee. In both cases it is tiresome and time-consuming to find analysis patterns among the wealth of contributions. Even then there is no guarantee of finding a suitable pattern for a particular problem.

3.2.1 Pattern Collections

Pattern collection is a generic term for all the ways in which several patterns are documented together; this could be a book, a web-site, an article in a journal or a conference publication. The patterns in pattern collections usually come from the same author and are documented in the same or similar fashion. A pattern collection needs not be organised in itself; but if it is organised, then it belongs to one of the two types of collections, that is, pattern language or pattern catalogue.

Conference proceedings cannot count as pattern collections, as their main purpose is not the collection of patterns, but the documentation of the entire annual conference results. Because of their distribution over many web-sites which do not have a search function, not even the minimum of accessibility a pattern collection should provide is granted.

3.2.2 Pattern Languages

A pattern language is a collection of related patterns for solving a specific problem, with the pattern relationships forming the organisation within the pattern language. The pattern language grammar

consists of rules that dictate how the patterns are combined together (and, if necessary, in which order) in order to solve the problem.

A pattern language is, then, a form of organisation of related patterns that are to be used in accordance with specified rules. BRAGA et al. (2007) explain the way in which a pattern language for analysis patterns can be created.

Several pattern languages for analysis patterns have been published (see section 2); they are, however, usually only published in conference proceedings, and are therefore not easily accessible.

3.2.3 Pattern Catalogues

A pattern catalogue is a collection of individual patterns which are organised according to at least one criterion (e.g. the domain in which they solve a problem): the book FOWLER (1997) is a pattern catalogue, for example. The patterns are organised based on the domains in which they were found, and the catalogue can be searched via the table of contents.

Pattern catalogues can take various forms (web-site, book etc.) and are different from mere pattern collections through an element of organisation. The higher the degree of organisation (measured e.g. by the number of available sorting criteria), the easier it is for the catalogue user to find a suitable solution for his or her problem; an electronic catalogue with a search function offers an even higher degree of accessibility. Unlike a pattern language a pattern catalogue contains patterns which are generally not related among each other.

Up till now, analysis pattern catalogues have only been available as print books (e.g. COAD et al. 1995, ARLOW and NEUSTADT 2004). Although most other analysis patterns are documented electronically, it is only in conference proceedings, something that considerably reduces their accessibility. The only known attempt at a freely available pattern catalogue for analysis patterns is that of DE NÓ (2005, see section 2.3.6).

3.3 Ideal Documentation of Analysis Patterns

As the previous chapters have already made clear, there has so far been no centralised, freely accessible facility for organising and documenting analysis patterns. Many authors in the literature have recognised the need for a uniform pattern template (e.g. PANTOQUILHO et al. 2003) and the need for the organisation of analysis patterns in a catalogue (e.g. LISBOA et al. 1998, FERREIRA AND LOUCOPOULOS 2003, DE NÓ 2005). In the following, an attempt will be made to establish characteristics and criteria for the ideal documentation of analysis patterns with the help of existing approaches in the literature.

Features that a pattern catalogue should have:

- It should be easy to access from anywhere in the world
- The user should be able to expand it and improve it
- It should have patterns that are comparable and comprehensible
- It should have an efficient search function
- It should have patterns that are categorised
- It should be thorough and complete
- It should support CASE Tools

Today, a web-based Wiki (WIKIPEDIA 2008a) may probably represent the best form for a pattern catalogue (see also DE NÓ 2005). A Wiki available in the internet offers free access and also the function inherent in each Wiki, which is the possibility for addition and extension through the user. Changes of a pattern by a user require a version management; thus the development of a pattern can be tracked and a potentially dissatisfying change can be reversed. In order to avoid language barriers the catalogue must be kept in English.

In order to guarantee the comparability and comprehensibility of the patterns provided by the catalogue, the patterns have to be entered using the most comprehensive template possible. The template serves as a guideline for the author as soon as the pattern is placed into the catalogue. At the same time it helps the user to understand the patterns more easily and to compare several of them. Currently, one of the most suitable templates for it would probably be that of PANTOQUILHO et al. (2003) which was introduced in 3.1.3.

An effective search function is a key component of a pattern catalogue, as a pattern that is not quickly and efficiently identified as suitable will not be re-used. For this reason, an efficient classification scheme, such as that of FERREIRA and LOUCOPOULOS (2001) (see 2.18) is needed. The classification is an obligatory feature for all entered patterns, and allows a targeted search for patterns for particular types of problems. At the same time, the catalogue can also be simply flicked through based on terms from the term space.

Presumably, considerable time and effort will be required in order to achieve a complete pattern catalogue; completeness, however, is an important criterion if the catalogue is to be accepted by its users.

The setting-up of a complete analysis pattern catalogue with all the functions required is a project of large dimensions, and should therefore not be carried out by individuals but by an international institution or an organisation (e.g. Object Management Group). This ensures that the catalogue can be established lastingly, and that resources for its maintenance are available.

The import of existing analysis patterns into the catalogue constitutes a great challenge. The patterns must be transferred into the catalogue template from their various original forms and classified; this work requires experts in the field of analysis patterns. Ideally, the respective authors can be persuaded to enter their patterns into the catalogue.

The entry of the patterns into the catalogue should go together with the setting-up of some convenience functions. For example, the pattern models in the catalogue (static as well as dynamic) should be available not only as plain graphics files, but also as XMI files (XML Metadata Interchange). In this way, an analysis pattern can be easily imported into most UML CASE Tools for instantiation.

A further key element for the success of a pattern catalogue is that it is widely known of and accepted. Only an active user-community that consistently enhances existing patterns and enters new patterns into the catalogue guarantees its long-term success and thus prevents organisation and documentation problems with analysis patterns. One step towards this would be campaigns carried out in the scientific as well as in the business fields, for example at conferences. Here it is important to convince companies of the advantages of an active participation by their employees in the expansion of the catalogue, and also to combat possible reservations that company secrets would be made public.

4 Application of Analysis Patterns

After the previous chapter's presentation of the conditions necessary for an efficient application of analysis patterns, this chapter will deal with important aspects of their application. This includes the introduction of two fundamental methods of analysis pattern application. In addition, the traceability of analysis patterns after their application in the finished analysis model is examined. An example of the creation of an analysis model with the help of patterns is provided, and the chapter finishes with an evaluation of the pros and cons of analysis pattern application.

4.1 Methods for the Application of Analysis Patterns

This section gives an extensive description of two methods for the application of analysis patterns, which will then be matched to the approaches discussed in the literature.

The methods are presented in the context of a simplified software development process (see Fig. 4.1). This consists of the analysis of the requirements, the creation of the analysis model, the creation of the design model, and the implementation. Other phases (for example, test, maintenance) are omitted, as they do not offer any additional contextual information. The methods presented develop the creation phase of the analysis model to a higher degree, with the analysis pattern application explicitly not being limited only to the static aspects of the model, but also extending to the dynamic ones.

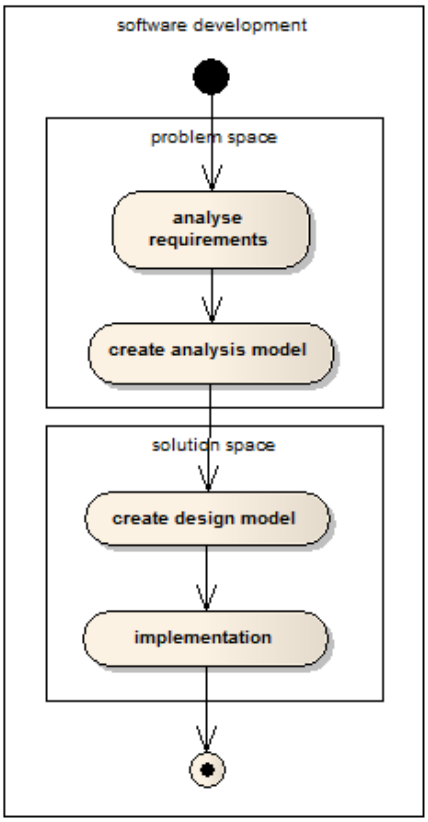


Fig.4.1. Simplified software development process

Both methods, the Flexible Pattern Oriented Modelling Method (PMM), as well as the Validation and Optimisation Method (VOM), sum up the approaches that have been presented in the literature so far. All the methods that have been previously introduced can be subsumed under them. They

were named according to their properties as they affect the way of pattern application. They are illustrated in the form of a UML activity diagram.

Both methods require the availability of one or more sources of analysis patterns (e.g. pattern catalogue); they are, however, designed in such a way that they allow modelling even when there is no support from analysis patterns available.

4.1.1 Flexible Pattern-Oriented Modelling Method (PMM)

Fig. 4.2 shows the sequence of the Flexible Pattern-Oriented Modelling Method (PMM). This method is characterised by the integral use of analysis patterns in the analysis model development. It allows the use of analysis patterns at every stage of the creation of the model.

The method starts after the requirements analysis and, depending on the available patterns, begins either with the identification of the central classes of the model (*identify central classes*), or directly with the determination of a part of requirements for which a patern is searched (*select new part of requirements*). The first alternative turns out to be useful for specific pattern catalogues. If the patterns of COAD et al. (1995) or NICOLA et al. (2002) are used, their low complexity requires that the process begins with the modelling of the central classes, and the model is only completed afterwards with the support of patterns (see 4.1.3.2).

If no matching pattern can be found for the requirements under consideration (*no matching pattern found*), the corresponding part of the model has to be modelled by hand (*model selected part of requirements from scratch*). If, on the other hand, a matching pattern is found *(matching pattern found*), then this is instantiated (*instantiate analysis pattern*) and subsequently integrated into the model (*integrate analysis pattern into model*).

Then either the search for another analysis pattern begins, (*analysis model not complete*) or the model is complete and the design phase can be commenced.

As we shall see later the PMM combines the known approaches for the application of analysis patterns from the literature (see 4.1.3) in a flexible manner. Particular value was put on the method allowing itself to be adapted to the available analysis patterns at every stage of the model development. In particular the PMM can (formally) also be applied if only a pattern catalogue of limited size is available.

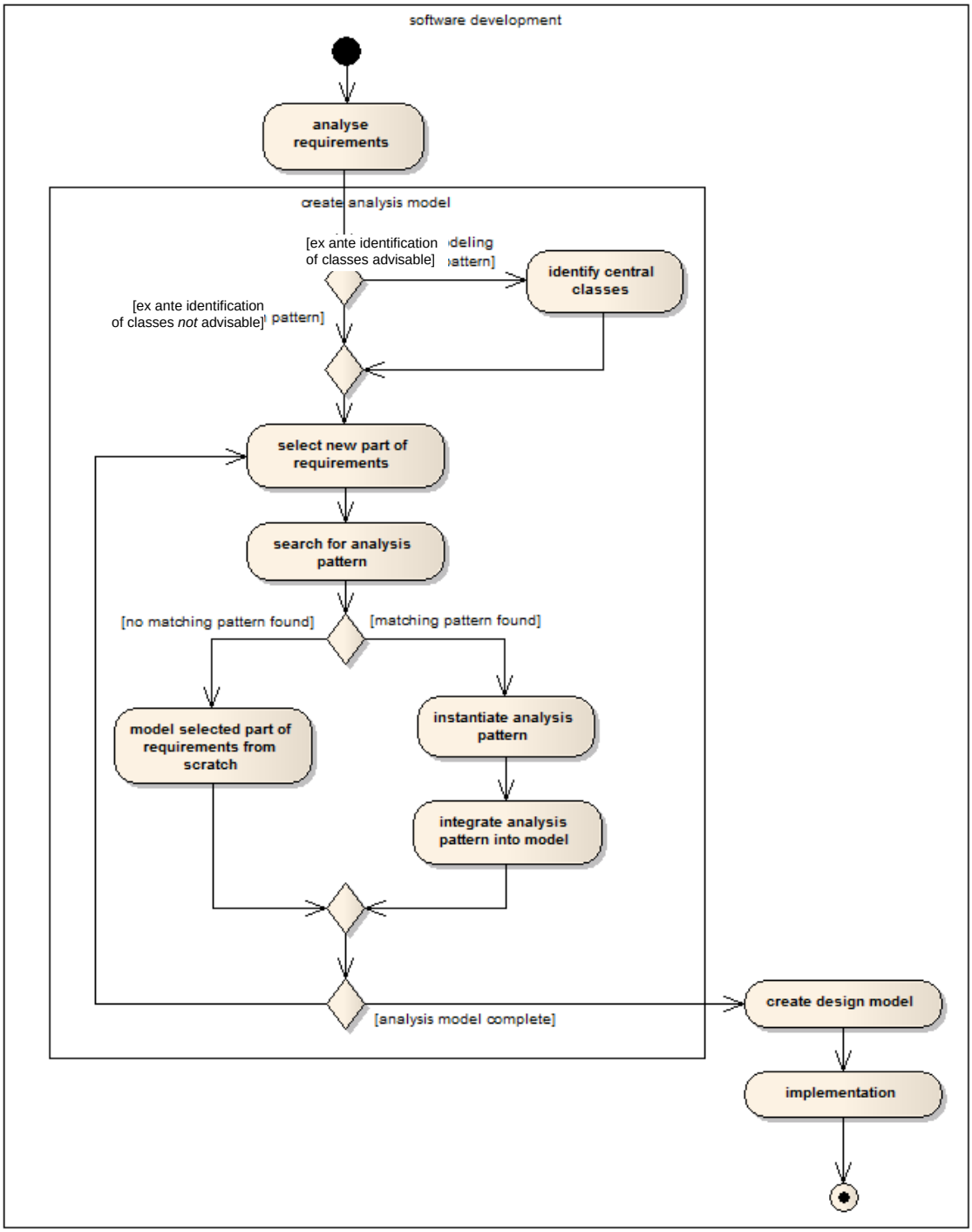


Fig. 4.2. Flexible Pattern Oriented Modelling Method

4.1.2 Validation and Optimisation Method (VOM)

The sequence of the Validation and Optimisation Method (VOM) is shown in Fig. 4.3. With this method analysis patterns are applied only after completion of the analysis model; the existing model is then validated and optimised with the help of matching analysis patterns.

The creation of the analysis model and the review of the model with analysis patterns do not have to be carried out by the same people. The method can also be applied, therefore, in order to improve older models or models that were created by others. For example, when advisers join a project that has already been started, with VOM they can check the existing analysis model.

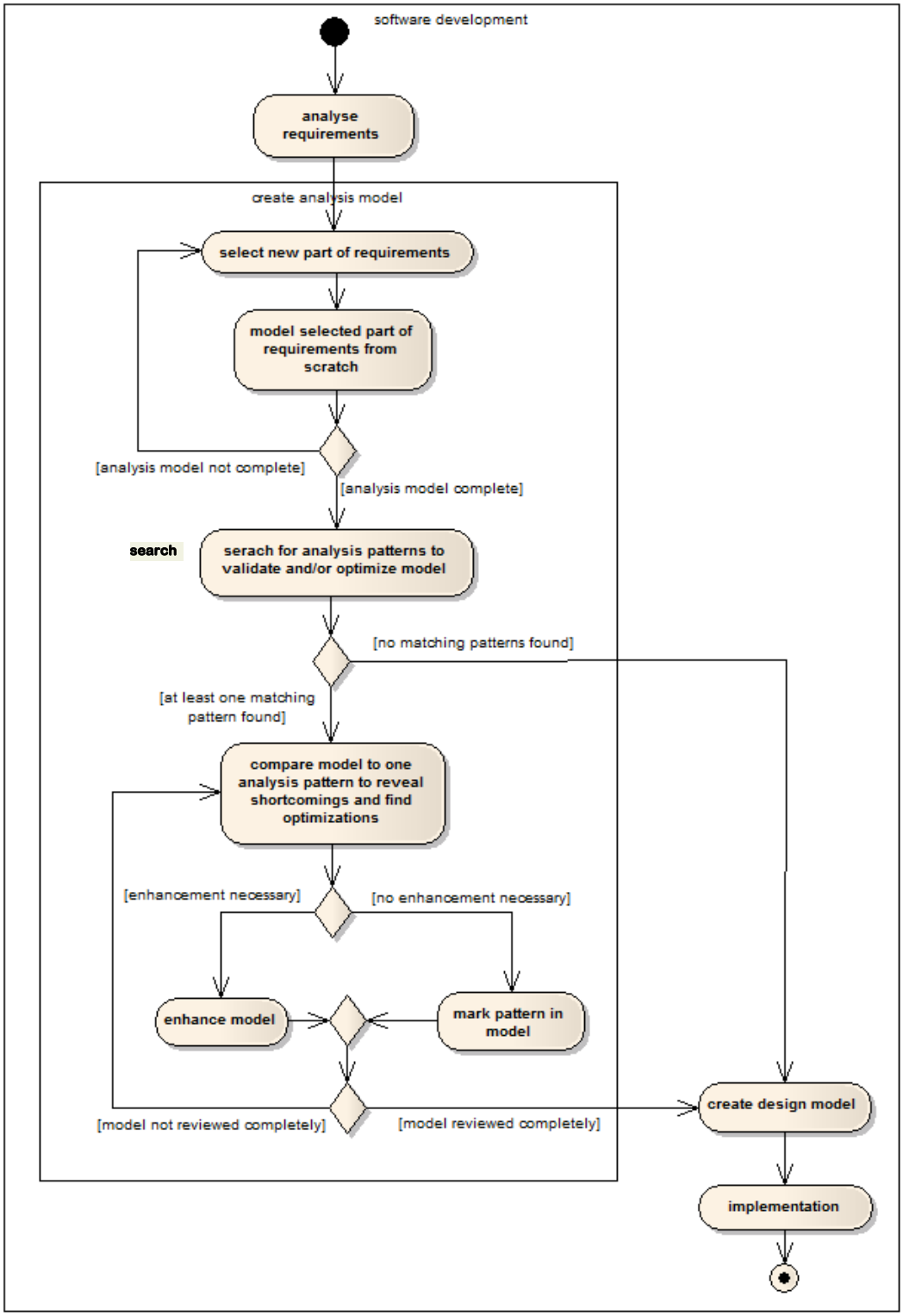


Fig.4.3. Validation and Optimisation Method

After the requirements analysis the analysis model is modelled without pattern support (*model selected part of requirements from scratch*). This activity is continued until the model appears to be complete (*analysis model complete*).

Finally, the model is validated and optimised with analysis patterns. All the patterns that match the problems solved in the analysis model have to at first be searched for – e.g. in a pattern catalogue

(*search for analysis pattern*). If no matching patterns are found (*no matching patterns found*), the design phase is commenced.

If at least one matching pattern is found (*at least one matching pattern found*), the first pattern (either the only one or the most suitable one) is compared to the model in order to detect deficits and find opportunities for optimisation (*compare model to one analysis pattern*). If an enhancement is necessary or possible (*enhancement necessary*), it is carried out (*enhance model*). If no enhancement is necessary, the pattern is marked in the model in order to improve the documentation and the comprehensibility (*mark pattern in model*). If the review of the model is complete, with all patterns found having been applied, the design phase can be commenced (*model reviewed completely*). If the review is not yet complete, the model is compared to the next pattern, and so on.

Again, the VOM can (formally) also be used if only a small pattern catalogue is available. Obviously the main difference between the methods PMM and VOM relates to the moment of pattern application. While in PMM patterns are integrated in the analysis model as soon as possible the VOM envisages the integration of patterns only after a complete model has been established.

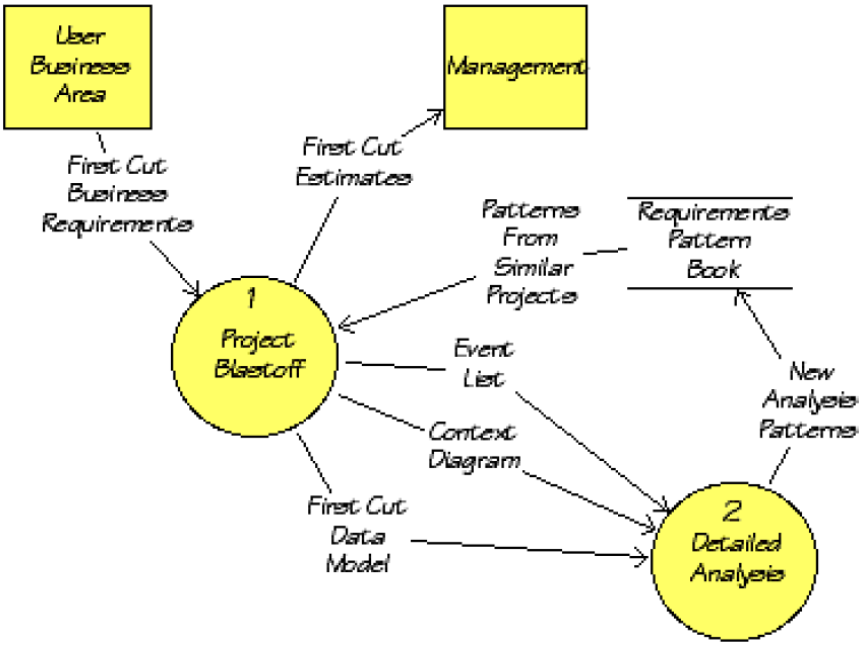
4.1.3 Application Methods in the Literature

In the following, the approaches for the application of analysis patterns from the literature examined in chapter 2 are presented and compared to the methods introduced in 4.1.1 and 4.1.2. Only explicitly described approaches were taken into account, implicit descriptions, e.g. through examples, are not considered here.

4.1.3.1 Suzanne Robertson, Kenneth Strunch

ROBERTSON and STRUNCH (1993) present a method that at the start of a project includes, first of all, a search for matching patterns for the analysis model. The method does not require the analysis to be started with a pattern; if no pattern is found, the analysis is continued manually. The method described by Robertson and Strunch is analogous to PMM.

Fig. 4.4 shows the analysis process according to ROBERTSON and STRUNCH (1993). During the so-called ‘Project Blastoff’ an attempt is made to identify patterns from the pattern book that are suitable for the project. New patterns that are discovered during the ’Detailed Analysis’ are included in the pattern book.



Source: based on ROBERTSON AND STRUNCH 1993

Fig. 4.4. Analysis process with patterns based on Robertson and Strunch

4.1.3.2 Peter Coad, David North, Mark Mayfield

The pattern-application method suggested by Coad et al. (COAD et al. 1995, p. 290) is tailored to their pattern catalogue introduced in the same paper. Because of the low complexity of the patterns (only two classes), the central classes of the model are identified at first; their relationship to each other is established with the help of the patterns from the pattern catalogue. Thus the analysis model is created through repeated application of the patterns until all the classes identified are related to each other. This method can be seen as a variation from the PMM, which requires the availability of suitable patterns.

4.1.3.3 Martin Fowler

Fowler suggests two kinds of analysis pattern application. On the one hand, he uses patterns as a starting-point for a model (FOWLER 1997, p.12), from which point the remaining part of the analysis model can be developed. This corresponds to the PMM. On the other hand, he uses patterns for the review of existing models (FOWLER 1997, p.12), which corresponds to the VOM.

4.1.3.4 Jugurta Lisboa Filho, Cirano Iochpe, Kate Beard

LISBOA et al. (1998) give a short description of the manner in which the analysis-phase process changes through the application of analysis patterns. After the conclusion of the requirements analysis, patterns should be sought that fulfil the requirements for their use in the analysis model. This process can be illustrated with PMM. Lisboa et al. do not provide more detailed information or a description of the process.

4.1.3.5 Eduardo B. Fernandez, Xiaohong Yuan

FERNANDEZ and YUAN'S (2000) method for the application of the semantic analysis patterns starts with a search for suitable analysis patterns. It can be illustrated with the PMM. They consider the availability of a pattern catalogue with different types of analysis patterns and even design patterns

to be a basic prerequisite. The first step involves the application of the semantic analysis patterns. A check is carried out to see whether there are already matching or similar concrete patterns for the implementation of the use cases and requirements, which may then be used. If such patterns are not found, an attempt is made to specialise matching and abstract patterns in a suitable way. Smaller patterns (sub-patterns) can then be used for those requirements which have not yet been met or have only been partly met. If no more matching patterns are found, the model has to be completed through ad hoc modelling. In this way the modelling process can be considerably accelerated, as large parts of the model are combined together from patterns. At the same time the quality of the model is improved, as semantic analysis patterns are derived from 'good' models that are already functioning.

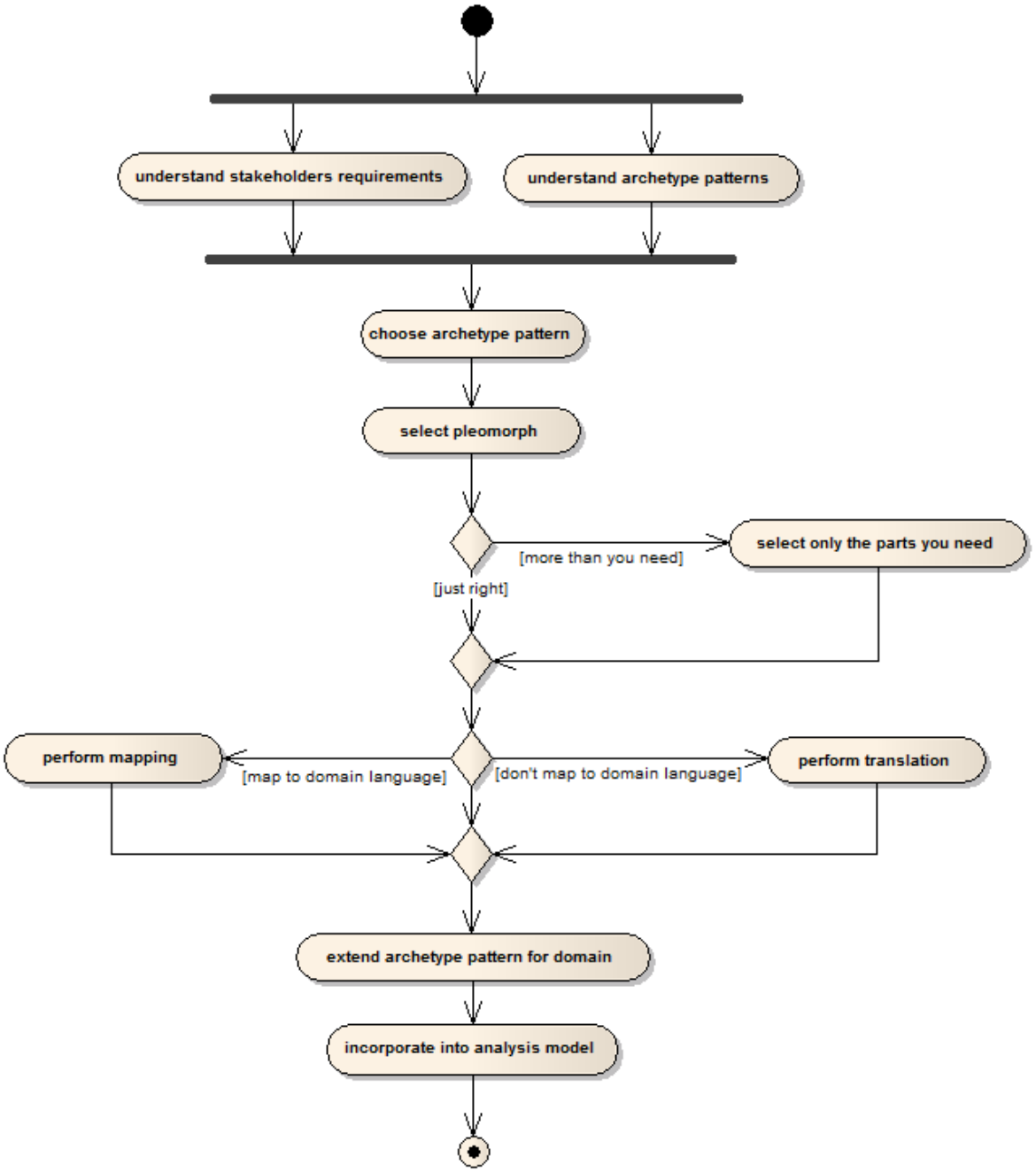
4.1.3.6 Sascha Konrad, Betty H. C. Cheng, Laura A. Campbell

For the application of analysis patterns in embedded systems, KONRAD et al. (2004) use a method that selects suitable patterns as a first step after the requirements analysis. It is assumed that there is an adequate pattern catalogue available. Their method can be illustrated with the PMM.

4.1.3.7 Jim Arlow, Ila Neustadt

The method described by ARLOW and NEUSTADT (2004, p. 31) for the application of archetype patterns can be illustrated with the PMM. It starts with the selection of a matching pattern, and in their method Arlow and Neustadt assume that there is always a matching pattern available for the problem.

Archetype patterns can either be applied 'manually' in the conventional way, or in the automated MDA process. Fig. 4.5 shows the process for the manual instantiation of archetype patterns. First of all, a suitable pattern is chosen in accordance with the requirements; then the most suitable pattern variation is selected and everything of the pattern that is not needed is omitted. Finally, the pattern is instantiated and, if necessary, domain-specific aspects are added, before it is integrated into the analysis model.



Source: based on ARLOW and NEUSTADT 2004, p.31

Fig.4.5. Process for the instantiation of business archetype patterns

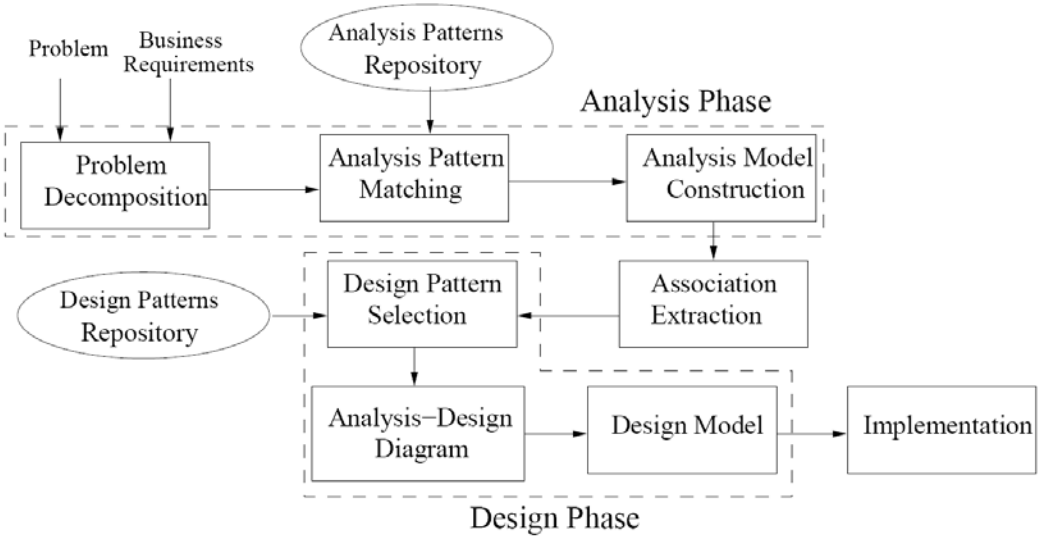
4.1.3.8 Narasimha Bolloju

In his work BOLLOJU (2004) names two ways of applying analysis patterns. First of all he optimises existing analysis models with the help of patterns. This means that the model is compared to patterns in a pattern catalogue and thus possible errors and omissions are spotted, which corresponds to the VOM. Then, in order to complete the model, and similar to Coad et al., he observes the application of patterns onto classes that have already been identified. This process can be illustrated with the PMM.

4.1.3.9 Haitham Hamza and Yi Chen

HAMZA and CHEN (2005) present an analysis and design method supported by patterns, Pattern-Driven Analysis and Design Method (PAD). The part of the method related to the analysis phase starts with the search for a suitable analysis pattern, whereby a pattern catalogue is assumed to be available. This method can be illustrated with the PMM.

The PAD method (Fig. 4.6) is founded on the systematic use of patterns in all phases of software development, from the analysis up to the implementation. Here, the analysis model is developed with analysis patterns, and then converted into a design model with the help of design patterns. The systematic application of the patterns improves their effectiveness still further.



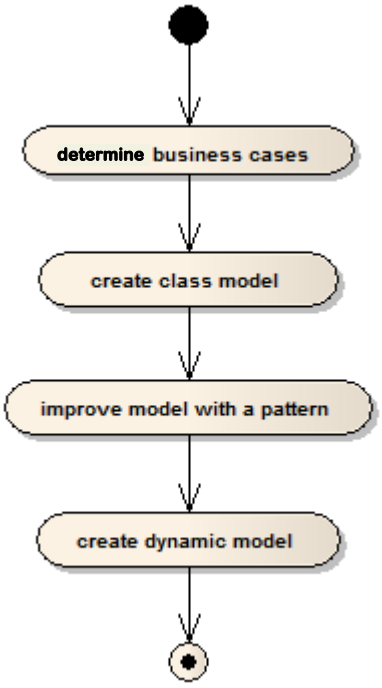
Source: HAMZA and CHEN 2005

Fig.4.6. The PAD method

4.1.3.10 Diego Calvo de Nó

In his Master's thesis DE NÓ (2005) documents two methods for applying analysis patterns. The method he describes as the integration method aims at the improvement of analysis models at a later stage, and corresponds to the VOM. De Nó's second method, which corresponds to the PMM, is described by him as the "building-set method", and it starts with the pattern search.

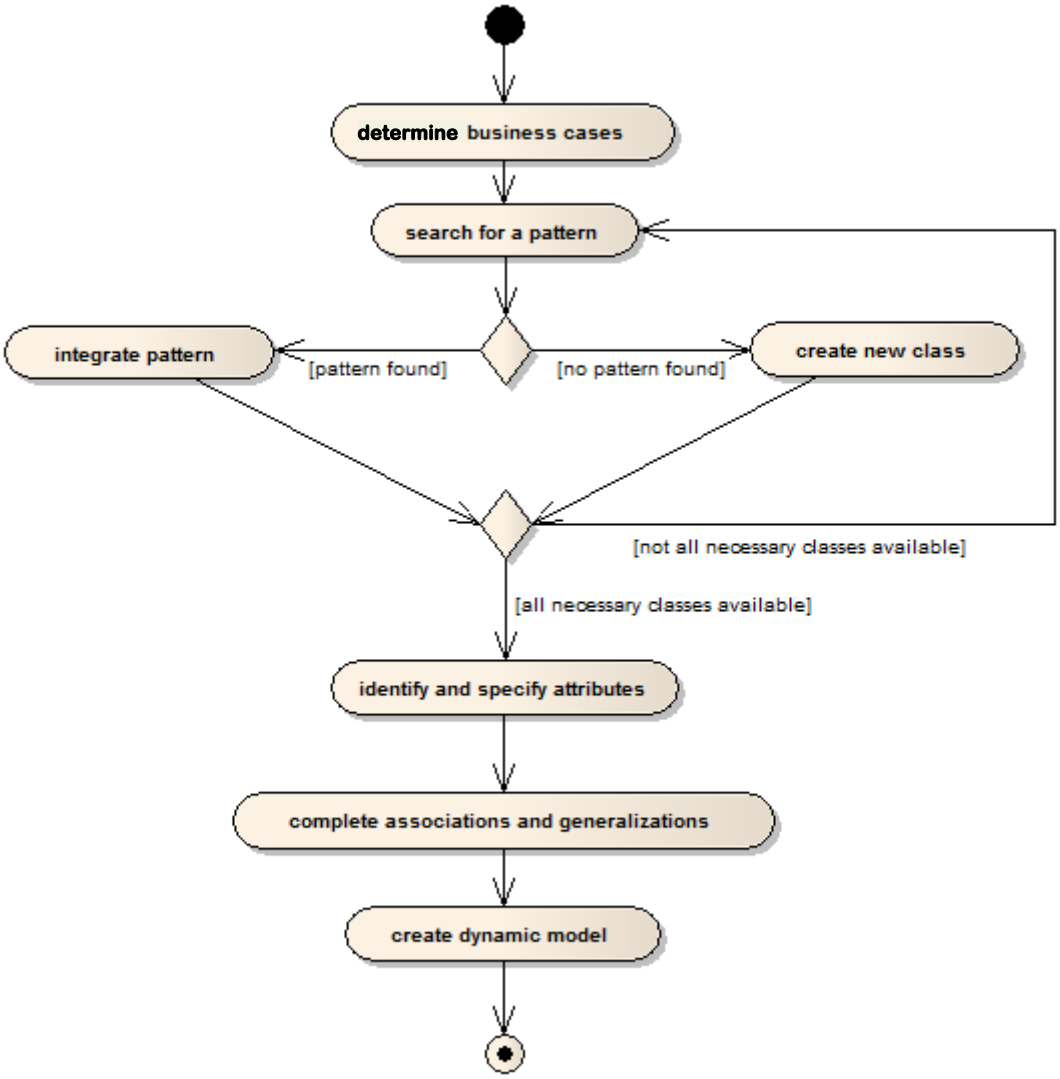
The integration method comprises primarily the improvement and validation of an already existing analysis model. This process had already been described, but not explicitly visualised (Fig. 4.7), by BOLLOJU (2004) before de Nó described it.



Source: DE NÓ 2005, p. 13

Fig. 4.7. Integration-method process

De Nó's building-set method distinguishes itself through the application of analysis patterns at the very beginning of the modelling process (Fig. 4.8). If possible, the modelling of the analysis model is started with the instantiation of a pattern. Similar methods for pattern application had already been presented in the literature before de Nó (e.g. ARLOW and NEUSTADT 2004, p. 31). De Nó, however, is the first to explicitly describe the possibility of resorting to manual modelling if no suitable pattern is found.



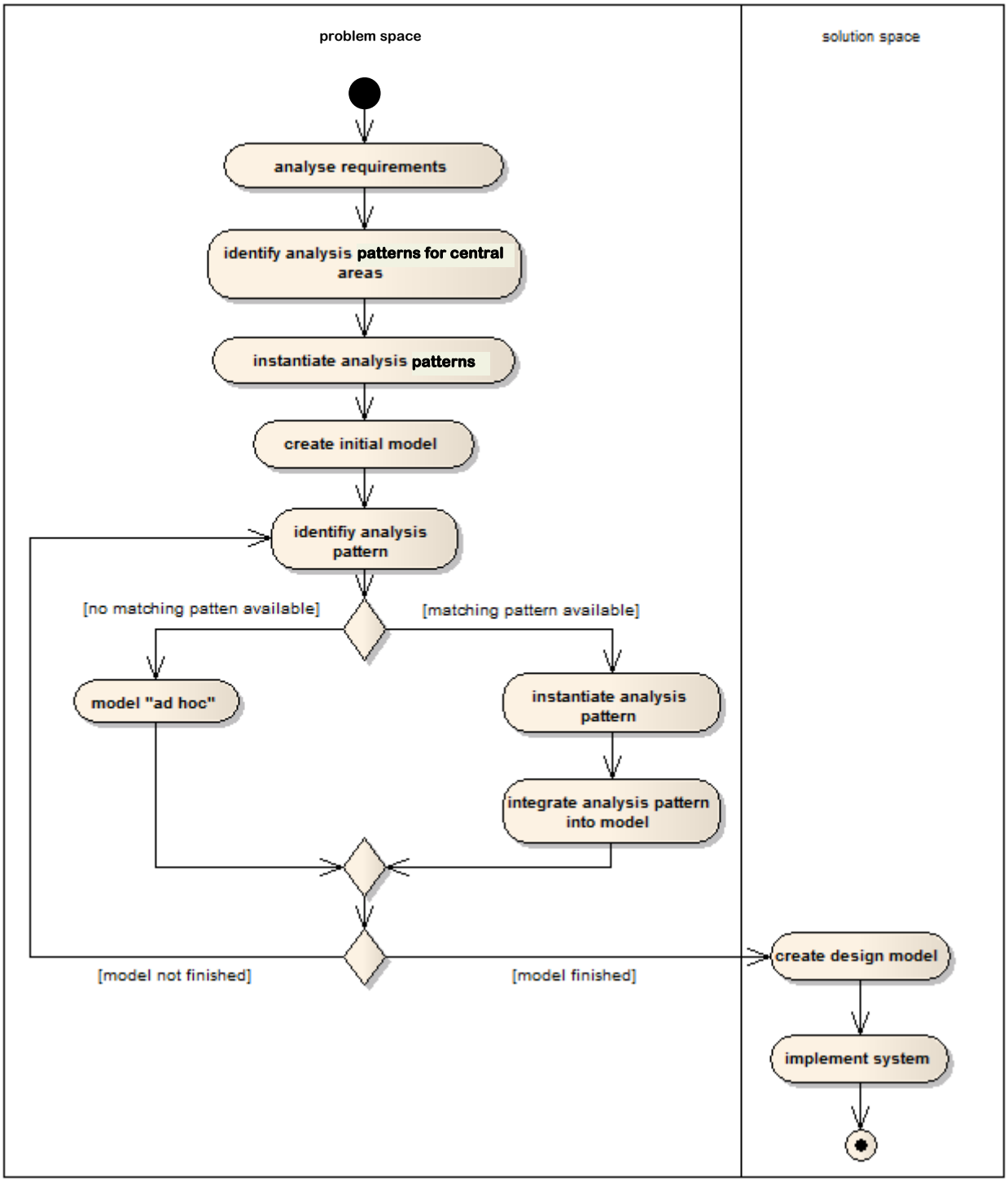
Source: DE NÓ 2005, p. 20

Fig. 4.8. Building-set method process

4.1.3.11 Ralph Kemp

In his Master's thesis (KEMP 2006) Kemp presents four methods of analysis-pattern application ('pattern-selection ante', 'pattern-selection intra', pattern-selection ante et intra' and 'pattern-selection post'). The methods 'pattern-selection ante' and 'pattern-selection intra' can be considered to be special cases or variations of the PMM. The method 'pattern-selection ante et intra' corresponds to the PMM and the method 'pattern-selection post' to the VOM.

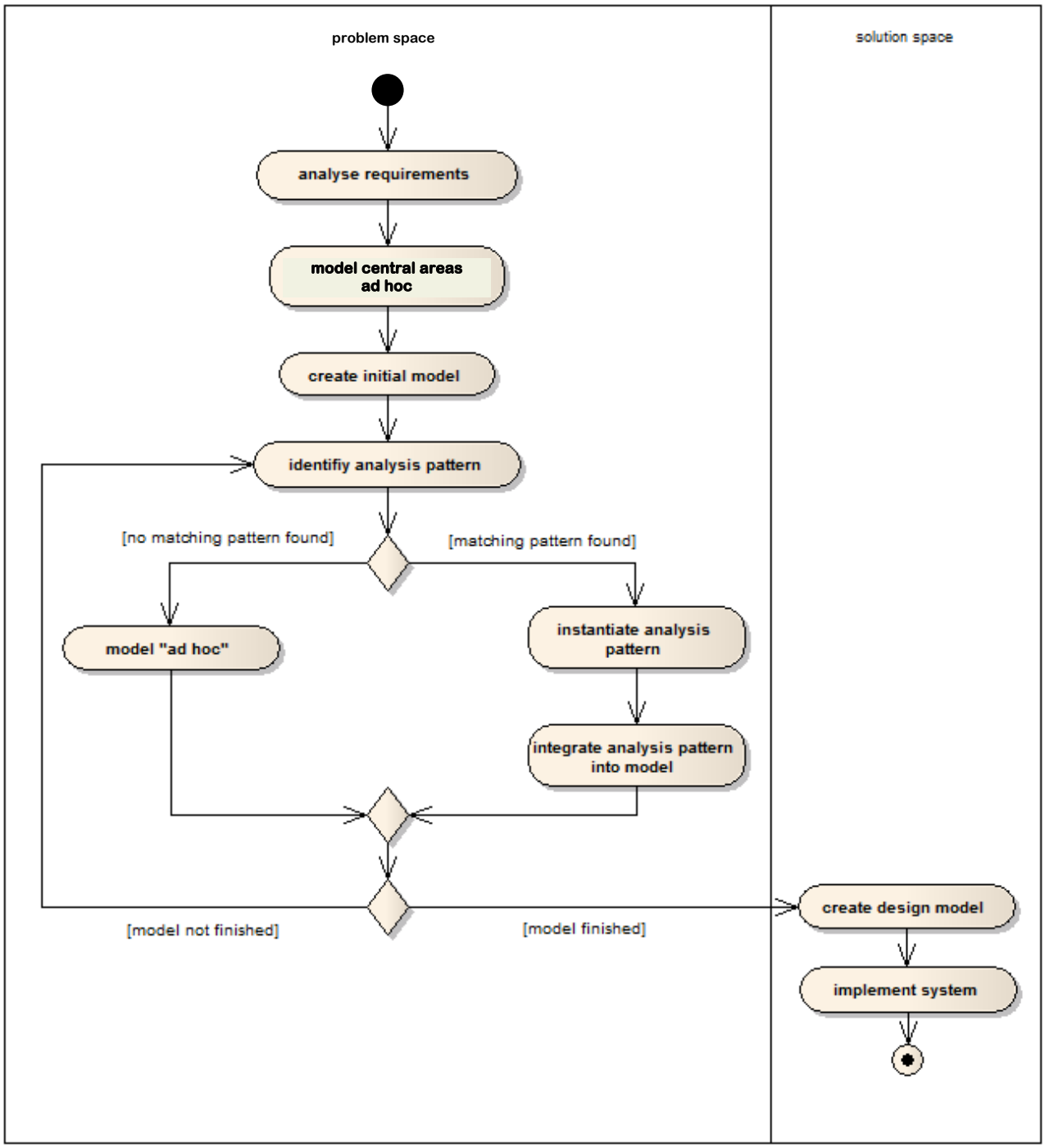
The 'pattern-selection ante' describes a process that strictly requires the creation of an analysis model to be started with the use of one or more analysis patterns (Fig. 4.9).



Source: KEMP 2006, p. 44

Fig. 4.9. 'Pattern-selection ante' process

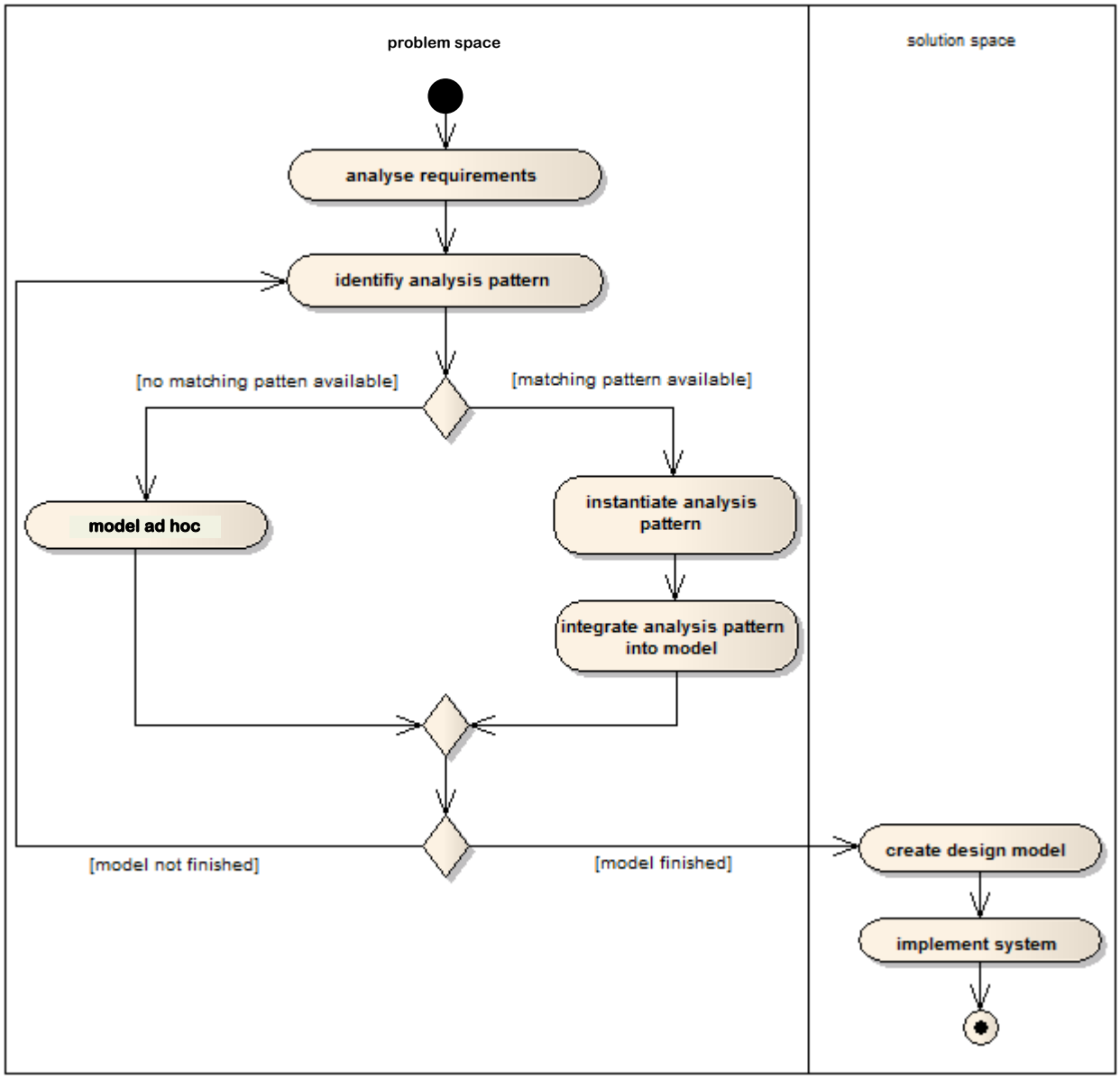
The process 'pattern-selection intra' must begin with the creation of an ad hoc starting model, and allows the use of analysis patterns only for refinement (Fig. 4.10).



Source: KEMP 2006, p. 45

Fig.4.10. 'Pattern-selection intra' process

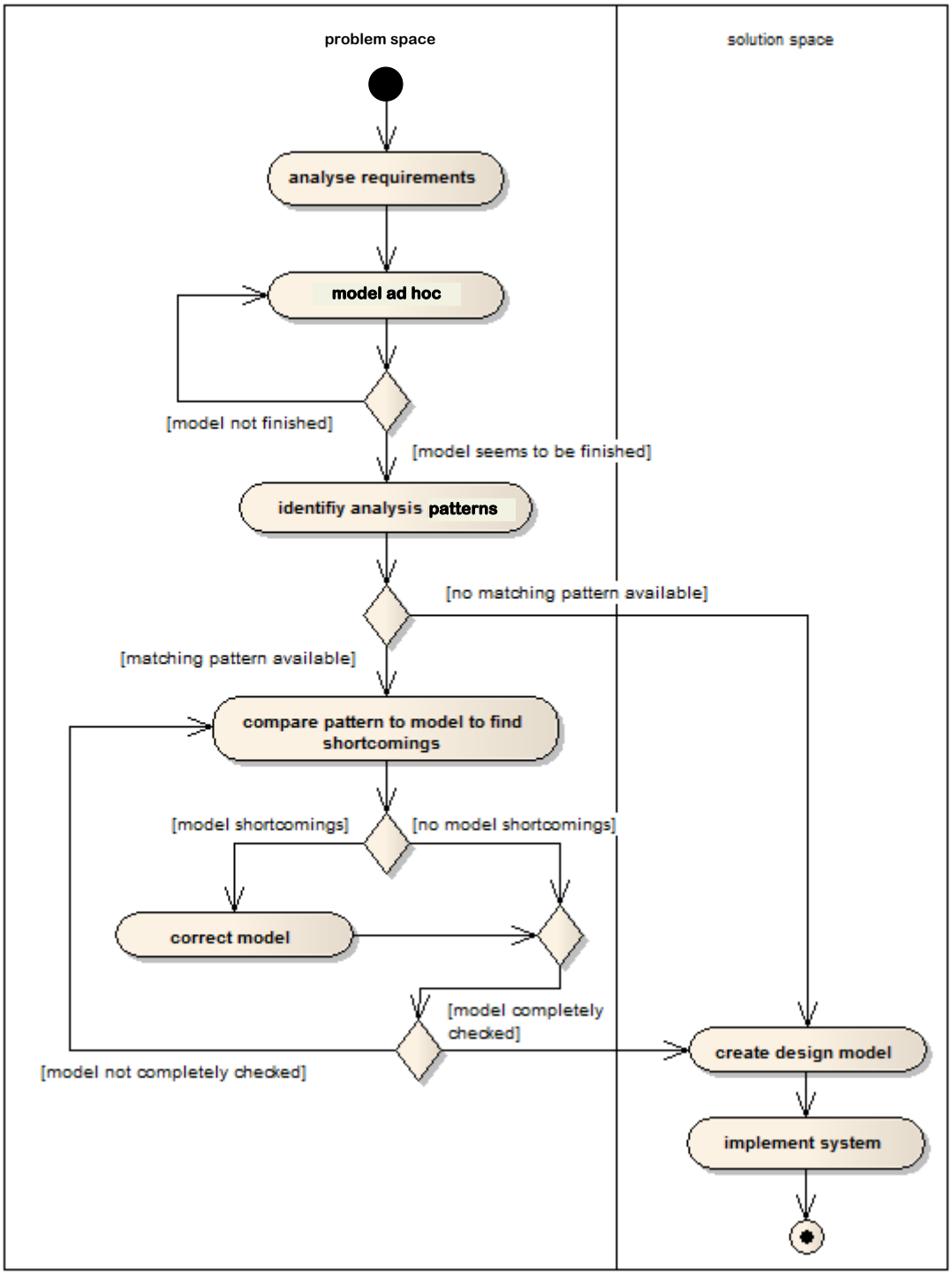
The application using the process 'pattern-selection ante et intra' is the most flexible. If analysis patterns are available as the starting point, the modelling can be begun with them. If no matching patterns are available, the modelling can also be begun ad hoc.



Source: KEMP 2006, p. 47

Fig. 4.11. 'Pattern-selection ante et intra' process

The 'pattern-selection post' process describes the improvement and validation of a model with the help of analysis patterns at a later stage.



Source: KEMP 2006, p. 48

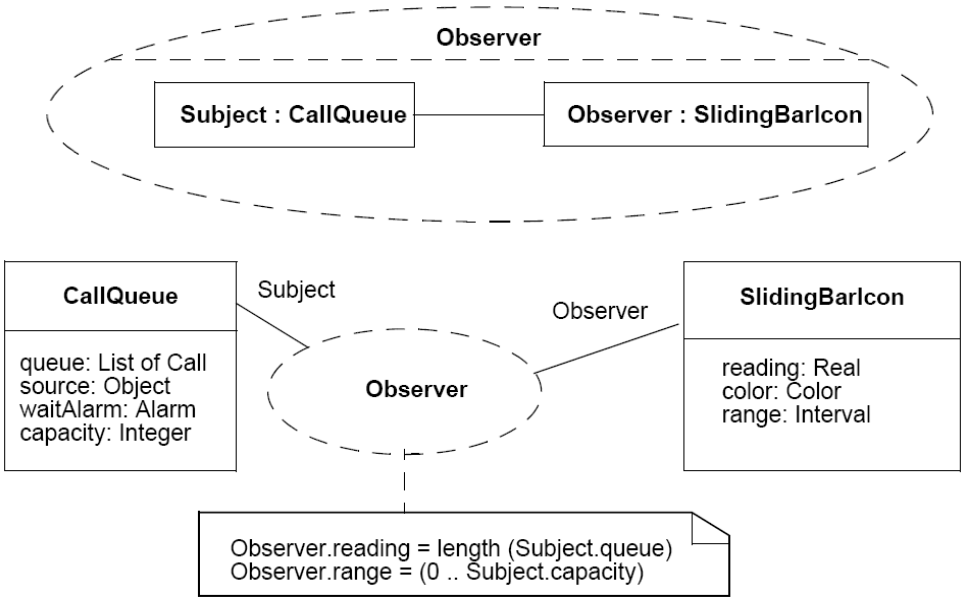
Fig. 4.12. 'Pattern-selection post' process

4.2 Traceability of the Pattern Applications in the Analysis Model

The ability to trace analysis patterns in a model will in most cases only be significant after its completion. The knowledge of which patterns were used in a model makes later changes considerably easier, for example when working with a software development process with several iterations, or in the maintenance phase. Users who are not familiar with a model but are familiar with the patterns in it will then be able to understand the model more quickly, at a model review, for example.

However, the traceability of patterns in a model can easily be lost, as numerous patterns are used in analysis models when there are complex problems to solve. When these patterns are instantiated, class names, attributes, relationships, etc. are adapted to the problem context, so that only the structure of the original pattern is recognisable. This, however, might be changed during the integration of the analysis pattern into the model (omitting pattern parts that are not needed, adding of more classes, merging with already existing classes), which results in the analysis pattern being absorbed into the model and no longer being easy to recognise.

In the literature it is BENDER (1999, p. 37ff) who first proposed a possible solution for this problem. He suggests bordering the patterns, shading this area in colour, and giving it the name of the pattern. If there are complex models with many converging patterns, however, this method causes the clarity to suffer and does not take into account the dynamic elements of the model. According to HAMZA and FAYAD (2004a), it is sufficient to use only stable analysis patterns (see 2.22), as their structure and, as a consequence, their traceability is retained during the modelling. This, though, has the disadvantage that for modelling only stable analysis patterns may be used. The Unified Modelling Language (UML 2007, p. 170) likewise offers a notation for the labelling of patterns; for the so-called 'collaboration', however (Fig. 4.13), the usage is only intended for standard design patterns . This notation also has the disadvantage of lacking clarity in more complex class diagrams and not providing traceability for the dynamic elements of the model. Note that model classes frequently simultaneously belong to multiple patterns.



Source: UML 2007, p. 170

Fig.4.13. UML Collaboration Notation using the example of the Observer Design Pattern

Up till now, none of these proposals has been taken up by the authors that have provided examples of application with their patterns.

The most promising approach towards marking the origin of the classes in an analysis model has come from KEMP (2006, p. 59). He suggests labelling each class that is an instance of a class from a pattern with a UML stereotype. The stereotype notation would then be «pattern name.meta-class name», with the meta-class name representing the name of the class from the pattern. If a class originates from several patterns (merging during the integration, see above), a stereotype is added to it for every one of the original patterns. This approach to labelling preserves the clarity even in complex class diagrams, and enables a simple reading of the origin of each class. If dynamic model elements – for example, sequence diagrams – are being created, the stereotypes allocated in the class diagram are taken over as well (using a suitable CASE tool), an approach that guarantees the traceability, even in the dynamic model.

For the reasons given in 4.1, the notation introduced by KEMP (2006) is at first translated into English. In addition, the pattern origin is supplemented with another important piece of information, the pattern catalogue from which the pattern originates. This guarantees that, in the case of patterns with the same name or a similar name, the pattern origin is clearly stipulated. The reasons for the importance of the pattern catalogue for the application of analysis patterns are explained in detail in 4.1 and in 3.

Consequently, the expanded notation is «PatternCatalog.PatternName.PatternClassName». PatternCatalog indicates the pattern catalogue that is used; PatternName, the pattern used from the catalogue; and PatternClassName denotes the class name used in the analysis pattern. Fig. 4.14 shows the notation applied to a class and gives an example for the labelling of the class 'Person,' instantiated as 'Employee' from FOWLER's (1997) party pattern.

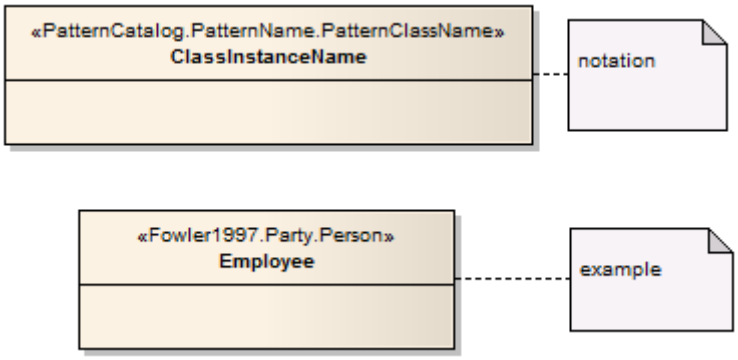


Fig.4.14. Notation and example for the labelling the origin of a class

4.3 Example for the Application of Analysis Patterns

This section demonstrates with the help of a simple example the application of analysis patterns with the Flexible Pattern Oriented Modeling Method (PMM). For reasons of space, the analysis is limited to the most important use case, and the analysis model is only described as far as it is necessary for the demonstration of the PMM. The validation of the example, e.g. through the modelling of the dynamic elements, was deliberately ignored. This is the only way in which a complex non-trivial example for the application of the PMM could be demonstrated without going beyond the scope of this study. The extent of the example was considered to be of more importance than its validation.

4.3.1 Computer Online Shop Case-Study

Computer plc makes PCs and laptops. So far it has only supplied its products to large retailers; now, however, private customers as well as companies will be able to order products directly online.

The customers will be able to choose and order the desired products in an online shop. Their request will create an order in the system, generate an invoice, and arrange the despatch.

The data for the product catalogue will not be processed by the online shop system; it is taken directly from the existing company database.

4.3.2 Central Use-Case

The central Use-Case of the online shop is the placing of the order by the customer (Fig. 4.15). The process of ordering triggers various other processes.

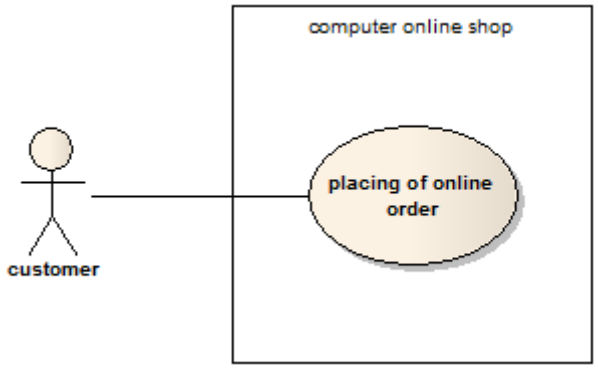


Fig. 4.15. Central use-case of the computer online shop

The following use-case template (from BORTFELDT 2001) describes the use-case 'placing of online order.'

Aim:
A customer order is to be converted into an order, an invoice and a delivery.
Agent:
Customer
Category:
Primary
Initiating event:
The customer has sent his order in the online shop.
Precondition:
The customer has registered and placed his product in the basket.
Postcondition if successful:
The order is stored in the system and the delivery process is started.
Postcondition if unsuccessful:
The order could not be stored and an error message is issued to the customer.
Standard specifications:
1 The customer places the order
2 The system checks the order and stores it
3 The system generates an invoice for the order
4 The system generates a despatch order
Extension of the standard specifications:
1c The customer cancels his order
Alternatives for the standard specifications: –.

4.3.3 Creation of the Analysis Model

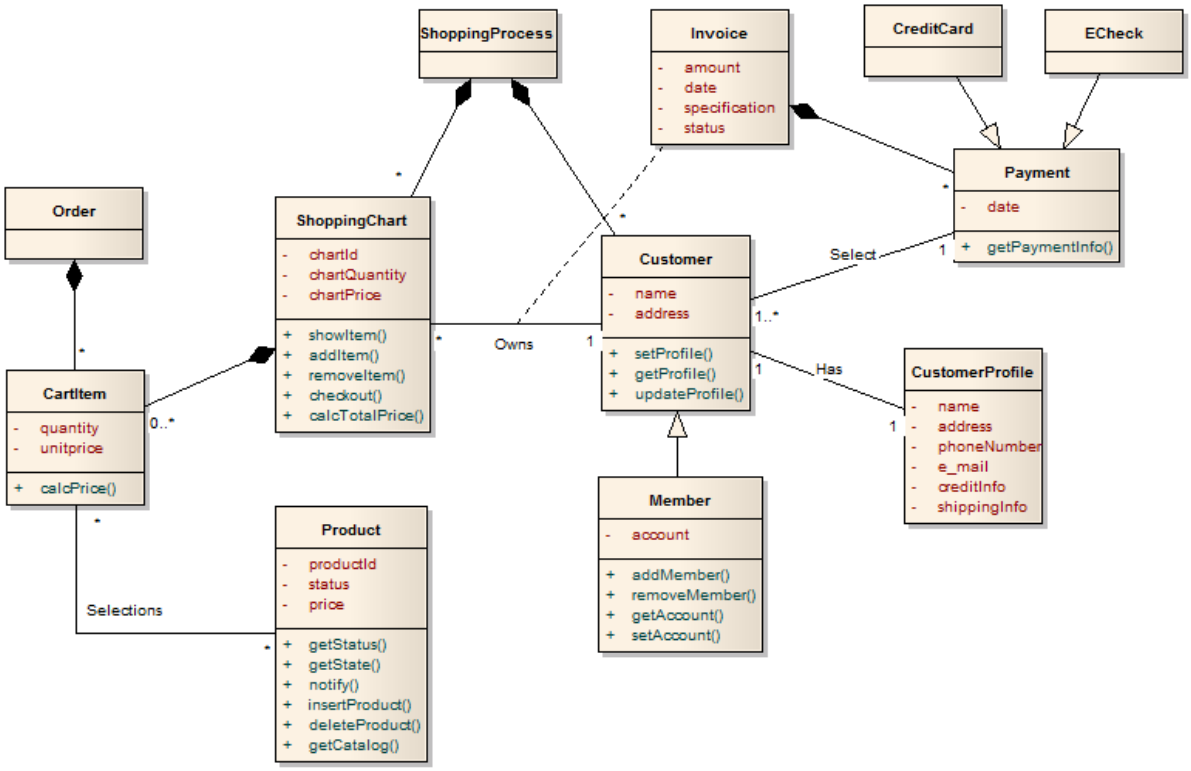
In the following the creation of the analysis model for the computer online shop is described. This involves the use of the Flexible Pattern Oriented Modelling Method (PMM) introduced in chapter

4.1.1. Suitable analysis patterns were chosen from the literature presented in chapter two. Owing to the lack of a pattern catalogue, the selection was carried out by hand through a search in books and PDF files.

First of all, the pattern that best describes the online shop system was chosen.

4.3.3.1 Search for and Instantiation of the First Analysis Pattern

The first pattern identified as suitable is the 'Shopping Process Pattern' from FERNANDEZ et al. (2001b).



Source: FERNANDEZ et al. 2001b

Fig. 4.26. Shopping Process Pattern

It has all the classes necessary for an online shop. A customer is given a profile and to order he takes on the role of the registered customer. He can add products to his shopping basket and delete them. The invoice is generated depending on the chosen method of payment.

Fig. 4.17 shows the instantiated Shopping Process Pattern. The 'ShoppingProcess' class was omitted, as it is not required. For each class of the instantiated pattern, the corresponding class name in the original pattern was entered as the stereotype (see 4.2).

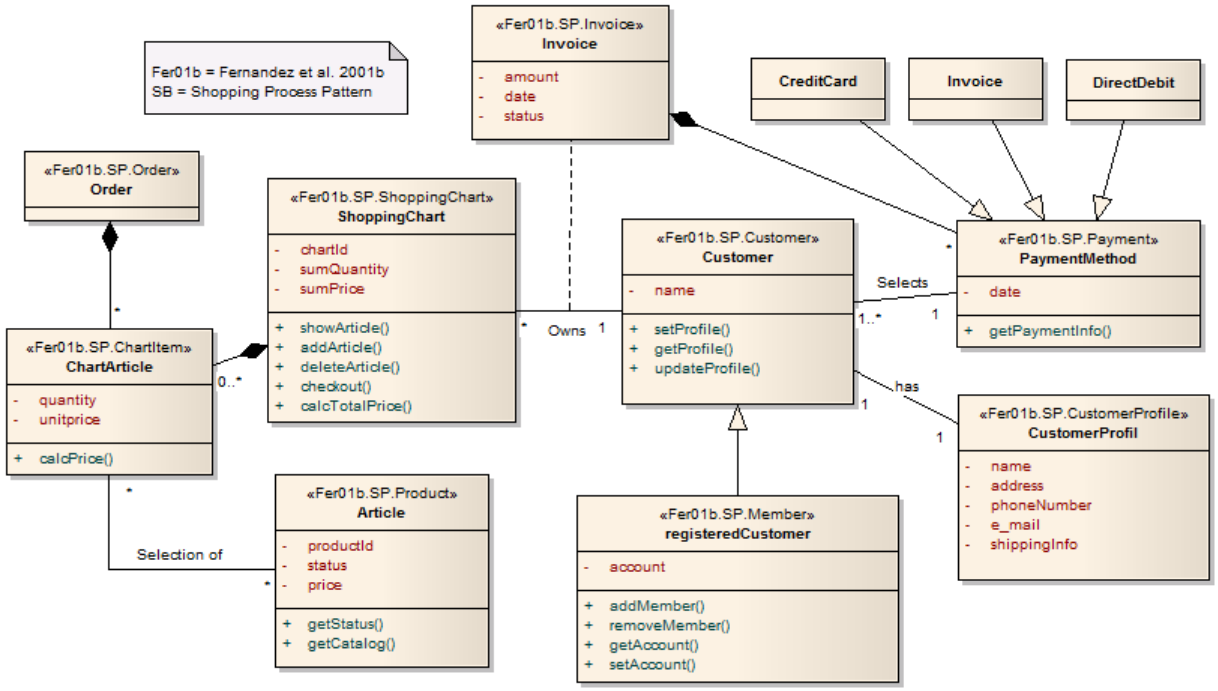
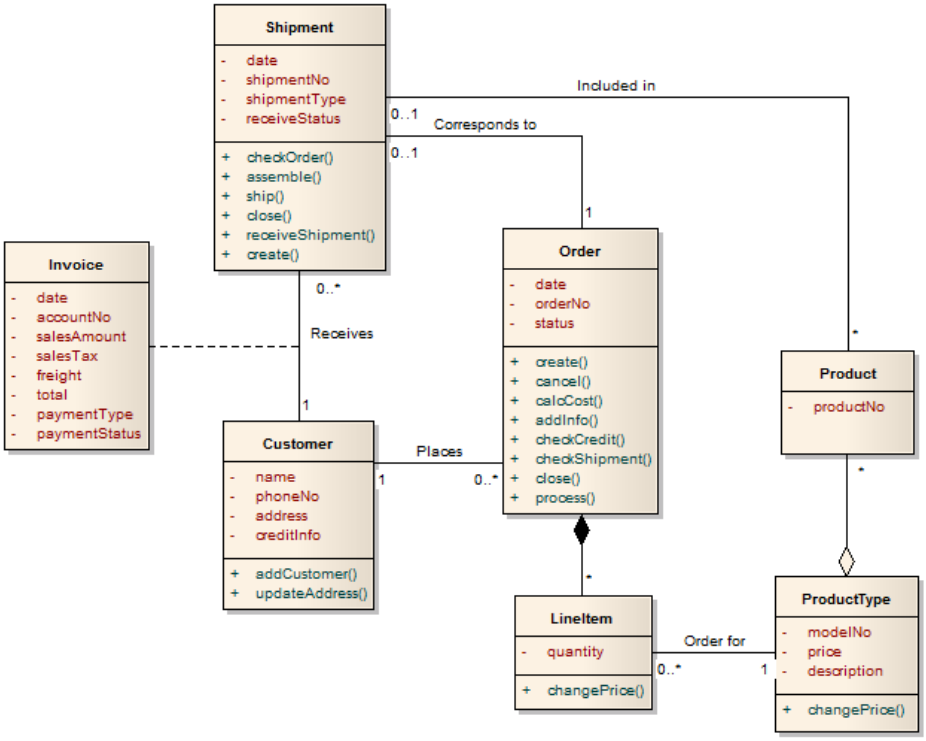


Fig. 4.17. Instantiated Shopping Process Pattern

The instantiated Shopping Process Pattern has now become the initial model. Building upon this, another analysis pattern is now searched for that expands the model according to the requirements.

4.3.3.2 Search for and Instantiation of a Second Analysis Pattern

The second pattern identified as matching is the ‘Order and Shipment of a Product Pattern’ from FERNANDEZ et al. (2000c).



Source: FERNANDEZ et al. 2000c

Fig. 4.18. Order and Shipment of a Product Pattern

The pattern comprises the new class ‘Shipment’ and supplements the classes that already exist in the initial model with new attributes and operations regarding delivery.

Fig. 4.19 shows the instantiated ‘Order and Shipment of a Product Pattern.’ Attributes and operations that are not needed (e.g. customer data that is to be stored in the customer profile) have already been omitted.

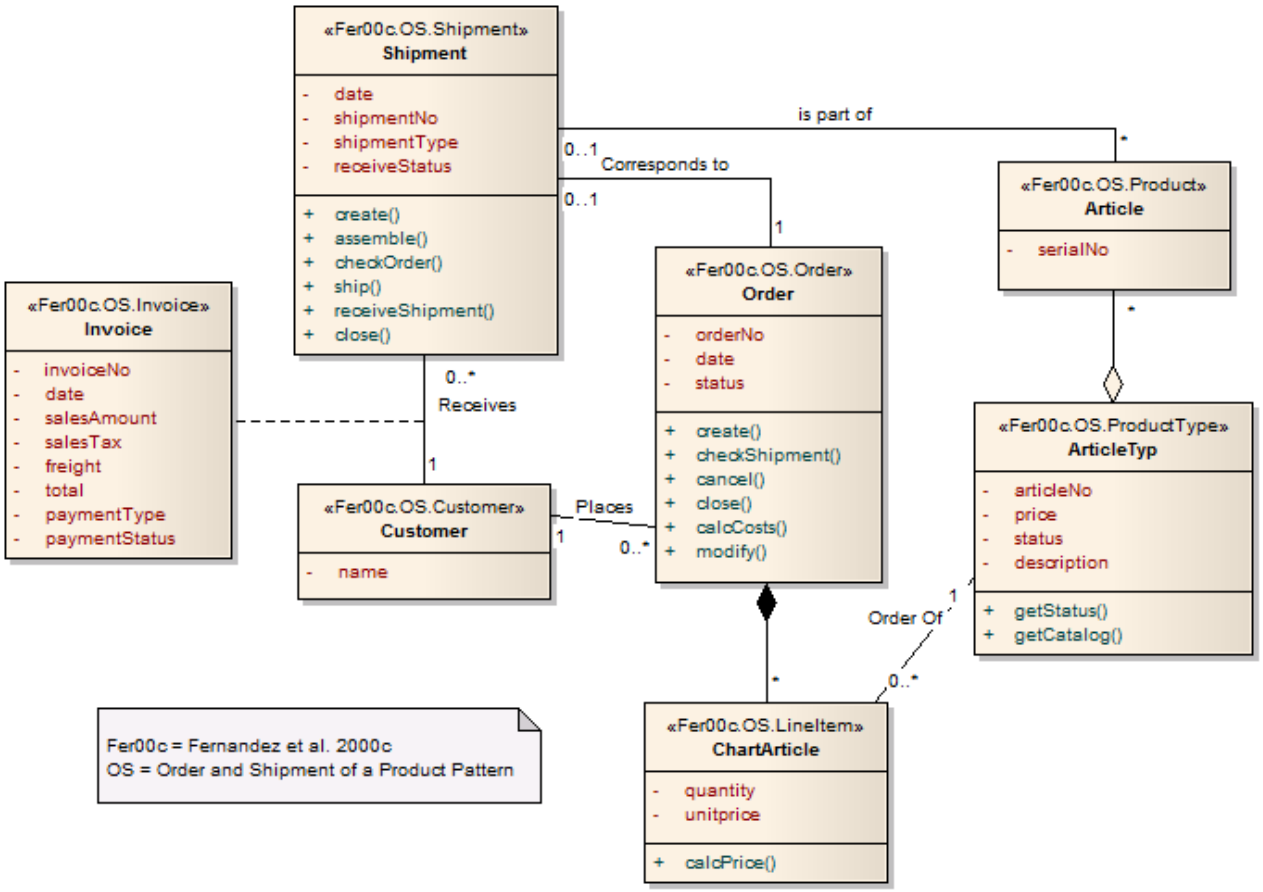


Fig.4.19. Instantiated Order and Shipment of a Product Pattern

4.3.3.3 Integration of the Second Pattern into the Model

Fig. 4.20 shows the integration of the instantiated ‘Order and Shipment of a Product Pattern’ into the initial model. The pattern origin of the individual classes can be easily read from the stereotype of each class.

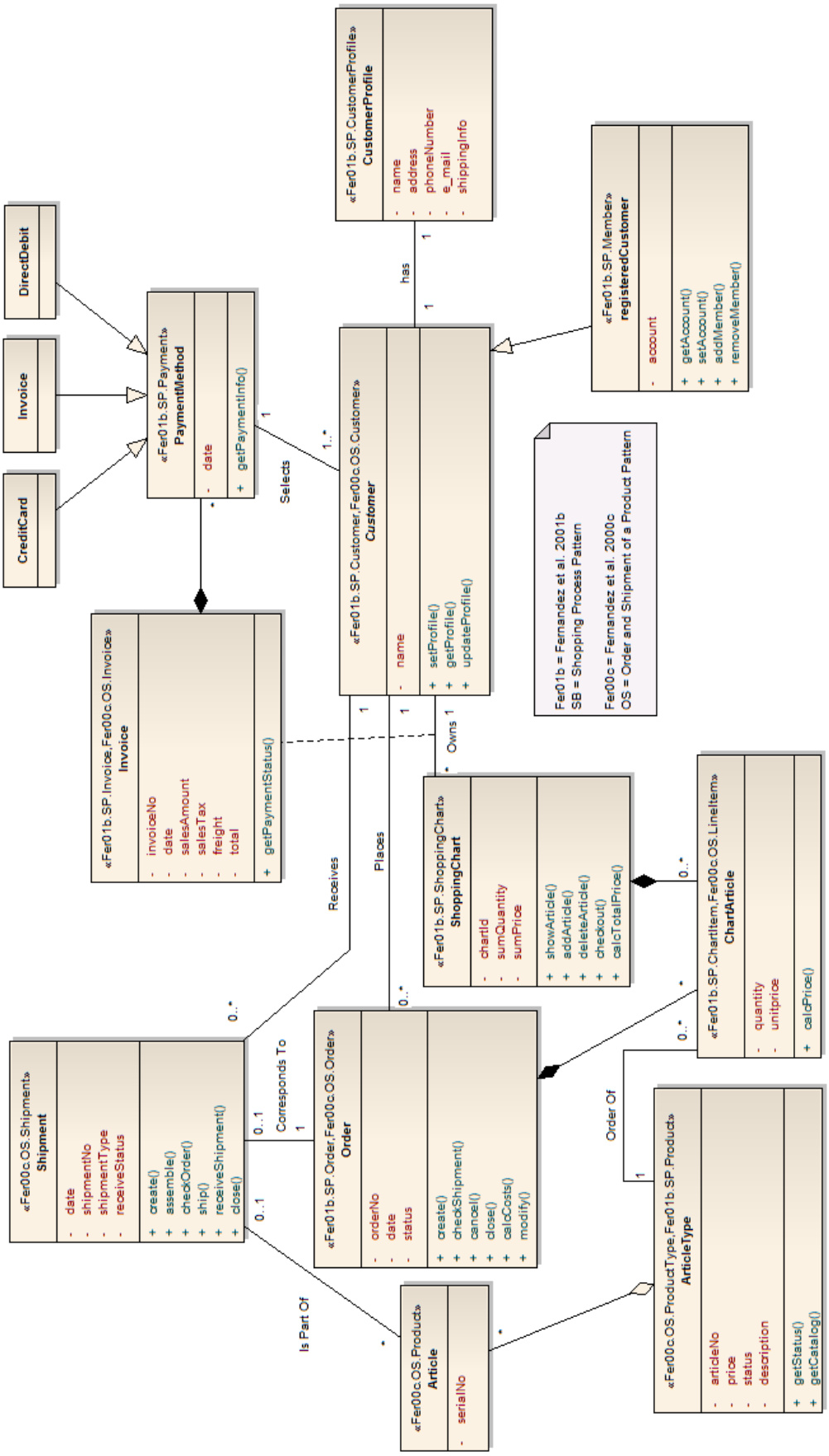
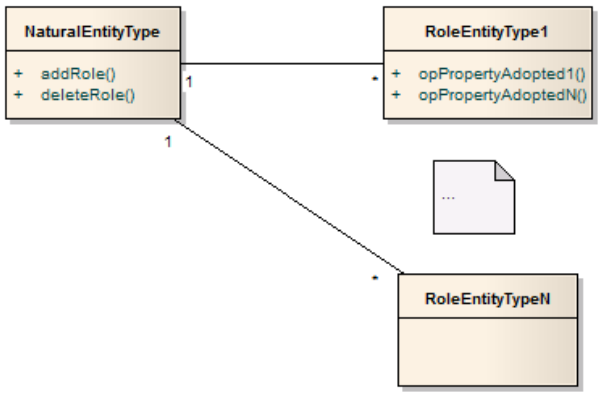


Fig.4.20. Integration of the Order and Shipment of a Product Pattern into the model

4.3.3.4 Search for and Instantiation of a Third Analysis Pattern

The third pattern identified as matching is the 'Roles as Entity Types Pattern' from CABOT and RAVENTÓS (2004).



Source: CABOT and RAVENTÓS 2004

Fig.4.21. Roles as Entity Types Pattern

The pattern is used to relate the role of 'registered customer' to the customer through an association, thus replacing the original inheritance relationship.

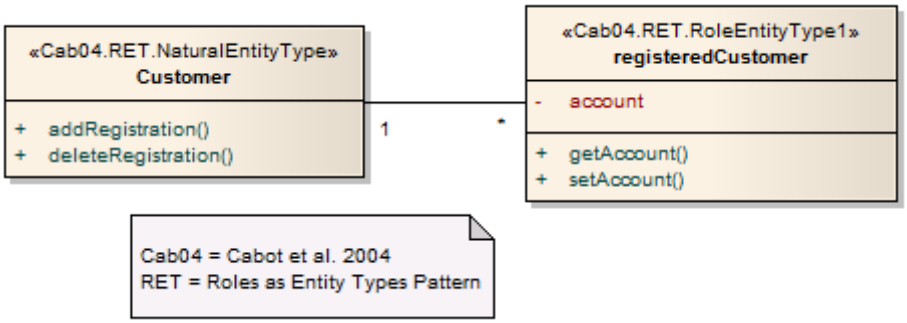


Fig. 4.22. Instantiated Roles as Entity Types Pattern

4.3.3.5 Integration of the Third Pattern into The Model

Fig. 4.23 shows the integration of the instantiated 'Roles as Entity Type Pattern' into the analysis model.

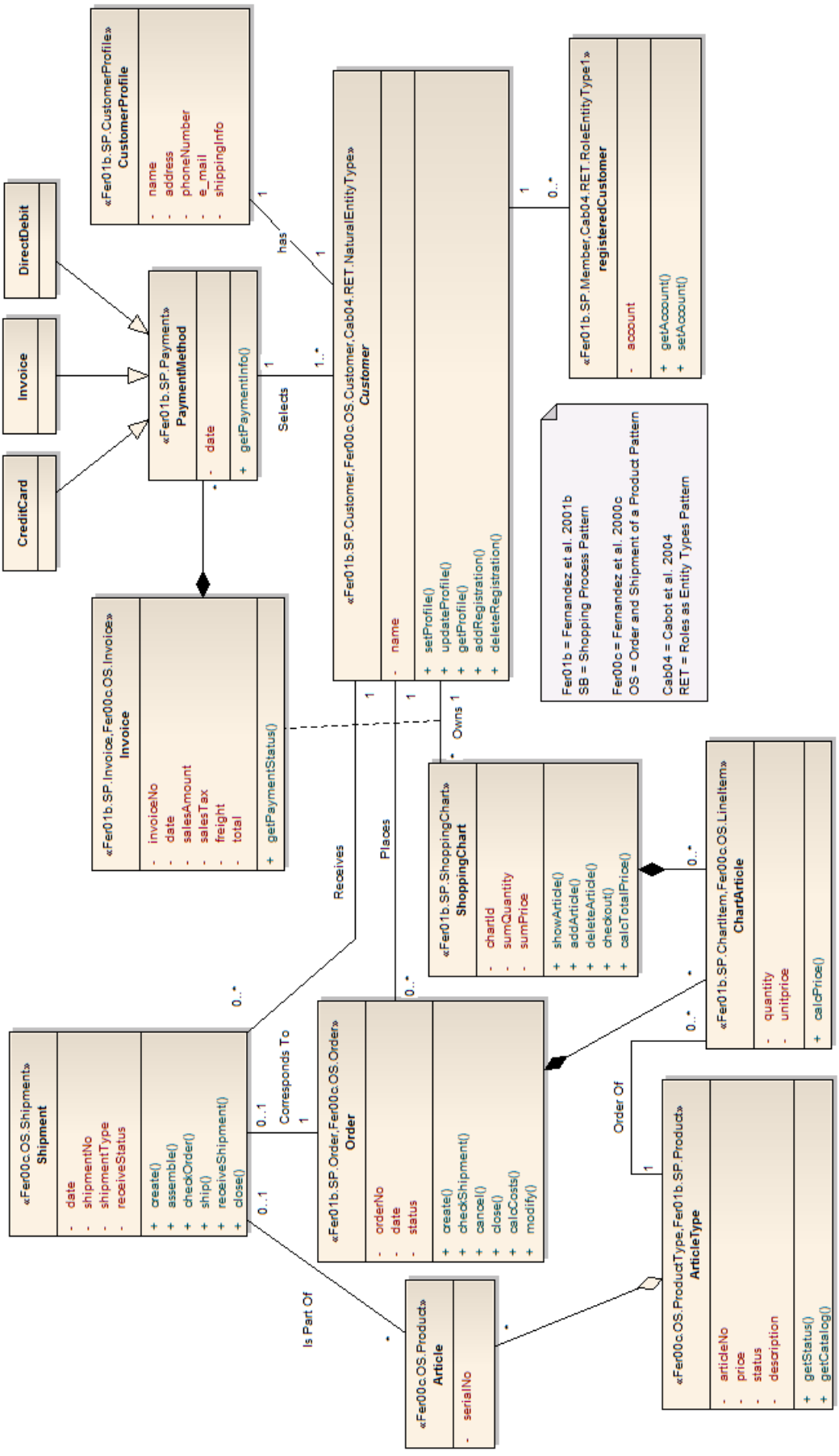
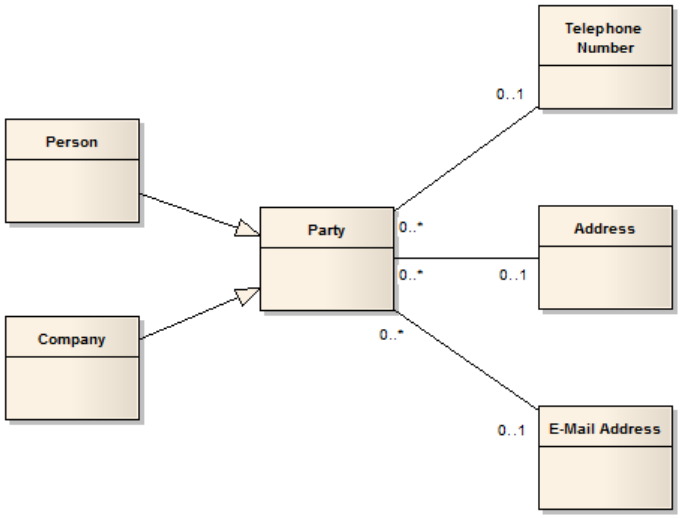


Fig. 4.23. Integration of the ‘Roles as Entity Type Pattern’ into the analysis model

4.3.3.6 Search for and Instantiation of a Fourth Analysis Pattern

The fourth pattern identified as matching is the ‘Party Pattern’ from FOWLER (1997).



Source: FOWLER 1997

Fig.4.24. Party Pattern

With the instantiation of the pattern (Fig. 4.25) the customer is sorted into ‘private person’ and ‘company.’ Since the customer is no longer to be used as an object, he is converted into an abstract class. The classes ‘Address’ etc. that are included in the party pattern have already been recorded via the ‘Customer Profile’ class, and can therefore be omitted.

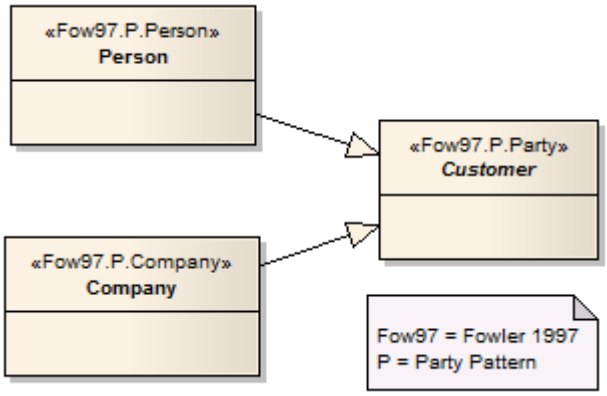


Fig. 4.25. Instantiated Party Pattern

4.3.3.7 Integration of the Fourth Pattern into the Model

Fig. 4.26 shows the integration of the instantiated party pattern into the analysis model.

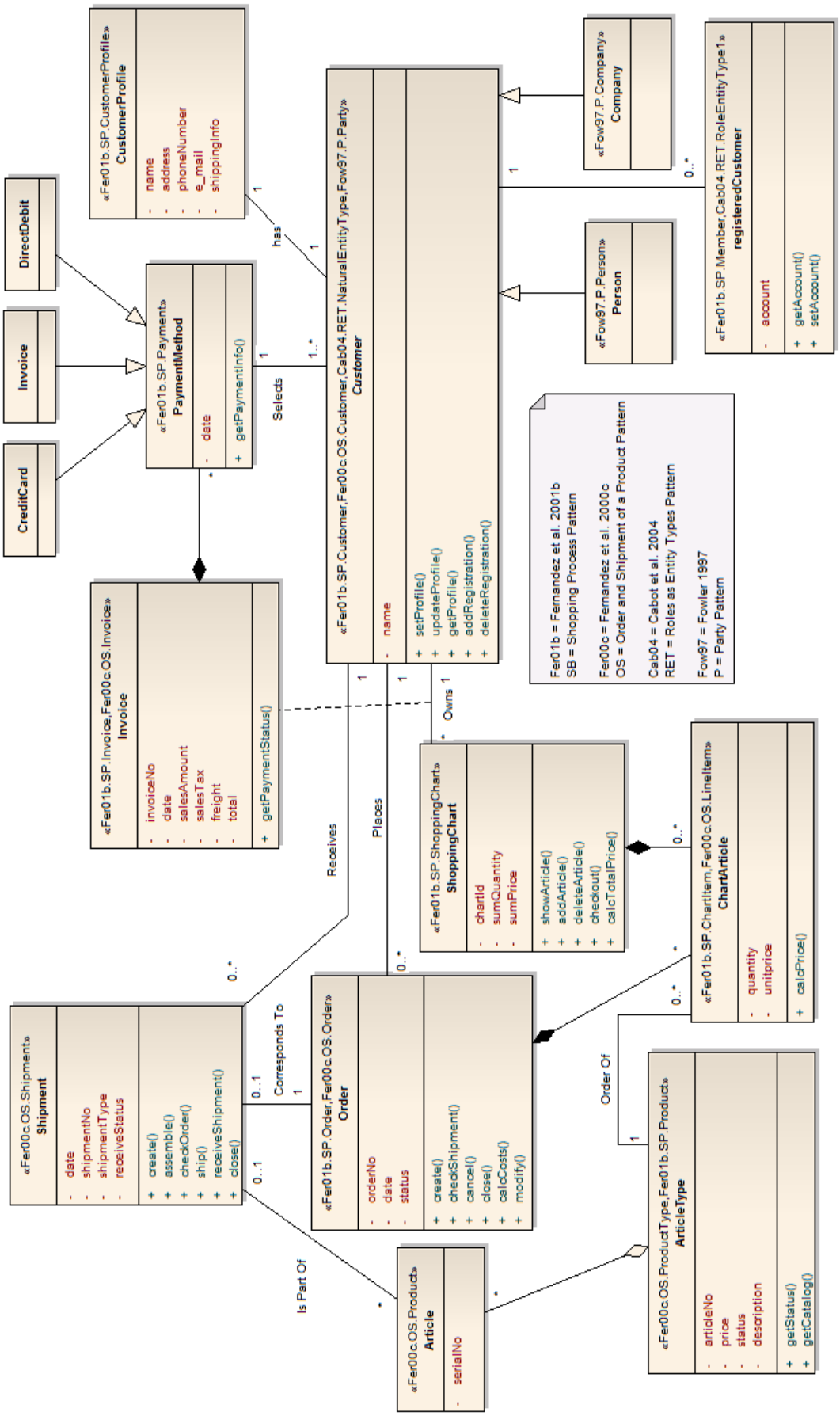


Fig. 4.26. Integration of the Party Pattern into the analysis model

4.3.3.8 Concluding Remarks on the Example

To conclude the example it should be noted that without the knowledge of the literature presented and, therefore, without the knowledge of existing analysis patterns, the creation of the analysis model using the PMM would have taken considerably longer. In this case, modelling by hand would probably have even been quicker.

The first selection criterion for the search for matching patterns for the example was the pattern name. Pattern names that did not contain sufficient information regarding the potential use of the pattern for the problem at hand might not have been considered at all. This is another example of how important it is to have a pattern catalogue with an intelligent search function.

4.4 Pros and Cons of Analysis Pattern Application

The previous chapters clearly demonstrated the advantages of using analysis patterns as opposed to conventional modelling. The following gives a brief summary of the advantages and disadvantages of analysis pattern application.

4.4.1 Pros

The re-use of analysis patterns can considerably reduce the time as well as the costs of the analysis phase, without diminishing the quality (HAMZA et al. 2003a). Time and costs are reduced by the direct use of the expert knowledge encapsulated in the analysis patterns, which saves developing the solution for each application from scratch (HAHSLER 2001, p. 1). In addition, the quality of the analysis is improved through the tried-and-tested solutions contained in the analysis patterns (FERNANDEZ and YUAN 2000b).

It is not only the analysis phase that is accelerated with the help of analysis patterns, but also the design and implementation, as conceptual errors also affect other phases of the development. A conceptual error cannot be rectified either in the design or in the code (FERNANDEZ and YUAN 2000b). According to BOLLOJU (2004), conceptual errors are a main reason for the failure of software projects, and therefore the improvement in quality of the conceptual model through the use of analysis patterns is a crucial factor in the further progress of the software development.

Analysis patterns can improve the communication within a software project. Patterns with unambiguous names ensure that the concept of a pattern can be addressed via its name. Thus developers can discuss problems more effectively, relatively independent of their level of experience (KONRAD et al. 2004, p. 971).

To maintain or change a conceptual model, in particular one that is not self-made, is, depending on the complexity, no easy task. The intention of the developer must first of all be understood and the starting-points for the change must be found. If the model was created with analysis patterns, with the help of the pattern documentation, it can be understood more easily and the starting-points for a change become more apparent. Ideally, new patterns can simply be added, or a new pattern can be exchanged for the one that represents the new requirements. This, however, requires that the patterns used were labelled when the model was created (see 4.2).

4.4.2 Cons

As made clear in chapter three, patterns must first be created and documented before they can be used. Ideally, they will then be filed in a pattern catalogue. This involves a considerable expense of time and money before the advantages of the analysis patterns can be profited from.

Even if a complete pattern catalogue that supports the developer with an intelligent search for the matching pattern is available, the selection of suitable patterns is still an expensive and time-consuming business, as in order to be evaluated and selected the patterns must first of all be understood.

Analysis patterns offer approaches to solutions for problems, but not complete solutions. In a normal case patterns cover the core of the problem, but then have to be expanded and changed in order to fully model the problem that is to be solved. If this fact is disregarded, and analysis patterns are misunderstood to be complete solutions, then incomplete models are the result.

5 Summary

This report attempts to establish and summarise the present state of literature on the subject of analysis patterns. The aim was to provide an overview of the existing literature that was as comprehensive as possible, thus granting the reader a more in-depth introductory guide to the field of analysis patterns, while identifying possible deficiencies which complicate the application and proliferation of analysis patterns.

It becomes evident from the literature overview that analysis patterns have been dealt with extensively in literature. There are a considerable number of analysis patterns that have been published either in books, or in scientific publications. Although their number is not comparable with that of design patterns (see HENNINGER and CORREA 2007), it can be concluded that this is due to the fact that analysis patterns belong to the problem space. The problem space is only dealt with by a small, specialised circle of people (anyone can code; not everyone can analyse), who, furthermore, work in different domains. In contrast, design patterns are not domain-dependent in the solution space, and they are closer to the code than analysis patterns. Thus, almost every programmer may benefit from design patterns, which obviously boosts both their dissemination and consistent enhancement.

The main factor preventing the dissemination of analysis patterns was identified as their lack of accessibility. Reduced or non-existent organisation and incomplete documentation make the application of analysis patterns more difficult. A considerable part of the time spent on this report was devoted to the search for relevant literature. Even now, the analysis patterns mentioned in literature are not yet organised and documented in a unified pattern template, which would simplify their application. If the systems analyst of a company has the choice of whether to build his model with the help of analysis patterns – which he (or she) first has to search for in the internet, without being sure that a matching pattern does even exist, or that he simply has not found it yet – or whether he should rather build his model from scratch, he will probably decide to save time and choose the latter.

The widespread assumption that the state of organisation and documentation of design patterns is much better is disproved by the study from HENNINGER and CORREA (2007). Owing to the sheer disorganised mass of design patterns, the solution of this problem is rather more difficult than it is with analysis patterns.

At the end of chapter three a possible form of pattern organisation was presented, proposing a Wiki with a comprehensive template. The implementation of this proposal with analysis patterns that already exist would be an important step towards allowing their use among developers in a commercial environment; the provision of a catalogue alone, however, is not sufficient: the catalogue would have to be adequately promoted e.g. at conferences, linked via the OMG web-site etc.

Overall, apart from setting up a complete pattern catalogue, another aim should be the active increase of the general awareness of analysis patterns and their advantages. A first step could be to raise the Wikipedia entries for analysis patterns to an informative level, one which will stimulate further use, with the results and the list of sources from this report; this should be carried out in German as well as, in particular, in English – the language of IT (BALZERT 2005 p. XI).

A comment to round off. The choice of language has a decisive influence on all types of measures and publications. Everything that is not written in English cuts itself off from the largest section of the international target group; and the concept of analysis patterns can only prevail on an international level.

Bibliography

AKAEM: Arbeitskreise Architektur- und Entwurfs-Muster, Begriffsdefinition. http://www.architekturmuster.de/index.php/Begriffsdefinition, 2008 (last visit 12/03/08)

Alexander, C., Ishikawa, S., Silverstein, M.: A Pattern Language. Towns, Buildings, Construction, Oxford University Press, 1977.

Andreu, E., Béjar, R., Latre, M. A., Martínez, S., Muro-Medrano, P. R.: Pattern-Based Approach to Support Automatic Homogeneous Map Labeling with Texts, Charts and Other Elements in a WMS. AGILE 2006, 9th AGILE Conference on Geographic Information Science: Shaping the future of Geographic Information Science in Europe, 2006.

Arias, M., Manjarrés, A., Díez, F. J., Pickin, S.: Construction of a Development Environment for GPMs Based on OO Analysis Patterns. Proceedings of KES 2003, 2003.

Arlow, J. and Neustadt, I.: Enterprise Patterns and MDA – Building Better Software with Archetype Patterns and UML. Pearson Education, Boston, Massachussets, USA 2004.

Auer M.: A Software Metric Pattern Dialect. Proceedings of VikingPloP 2002, 2002.

Balzert, H.: Lehrbuch der Objektmodellierung – Analyse und Entwurf. 2. Auflage, Spektrum Akademischer Verlag, Heidelberg 2005.

Balzert, H.: Lehrbuch der Objektmodellierung – Analyse und Entwurf. Spektrum Akademischer Verlag, Heidelberg 1999.

Bender, K.: Analysemuster in der Architektur kommerzieller Informationssysteme. Josef Eul Verlag, Lohmar 1999.

Bolloju, N. : Improving the quality of business object models using collaboration patterns. Communications of the ACM, Volume 47, No. 7, ACM-Press, USA, New York 2004.

Bortfeldt, A.: Objektorientierte Systemanalyse. Kurs-Nr. 818, FernUniversität in Hagen, Hagen 2001.

Braga, R. T. V., Germano, F. S. R., Masiero, P. C.: A Confederation of Patterns for Resource Management. Proceedings of PLoP 1998, 1998.

Braga, R. T. V., Germano, F. S. R., Masiero, P. C.: A Pattern Language for Business Resource Management. Proceedings of PLoP 1999, 1999.

Braga, R. T. V., Ré, R., Masiero, P. C.: A Process to Create Analysis Pattern Languages for Specific Domains. Proceedings of SugarLoafPLoP 2007, 2007.

Brown, W.J., Malveau, R.C., McCormick, H.W.S., Mowbray, T.J.: AntiPatterns: Refactoring Software, Architectures, and Projects in Crisis. J. Wiley, 1998.

Buschmann, F., Meunier, R., Rohnert, H., Sommerlad, P., Stal, M.: Pattern-Oriented Software Architecture, A System of Patterns. John Wiley & Sons Ltd, Chichester, England, 1996.

Cabot, J., Raventós, R.: Roles as Entity Types: A Conceptual Modelling Pattern. ER 2004, Lecture Notes in Computer Science Volume 3288, pp. 69-82, Springer-Verlag Berlin Heidelberg, 2004.

Chen, Y., Hamza, H. S., Fayad, M. E.: A Framework for Developing Design Models with Analysis and Design Patterns. IEEE International Conf. on IRI 2005, 2005.

Coad, P., Mayfield, M., North, D.: Object Models – Strategies, Patterns & Applications. Prentice Hall Publishing, USA, North Carolina, Rayleigh 1995.

Coad, P.: Object-Oriented Patterns. Communications of the ACM, Vol. 35, No. 9, ACM-Press, USA, New York 1992.

de Freitas Sodré, V., Lisboa, J. F., Vilela, V. M., Andrad, M. V. A.: Improving productivity and quality of GIS databases design using an analysis pattern catalog. Proceedings of the 2nd Asia-Pacific conference on Conceptual modelling - Volume 43; ACM International Conference Proceeding Series, Vol. 107, Australian Computer Society, Inc., Australia, Darlinghurst 2005.

de Nó, D. C.: Muster in der objektorientierten Analyse. Diplomarbeit, FernUniversität in Hagen, Hagen 2005.

De Rore, L., Snoeck, M.: An analysis pattern: 'Three-party pattern'. Proceedings of EuroPLoP 2005, 2005.

Devedzic, V., Harrer, A.: Software Patterns in ITS Architectures. International Journal of Artificial Intelligence in Education (IJAIED), Vol.15, No.2, 2005.

Fayad, M. E., Altman, A.: Introduction to Software Stability. Communications of the ACM,Vol. 44, No. 9, September 2001.

Fayad, M. E., Cangiano, G. R., Sánchez, H. A.: The Automation Analysis Pattern. Proceedings of the 1st Workshop on Stable Analysis Patterns, in conjunction with the 6th ACM/IEEE International Conference on the Unified Modeling Language (UML), 2003b.

Fayad, M. E., Das, S.: The Classification Stable Analysis Pattern. IEEE International Conf. on IRI 2007, 2007b.

Fayad, M. E., Das, S.: The Visualization Stable Analysis Pattern. IEEE International Conf. on IRI 2007, 2007a.

Fayad, M. E., Hamza, H. S., Stanton, V.: The Searching Analysis Pattern. Proceedings of SugarLoafPLoP 2004, 2004a.

Fayad, M. E., Hamza, H. S.: The AnyAccount Pattern. Proceedings of PLoP 2003, 2003a.

Fayad, M. E., Hamza, H. S.: The Trust Analysis Pattern. Proceedings of SugarLoafPLoP 2004, 2004b.

Fayad, M. E., Pradeep, R. S.: Evaluation Analysis Pattern. Proceedings of the 1st Workshop on Stable Analysis Patterns, in conjunction with the 6th ACM/IEEE International Conference on the Unified Modeling Language (UML), 2003d.
Fayad, M. E., Rajagopalan, J., Ranganath, A.: The Accessibility Analysis Pattern. Proceedings of the 1st Workshop on Stable Analysis Patterns, in conjunction with the 6th ACM/IEEE International Conference on the Unified Modeling Language (UML), 2003c.
Fayad, M. E., Telu, S.: The Learning Stable Analysis Pattern. IEEE International Conf. on IRI 2005, 2005.
Fernandez, E. B. and Yuan, X.: An Analysis Pattern for Reservation and Use of Reusable Entities. Proceedings of PLoP 1999. 1999.
Fernandez, E. B. and Yuan, X.: Securing Analysis Patterns. Proceedings of the 45th annual southeast regional conference ACM-SE 45, 2007.
Fernandez, E. B. and Yuan, X.: Semantic Analysis Patterns. 19th International Conference on Conceptual Modeling, ER2000, USA, Utah, Salt Lake City, 2000b.
Fernandez, E. B., Dai Fei: An Analysis Pattern for the Request and Allocation of Limited Resources. Proceedings of EuroPLoP 2002, 2002c.
Fernandez, E. B., Liu, Y., RouYi Pan: Patterns for Internet shops. Proceedings of PLoP 2001, 2001b.
Fernandez, E. B., Liu, Y.: The Account Analysis Pattern. Proceedings of EuroPLoP 2002, 2002b.
Fernandez, E. B., Pan, R.: The Sports Manager Pattern. Proceedings of PLoP 2002, 2002a.
Fernandez, E. B., Yuan, X., and Brey, S.: Analysis Patterns for the Order and Shipment of a Product. Proceedings of PLoP 2000, 2000c.
Fernandez, E. B., Yuan, X.: An Analysis Pattern for the Repair of an Entity. Proceedings of PLoP 2001, 2001a.
Fernandez, E. B.: Building Systems Using Analysis Patterns. Proceedings of third International Software Architecture Workshop (ISAW3), USA, Orlando, Florida 1998.
Fernandez, E. B.: Stock Manager: An Analysis Pattern for Inventories. Proceedings of PLoP 2000, 2000.
Ferreira, M. J., Loucopoulos, P.: Business analysis patterns: methodological and support environment aspects. 9th Americas Conference on Information Systems (AMCIS 2003), 2003.
Ferreira, M. J., Loucopoulos, P.: Organisation of analysis patterns for effective re-use. Proceedings of 3rd International Conference on Enterprise Information Systems (ICEIS 2001), 2001.
Fowler, M.: Analysis Patterns – Reusable Object Models. Addison-Wesley Publishing, 1997.
Fowler, M.: Analysis Patterns 2 – Work in Progress. http://www.martinfowler.com/ap2/index.html, 2007 (last visit 1/12/07)
Fülleborn, A., Heisel, M.: Methods to Create and Use Cross-Domain Analysis Patterns. Proceedings of EuroPLoP 2006, 2006.
Gaertner, N., Thirion, B.: Grafcet: An Analysis Pattern for Event Driven Real-time Systems. Proceedings of PLoP 1999, 1999.
Gamma, E., Helm, R., Johnson, R. and Vlissides, J.: Design Patterns – Elements of Reusable Object-Oriented Software. Addison-Wesley Publishing, Mountain View, California, USA 1995.
Geyer-Schulz, A., Hahsler, M.: Software Engineering with Analysis Patterns. Technical Report 01/2001, Institut für Informationsverarbeitung und -wirtschaft, Österreich, Wien 2001.
Geyer-Schulz, A., Hahsler, M.: Software Reuse with Analysis Patterns. Proceedings of AMCIS 2002, 2002.
Gyani, J., Murthi, P. R. K.: A Pattern Language for Online Share Trading. Proceedings of EuroPLoP 2005, 2005.
Hahsler, M.: Analyse Patterns im Softwareentwicklungsprozeß – Mit Beispielen für Informationsmanagement und deren Anwendungen für die Virtuelle Universität der Wirtschaftsuniversität Wien. Dissertation, Wirtschaftsuniversität Wien, Österreich 2001.
Hamza, H. S., Fayad, M. E.: A Pattern Language for Building Stable Analysis Patterns. Proceedings of PLoP 2002, 2002c.
Hamza, H. S., Fayad, M.E.: Model-based Software Reuse Using Stable Analysis Patterns. Proceedings of ECOOP 2002, 2002b.
Hamza, H. S., Chen, Y.: PAD: A Pattern-Driven Analysis and Design Method. OOPSLA 2005, 2005.
Hamza, H. S., Fayad, M. E.: Applying Analysis Patterns through Analogy: Problems and Solutions. Journal of Object Technology, Vol.3, No. 4, Schweiz, Zürich April 2004a.
Hamza, H. S., Fayad, M. E.: Stable Analysis Patterns. IEEE International Conference on Computer Systems and Applications 2006, 2006.
Hamza, H. S., Fayad, M. E.: The Negotiation Analysis Pattern. Proceedings of PLoP 2004, 2004.
Hamza, H. S., Mahdy, A., Fayad, M. E., Cline, M.: Extracting Domain-Specific and Domain-Neutral Patterns Using Software Stability Concepts. OOIS 2003, 2003b
Hamza, H. S., Mahdy, A., Fayad, M. E.: Towards A Framework For Analysis Patterns Evaluation. Proceedings of the 1st Workshop on Stable Analysis Patterns, in conjunction with the 6th ACM/IEEE International Conference on the Unified Modeling Language (UML), 2003a

Hamza, H. S.: A Foundation for Building Stable Analysis Patterns. Master Thesis, University of Nebraska, USA, Nebraska, Lincoln, 2002a.
Hamza, H. S.: Improving Analysis Patterns Reuse: An Ontological Approach.
Hamza, H. S.: On the integration of stable analysis patterns with traditional patterns. Proceedings of JIISIC 2004, 2004b.
Hamza, H. S.: Towards stable software analysis patterns. Proceedings of OOPSLA 2002, 2002d.
Harrer, A., Devedzic, V.: Design and Analysis Patterns in ITS Architectures. Proceedings of International Conference on Computers in Education 2002. IEEE Computer Society, Los Alamitos, CA, 2002.
Hay, D. C.: Data Model Patterns: Convention of Thoughts. Dorset House Publishing, New York, USA 1995.
Henninger, S., Correa, V.: Software Pattern Communities: Current Practices and Challenges. Proceedings of PLoP 2007, 2007.
Jacobsen, E. E., Kristensen, B. B., Nowack, P.: Patterns in the analysis, design and implementation of frameworks. Proceedings of COMPSAC 1997, 1997.
Johannesson, P. and Wohed, P.: Deontic Analysis Patterns. Proceedings of the 3rd International Workshop on the Language Action Perspective on Communication Modelling - LAP98, 1998.
Kemp, R.: Anwendung von Mustern in der objektorientierten Analyse. Diplomarbeit, FernUniversität in Hagen, Hagen 2006.
Konrad, S., Cheng, B. H.C., Campbell, L. A.: Object Analysis Patterns for Embedded Systems. IEEE Transactions on Software Engineering, 2004.
Lisboa F., J., de Freitas Sodré, J., Daltio, J., Rodrigues Júnior, M. F., Vilela, V.: A CASE Tool for Geographic Database Design Supporting Analysis Patterns. ER Workshops 2004, Lecture Notes in Computer Science Volume 3289, pp. 43–54, Springer-Verlag Berlin Heidelberg, 2004.
Lisboa F., J., Iochpe, C., Beard, K.: Applying Analysis Patterns in the GIS Domain. Proceedings of SIRC 1998, 1998.
Lisboa F., J., Iochpe, C., Borges. K. A. V.: Analysis Patterns for GIS Data Schema Reuse on Urban Management Applications. CLEI Electronic Journal Volume 5 Number 2, 2002.
Lisboa F., J., Iochpe, C.: Specifying analysis patterns for geographic databases on the basis of a conceptual framework. Proceedings of ACM GIS 1999, 1999.
Manjarrés, A., Sunyé, G., Pollet, D., Pickin, S., Jézéquel, J. M.: AI Analysis Patterns as UML Meta-Model Constructs. Proceedings of the 14th international conference on software engineering and knowledge engineering (SEKE) 2002, 2002.
Mossé, F. G.: Modeling Roles - A Practical Series of Analysis Patterns. Journal of Object Technology, Volume 1, Number 4, 2002.
Nicola, J., Mayfield, M. and Abney, M.: Streamlined Object Modeling – Patterns, Rules and Implementation. Prentice Hall Publishing, USA 2002.
Pantoquilho, M., Raminhos, R., Araújo, J., Moreira, A.: A Systematic Analysis Patterns Specification. Proceedings of IECS 2006, 2006.
Pantoquilho, M., Raminhos, R., Araújo, J.: Analysis Patterns Specifications: Filling the Gaps. Proceedings of VikingPLoP 2003, 2003.
Pickin, S., Manjarrés, A.: Describing AI Analysis Patterns with UML. Proceedings of UML 2000, 2000.
Pleß, V., Pankratz, G., Bortfeldt, A.: The Rate Structure Pattern: An Analysis Pattern for the Flexible Parameterization of Charges, Fees and Prices. Diskussionsbeitrag des Fachbereichs Wirtschaftswissenschaft der FernUniversität in Hagen, Nr. 386, Hagen 2006.
Purao, S., Storey, V. C. and Han, T.: Improving Analysis Pattern Reuse in Conceptual Design: Augmenting Automated Processes with Supervised Learning. Information Systems Research, Vol. 14 , No. 3, Institute for Operations Research and the Management Sciences (INFORMS), USA 2003.
Ré, R., Braga, R. T. V., Masiero, P. C.: A Pattern Language for Online Auctions Management. Proceedings of PLoP 2001, 2001.
Riehle, D., Züllighoven, H.: Understanding and Using Patterns in Software Development. Theory and Practice of Object Systems 2, 1, 1996.
Robertson, S., Strunch, K.: Reusing the Products of Analysis. Second International Workshop on Software Reusability, 1993.
Rupp, C., Dallner, J.: Mustergültige Anforderungen. Objektspektrum 3/2001.
Sánchez, H. A., Binbin Lai, Fayad, M. E.: The Sampling Analysis Pattern, IEEE International Conf. on IRI 2003, 2003.
Seruca, I., Loucopoulos, P.: Towards a systematic approach to the capture of patterns within a business domain. Journal of Systems and Software, vol.67, issue 1, 2002.
Sesera, L.: A Recurring Fulfilments Analysis Pattern. Proceedings of PLoP 2000, 2000b.
Sesera, L.: Analysis Patterns. Proceedings of SOFSEM 2000, 2000a.
Sesera, L.: Hierarchical Patterns: A Way to Organize (Analysis) Patterns. Journal of Systemics, Cybernetics and Informatics, Volume 3 Number 4 2004.

Sesera, L.: Obligation-Fulfillment: A Pattern Language for Some Financial Information Systems. Proceedings of EuroPLoP 2005, 2005.
Sorgente, T., Fernandez, E. B., Larrondo Petrie, M. M.: Analysis Patterns for Patient Treatment. Proceedings of PLoP 2004, 2004.
Sorgente, T., Fernandez, E. B., Larrondo Petrie, M. M.: The SOAP Pattern for Medical Charts. Proceedings of PLoP 2005, 2005.
Ubezio, L., Raibulet, C., Carpinato, A.: Novel Analysis Patterns in the Context of the Managed Investments Instruments. ADBIS Research Communications, 2006.
UML: OMG Unified Modeling Language, Superstructure. Object Management Group Inc., Version 2.1.2, http://www.omg.org/docs/formal/07-11-02.pdf, 2007. (last visit 2/03/08)
Wikipedia: Wiki. http://de.wikipedia.org/wiki/Wiki, 2008a (last visit 26/02/08)
Wohed, P.: Conceptual Patterns – a Consolidation of Coad's and Wohed's Approaches. Proceedings of the 5th International Conference on Applications of Natural Language to Information Systems, NLDB 2000, Springer, 2000b.
Wohed, P.: Conceptual Patterns for Reuse in Information Systems Analysis. Proceedings of the 12th International Conference on Advanced Information Systems, CAiSE 2000, Springer, 2000a.
Wohed, P.: Tool Support for Reuse of Analysis Patterns – a Case Study. Proceedings of the 19th Int. Conference on Conceptual Modeling, ER 2000, Springer, 2000c.
Yuan, X., Fernandez, E. B.: An Analysis Pattern for Course Management. Proceedings of EuroPLoP 2003, 2003.
Zhen, L., Shao, G.: Analysis patterns for oil refineries. Proceedings of PLoP 2002, 2003.
Zhen, L.: Analysis Patterns for Materials and Products of Refineries. Proceedings of PLoP 2003, 2003.