Association for Information Systems
AIS Electronic Library (AISeL)

AMCIS 2003 Proceedings

Americas Conference on Information Systems (AMCIS)

December 2003

Business Analysis Patterns: Methodological and Support Environment Aspects

Maria Ferreira
Universidade Portucalense

Pericles Loucopoulos
University of Manchester Institute of Science and Technology

BUSINESS ANALYSIS PATTERNS: METHODOLOGICAL AND SUPPORT ENVIRONMENT ASPECTS

Maria João Ferreira
Universidade Portucalense
mjoao@uportu.pt

Pericles Loucopoulos
University of Manchester Institute of Science and Technology
pl@co.umist.ac.uk

Abstract

In software engineering, particularly in requirements engineering, saving time and improving the quality of products through reuse is possible. Experienced developers reuse their own ideas by recognising problems that are similar to other problems that they have solved before. They then apply the best solution they had used previously. Patterns crop up from reusing a recurring problem/solution. In the late 1980's, patterns were introduced as a concept in computer science and they provided a way to save (preserve) and transmit knowledge and experience. This paper describes a prototype tool (RIBAP) that supports working with business analysis patterns when developing or maintaining a system. Aspects of a methodological and support environment are also discussed.

Keywords: Requirements engineering reuse, reuse of business analysis patterns, reuse process

Introduction

The success of the development process of a software system in software engineering critically depends on the designer's understanding of the problem. This understanding is condensed in requirements specification, which should unambiguously, completely and consistently (IEEE, 1984; Holbrook, 1990) describe requirements. Formal verification of software is always based on specifications. It is therefore tied to the quality of specifications. Indeed, specifications act as a trigger (Rolland and Prakash, 1999) and a channel of communication throughout the software development process. The process that ultimately leads to these specifications is called Requirements Engineering (RE). Consequently, requirements engineering (RE) is a branch of software engineering (Zave, 1997) and it has been traditionally looked upon as the "front-end" of the software life cycle (Loucopoulos and Karakostas, 1995; Filippidou, 1997). Its main goal is to produce specifications from informal ideas for the problem that is to be solved (Kotonya and Sommerville, 1996).

RE products are the foundations of total subsequent systems development. Their success depends on the quality of the requirements engineering process. In the software development community, it is generally agreed that RE is one of the most important (Pohl, 1996) and critical activities in the software life cycle (Kramer and Ng, 1988; Hsia et al. 1994). Consequently, the main reason for spending time and efforts on the RE process and its improvement arises from the goal to develop right from the start as opposed to repairing at the end. Errors occurring in the RE stage, generate a large number of errors in other stages and the cost of repairing such errors is extremely high (Davis, 1993). Although the importance of good and well defined RE processes is generally accepted and RE has been established as a distinct field of research and practice (TSE, 1997), a lot of progress has been achieved. Nevertheless, RE is still quite an undeveloped discipline (Berry and Lawrence, 1998), and systems are delivered in poor quality, too late, over budget and often do not do what users really want them to do. In some extreme cases, they become useless (Kotonya and Sommerville, 1996) because of the poor processes of requirements engineering (Kotonya and Sommervile, 1997).

The RE process therefore requires improvement (William and Kennedy, 1999). This situation raises the question of reuse in requirements engineering (Loucopoulos and Karakostas, 1995; Bellinzona et al. 1995; Knight and Kienzle, 1992; Grosz et al. 1999) to reduce development lead-time and at the same time increase software quality to acquire domain knowledge. In addition

to the intensive labour of requirements engineering, constant technological and economical changes require organisations to be capable of modifying and continuously extending their traditional practices. On the one hand, this leads to the demand for increasing complexity and constant evolution of information systems in organisations (Prekas et al. 1999), forcing software products to become more complex and to have shorter marketing times. On the other hand, the capacity of the software industry to deliver these in a timely and cost effective manner does not satisfy the organisation's needs (Ye et al. 2000). Reuse, a development approach using existing reusable artefacts to create new software systems or to update those in existence, crops up as a solution to tackle these problems (Staringer, 1994).

Development of software systems based on reuse is not new. Rather, it has been around since the beginning of computation in a non-formalised form. In other words, this helps any developer who tries to use his previous experience or past solutions informally in a present situation. Reuse may be introduced at any stage of software development (Bellinzona, 1995; Banker et al. 1993). In particular, reuse in requirements specification is a way to decrease efforts in specification development and to increase specification quality (Knight and Kienzle, 1992). More than 50 percent of requirements (Kotonya and Sommerville, 1997) that could be reused are represented by requirements/specifications related to constraints within a domain, to styles of data presentation or to policies of the organisation. Furthermore, reuse improves the designer's efficiency in requirements elicitation and validation through its assistance and guidance (Rolland and Prakash, 1999).

Despite its potential benefits, reuse is difficult [Glass, 1998], particularly with respect to requirements specification since it requires communication of complex specifications from the "expert". These must also be learnt either totally or partially by the re-user of the specification. Due to the nature of requirements specification, the respective essentials are not easy to describe and graphical models that are produced often lack or contain insufficient documentation.

An approach to improve this idea concerns reuse by using business analysis patterns. Patterns (Alexander et al. 1977) are a literal form and they have been used by some researchers as a means to capture, document and communicate the *best practice* to a given problem. Typically, a pattern has a descriptive name, a problem description for the pattern applicability, an analysis of the forces that the pattern addresses, a solution, the rational behind the solution, the solution usages, and a list of related patterns. The solution is presented in a specification model provided either from formal or informal requirements. Entities and their relationships are defined in business analysis patterns; patterns also link and structure domain concepts and aggregate entities that form higher abstractions.

Since the introduction of patterns into Computer Science, a large number of pattern libraries for different domains have been developed. In most cases, these patterns are represented in a 'flat' fashion. This makes their use difficult especially when there are a large number of patterns to be considered in a particular application. For this reason, a framework for business analysis patterns reuse is proposed. This framework will help a designer to locate and select patterns in an automated environment and will solve the problem he has in hand.

Section 2 of this paper focuses on the problems and challenges that are faced when business analysis patterns are reused. In section 3, a framework to reuse business analysis patterns is proposed from both the designer's and the reuse engineer's perspectives. Section 4 describes a design of the prototyping tool. The implementation of the respective tool is described in section 5. Section 6 handles the main conclusions of this work.

Problems and Challenges

Business analysis patterns are "groups of concepts that represent a common construction in business modelling …" (Fowler, 1997). Designers may re-use and incorporate business analysis patterns into their work. By doing so, they will reduce the time and the resources consumed in requirements engineering. Nevertheless, there are problems associated with the reuse of business analysis patterns that must be solved. In the following subsections, these problems are identified and discussed.

Reuse Process

It is widely accepted that reuse is an approach that can lead to substantial productivity gains, improvements to quality and reliability in software development. Effective reuse involves the following activities: (1) *Retrieval* - specifying a query and localising potential candidates against a repository; (2) *Evaluation* - determining the relationship between a retrieved candidate

and the specification of the desired business analysis pattern; and (3) *Revision* – changing the selected business analysis pattern to obtain the desired solution.

In an ideal scenario, a reuse tool provides automated assistance to all these activities. The higher the retrieval performance, the lower the reuse efforts will be during evaluation with respect to knowing which of the retrieved business analysis patterns best satisfy the desired specification and what adjustments must be carried out on the selected business analysis pattern.

Nevertheless, automation can be difficult since each activity requires different types of information regarding a business analysis pattern, namely: (1) *Retrieval* benefits from an abstract classification that supports an efficient comparison between potential candidates; (2) *Evaluation* requires information that allows the relationship between the retrieved candidate and the desired business analysis pattern to be established; and finally, (3) *Revision* requires information about the structure of the selected business analysis pattern and its relationship with other business analysis patterns.

Patterns Organisation

Two main types of pattern collections have been identified in the literature, namely, pattern catalogues and patterns languages. These two types of pattern collections vary in structure and interaction among patterns. This section revises the organisation of patterns to establish the main principles regarding the organisation of patterns and their reuse. This is done without resorting to automated support (i.e. manually).

Pattern Catalogues

Isolated patterns may not be very useful since they only cover a single, small problem. In order to cover many problems in software development, patterns are compiled into catalogues. These catalogues typically subdivide the patterns into at least a small number of broad categories. Patterns in a pattern catalogue may enjoy some kind of organisation provided by the catalogue author. Nevertheless, the real problem of organisation is not addressed. For example, patterns in the ELEKTRA project (Loucopoulos, 1997), are inserted into a catalogue organised by area. As a result, the surplus value of a pattern catalogue subsists in the quality of the patterns themselves rather than in their organisation.

Use (reuse) of a pattern catalogue requires, firstly, understanding the classification (organisation) established by the author of the catalogue, and secondly, locating relevant patterns in the catalogue.

Pattern Language

A pattern language is a collection of interrelated patterns organised in a coherent schema that solves a problem at a large-scale. Pattern organisation in a pattern language is established by the relationships between patterns. Briefly, a pattern language is a directed attribute graph with nodes describing patterns and edges representing the semantic relationships among them. A pattern language combines the nodes into an organisational framework. Relationships between patterns, namely, the *rules*, are as important as the patterns themselves since they establish the grammar of the language. As in linguistics, where words cannot build a language without connection rules, patterns cannot form a language without using any rules. A language establishes the rules to combine patterns. It indicates which patterns may be combined and how they must be combined in order to obtain a higher-level pattern, i.e. a higher-level solution. Building a system with a pattern language is similar to creating sentences, paragraphs and books in natural language (Alexander et al. 1977). A pattern language is a progression from higher-scale to lower-scale problems, where each pattern depends on the smaller patterns it contains (Coplien and Schmidt 1995).

A pattern language is much more than a pattern catalogue. In a pattern language, an order is established for patterns. There is a well-defined organisation defining which patterns may be accessed and in what ways this may be done in order to achieve a higher-level solution. Organisation is intrinsic to the language. A catalogue is the most common way of finding collected patterns. It is similar to a "*deposit of small pieces*" where the problem of organisation is neglected. The definition of an organisation or the classification for effective location and selection of patterns in an automated environment is therefore fundamental.

Business Aanalysis Patterns Reuse Framework

The purpose of creating the framework is to provide a systematic, searchable and retrievable way of managing a repository for business analysis patterns that will assist the reuse process. This framework is used as the basis to develop a software tool, namely, Reuse Infrastructure Based on Analysis Patterns (RIBAP).

Business Analysis Patterns Definition

Template of Business Analysis Patterns

Conceptual reuse models have not been the core focus of most efforts put into artefacts reuse. Automation of the reuse process requires an understanding of the aspects that have not been incorporated in a conceptual model.

A conceptual model provides a description of how a model performs its solution. For evaluation and revision purposes, we are interested in determining precisely what a conceptual model is like. A conceptual model included in a business analysis pattern may help resolve this problem since a business analysis pattern incorporates the "what", "why" and "how" into its structure. These three basic attributes enrich and facilitate reuse automation. A business analysis pattern provides the expressiveness and precision required to capture the essence of a conceptual model. Consequently, a template for the business analysis pattern was created to accomplish two activities of the reuse process, namely, evaluation and revision. According to the template, a business analysis pattern is decomposed into three parts: (1) *the pattern descriptor* is an explicit lexical description of the situation whose pattern is relevant for reuse; (2) *the pattern justifier* is an explicit lexical description of the rationale behind the solution; (3) *the pattern solution* (body) corresponds to the conceptual model that is presented to the designer when it is required. Consequently, the descriptor must be used in the evaluation process. Furthermore, the justifier must be used as a support when revising business analysis pattern(s) that best fits the designer's requirements. The pattern solution allows the effective reuse of the business analysis pattern and it may be used either "as-is" or revised (adapted).

Figure 1 and figure 2 present an example of a business analysis pattern extracted from the ELEKTRA project pattern catalogue (Loucopoulos, 1997). It is decomposed into three parts in accordance with the template. The business analysis pattern template permits efficient reuse of analysis patterns. In the reuse process, the retrieval process retrieves analysis patterns satisfying the request and the pattern *descriptor* is presented to the designer. Only the pattern description is presented to the user since the goal is to provide the designer with the required information to know in what situation the analysis pattern will be relevant to reuse. Once an analysis pattern is selected, the pattern *justifier* is shown. The designer may then decide whether the solution is suitable or whether it must be adapted or rejected. If the solution is suitable "as-is" or must be adapted, then the analysis pattern *body* may be visualised.

Pattern Number: 2.2.3
Name: Competency Management Process Indicators
Problem domain: Competency Management Indicators
Problem description: To measure/estimate the excellence of an organisational unit in handling its Human Resources planning and management activities.
Context:

- The HR planning and management process must be described.
- Ability to identify which factors are relevant to measure
- Access to values for chosen relevant factors

Type: Product pattern.

Descriptor

Figure 1. An Example of an Analysis Patterns Descriptor

Structure description/Rationale:

- Just-in-time access to necessary competency is a critical success factor for ESI companies, which operate in a deregulated market. Just-in-time access to necessary competency is a critical success factor for ESI companies, which operate in a deregulated market.

Modulation principle/Solution:

- Identify relevant factors to measure the efficiency of Human Resources planning and management activities.

Consequences:

- The organisational unit will be able to improve its Human Resources planning and management activities.

Related patterns:

- Change process patterns:
  - BusinessAlignedCompetency
  - ...
- Product patterns:
  - EmployeeIndicatorsOrganisationalUnitLevel
  - LeadershipIndicators
  - ...

Justifier

Known Applications: Vattenfall AB, Sweden

Examples:

Tips (for implementation:

Competency planning and management activities in a company operating in a flexible and dynamic environment must be efficient. This goal is measured by using process indicators. Process indicators for an organisational unit are, for instance:

- the result from an AUDIT grading of the competency planning and management proess
- employee leadership attitude indicators
- employee competency development attitude indicators

Body

Figure 2. An Exmaple of an Analysis Patterns Descriptor and Analysis Patterns Descriptor

Relationships between Business Analysis Patterns

The real strength of business analysis patterns comes into its own when multiple business analysis patterns are used together since a single business analysis pattern only covers a single, small problem. When a business analysis pattern is chosen, other business analysis patterns that may possibly be combined with it must be found. Relationships between patterns should help this activity. Although authors document these relationships, they tend to use their own nomenclature. Some of the work attempts to categorise different types of relationships, such as, (Noble, 1997; Zimmer, 1994). The Zimmer approach defines three types of relationships: (1) uses; (2) combined; and (3) similar.

The following two subsections will discuss the relationships between patterns according to the context of pattern languages and pattern catalogues in order to propose a relationship classification scheme.

➢Pattern catalogue

Usually, pattern relationships in pattern descriptions are expressed in sections such as "related patterns" "see also" or they are incorporated into the pattern description. Although it is noticeable that different types of relationships exist between patterns, no standard types have been acknowledged by the pattern community.

A relationship classification scheme has been proposed based on Zimmer's approach as well as on business analysis patterns and their mutual relationships put forward by Fowler (Fowler, 1997) and the ELEKTRA project pattern catalogue (Loucopoulos, 1997). There are five primary relationships and two secondary relationships in the classification scheme. The primary relationships consist of the three relationships defined by the Zimmer classification approach and the two proposed by ourselves. The two secondary relationships are classified as secondary because they do not establish a determinant relationship with a problem. Table 1, presents a summary of the proposed classification.

Using this classification schema, a "standard language" was established for the "*related patterns*" and the "*see also*" section of a pattern. This classification may be used on existing patterns to sort out relationships between patterns in existing business analysis pattern catalogues and on business analysis patterns that will be created. If business analysis patterns use this classification in an automated environment, then the repository supporting these relationships may be more consistent since all the relationships between patterns are easily expressed and controlled. Consequently, patterns related to any specific pattern are easily found. A designer easily sees not only a single pattern but also all the patterns associated with it. Furthermore, he understands the kind of relationships found between them.

Table 1. Relationship Classification Schema

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

➢Pattern language

Organisation principles for pattern languages have been explored in the previous section. Briefly, patterns in a pattern language are organised from the most general, large-scale patterns to the most specific, small-scale patterns, based on the relationships between patterns. A pattern language may be used as a "how to" guide, i.e., systematically, starting from the larger, generic pattern to the more specific using paths created by the relationships.

For this reason, a pattern language may be seen as a graph where patterns are nodes and relationships are edges. The set of paths formed by relationships determine alternative solutions to the same problem. Based on relationships, each path defines the order of the patterns, i.e., which patterns are entry points, which patterns follow and which are leafs. Consequently, we propose using the "*Uses*" relationship defined by the Zimmer's classification approach. This relationship permits shifting from a large pattern to a smaller one.

Business Analysis Patterns Retrieval Approach

The main aim of the business analysis patterns retrieval approach is to create a repository of business analysis patterns that may be reused from one problem to another. Building and using a business analysis pattern repository implies using the definition of an analysis-pattern retrieval model (Ferreira and Loucopoulos, 2002).

Definition

An analysis-pattern retrieval model is a triple:

(P, Q, R)

Where:

(1) **P** is the set consisting of the representations of business analysis patterns in a collection.
(2) **Q** is a set consisting of the representations of user needs - queries.
(3) **R** consists of the retrieval mechanism

R: P * Q → Real (signature of the function) is a ranking function that returns the rank of a pattern p IN P for a query q in Q.

The Classification Scheme

We propose a classification scheme (Ferreira and Loucopoulos, 2001) to organise business analysis patterns into the RIBAP repository where these business analysis patterns are classified in a "*problem-oriented"* perspective. In other words, the classification captures the intrinsic characteristics of the problem expressed in a business analysis pattern. These characteristics are described by a set of facets. The semantic information available on the classification scheme used to describe business analysis patterns is Domain, DomainArea, Process, ProcessOn and Type facets.

Table 2. Pattern Classification Scheme

Concept	Facet Identification	Facet Concept
Context (the "environment" in which business analysis patterns operate)	F1 F2	Domain (noun) DomainArea (noun)
Concept (abstraction captured in a business analysis pattern)	F3 F4	Process (verb) ProcessOn (noun)
Content (the implementation of that abstraction)	F5	Type (noun)

The Classification Process

Faceted classification may achieve a highly precise classification (Jr, 2001) for each business analysis pattern depending on which terms are adopted. The classification process corresponds to the extraction of either relevant information or terms for each facet from a business analysis pattern. A list with pre-defined terms is used in the classification process as the basis for classifying new business analysis patterns. Terms are presented dynamically. For example, if a reuse engineer decides to classify a business analysis pattern as part of the `<Electricity>` domain, then the facet domain will accept the Electricity value. Subsequently, when the reuse engineer selects the next facet, namely, the DomainArea, the terms that belong to it are shown and the process is repeated for all facets. At the end of this process, a business analysis pattern is associated with a set of terms (from the term space) for each facet, i.e., each business analysis pattern may be associated with one or more terms for each facet. Facets used by our classification are mandatory, i.e., reuse engineers are only allowed to save business analysis patterns in the repository after having proceeding with a well-conformed classification for business analysis patterns. Data (pattern name, problem description…) related to a business analysis pattern must also be supplied since lack thereof prejudices the comprehension of the business analysis pattern, and consequently, the probability of it being reused. To solve this problem, there is an advisory system similar to the one used in the classification scheme to help the reuse engineer document a business analysis pattern well.

We recommend that a vector of terms be used to represent business analysis patterns with each one representing a facet, where each facet is respectively weighted in line with its importance. Accordingly, a Business Analysis Pattern *p* consists of a set of terms from the term space with the respective weights corresponding to the facets.

Query Representation

From a reuse context and the user's point of view, retrieval may be classified as either implicit or explicit in accordance with the type of query.

An implicit query - *Browsing* - is a special retrieval operation (Constantopoulos and Pataki, 1992). It is associated with the inspection of feasible candidates for possible extraction although there are no predefined criteria - navigation. Navigation determines the sequence in which business analysis patterns are visited.

Once the object of interest has been identified, the system permits the neighbours of the current object of interest to be viewed within a given relationship. In view of the fact that there is a structure of relationships between business analysis patterns in a repository, the neighbourhood of the current business analysis pattern is defined by adjacent and subjacent links to it. Accordingly, the Browse function accepts the current object (business analysis pattern), the list of related business analysis patterns and their relationship types as input. Furthermore, it determines a local view centred on the current business analysis pattern.

An explicit query aims at identifying candidates by executing a preliminary assessment based on the classification scheme. The information that must be used in the query formulation is the same as that used in the classification process. Selection of the terms will be based on the designer's desires. Accordingly, the data described, namely, the facets terms, will be specified by the designer i.e. by the person who best knows what the needs are. Nevertheless, leaving one facet with no value should always be possible. Consequently, a query allows the designer to express fuzzy predicates for proximity searches (e.g. <find all business analysis patterns to manage an account>), content-based predicates (e.g. <find all business analysis patterns on accounting>) and structural predicates (e.g. <find all business analysis patterns containing a class diagram>).

We suggest that Boolean queries be used. Accordingly, the query language consists of the set T of facet terms from the faceted thesaurus and the operators <AND>, <OR> and <NOT>. The operator <AND> narrows the search by only retrieving those business analysis patterns that contain all the facet terms in the query. The operator <OR> expands the search by spewing out business analysis patterns with either one or both of the facet terms contained in the classification of the pattern. Consequently, the result is enlarged. The operator <NOT> limits the search to business analysis patterns that do not include any of the facet terms. This operator is used to exclude business analysis patterns that have been classified with the facet term(s) following the operator <NOT>. While <AND> may be applied to terms from different facets, <OR> and <NOT> may only be applied to terms within single facets. This is not a real limitation since users normally do not formulate such queries as <domain independent OR management>. An explicit query *q* is a set of terms selected from the term space and connected by the logical operators: <NOT>, <AND> or <OR>. A query is essentially a Boolean expression. The selection of facets, the terms sets within each facet and the operators is achieved by scrolling the list and clicking the terms using the mouse. Furthermore, the operators correspond to the query language.

BNF query syntax:

```
Boolean_query ::=   facet_term
                    |facet_term AND facet_term
                    |facet_term OR facet_term
                    |NOT facet_term
                    |Boolean_query AND Boolean_query
                    |Boolean_query OR Boolean_query
```

Retrieval Mechanism

The goal of the business analysis pattern mechanism is to find one or several business analysis patterns that either partially or fully match a query and to rank them according to the model described above.

Given a query, the retrieval mechanism uses the relationships that were created between business analysis patterns and the term space during the classification to match and retrieve the most relevant business analysis patterns for the query. This process is known as a relaxed match since an exact match is not insisted upon (i.e. it is relaxed).

If an exact match is achievable, then it will be performed. During this process, the Boolean query is converted into an SQL query. The Boolean-SQL algorithm was designed to convert a Boolean logic search string into a basic SQL statement. The algorithm supports <AND>, <OR> and <NOT> Boolean operators. Failure to include a Boolean operator between terms results in an invalid query, which is then rejected by the system.

After a total match has been performed, business analysis patterns that best fit the query will be retrieved. The condition for retrieval is relaxed to implement the ranked mechanism. A 'fuzzy Boolean' set of operators and facet weights is proposed. The

idea is for the meaning of <AND> and <OR> to be relaxed so that the elements appearing in some of the operators are retrieved instead of forcing an element to appear in all the operands <AND> or at least in one of the operators <OR>. The <AND> may require it to appear in more operands than the <OR>. Nevertheless, the operator <NOT> is considered as per the traditional meaning of the Boolean model. Moreover, when business analysis patterns have a larger number of elements in common with the query, these achieve a higher rank. During this process, the weight of facets is also considered.

The Matching Process

On pushing the 'fuzzy Boolean' model even further, the distinction between <AND> and <OR> is blurred. A business analysis pattern including more occurrences of the query terms is more likely to be relevant than a business analysis pattern with fewer occurrences of the same terms (Greiff, 2000). For this reason, a query simply becomes a list of words and all the business analysis patterns matching a portion of the designer's query are retrieved. Higher ranking is assigned to those business analysis patterns that have matched more terms of the query. The negation <NOT> handles the terms that are not desired by allowing the designer to express this in a query so that business analysis patterns including these are retired from the ranking computation.

A business analysis pattern vs. a terms matrix (BAPTM) is used since business analysis patterns and queries are made up of words. The BAPTM matrix has a column for each pattern and a row for each term with matrix entries (i.e. business analysis patterns that have been classified) incorporating those terms.

In order to answer this query, the system will first try to achieve a perfect match. If it succeeds, then the business analysis patterns satisfying the query will be shown. In the following step, business analysis patterns are ranked according to the number of terms each one has in common with the query and the facet weights. According to our definition, certain facets have priorities over others. For example, Domain is more important than DomainArea. This means that terms belonging to certain facets have higher priorities than others do. In order to incorporate this process in the ranking of business analysis patterns, an array of weights is created for each facet term. Patterns are ranked in accordance with the number and the sum of weights. If no terms are found to be in common with the query and the classified business analysis patterns by the end of the process, then the results of the BAPTM matrix will be empty. A message will be spewed out to the designer, informing him that no pattern satisfied the query.

RIBAP Design

The Reuse Infrastructure Based on Business Analysis Patterns (RIBAP) tool prototype is intended to help the reuse of business analysis patterns activities. The RIBAP prototype is a tool that coexists with other developed tools. These tools support the expression of a business analysis patterns solution, which may be EKD, UML and text in free natural language. The first tool supports the representation of EKD models and it was developed as part of the *DELOS* integrated environment (Klimanthianakis, 1997; Klimanthianakis and Loucopoulos, 1997). The second tool is Microsoft Visio and it is used to express UML models; finally, the third tool is Microsoft Word, used to express solutions in free natural language; Microsoft developed these two tools. The RIBAP tool prototype is designed as a standalone environment where these tools are integrated.

The RIBAP tool prototype is based on a business analysis patterns repository, namely, the RIBAP repository, a central element of the development environment that intends to help a designer in: (1) sharing the "best practices" on business (2) creating new solutions based on existing ones; and (3) redesigning existing organisational processes. To achieve these goals, the RIBAP tool prototype supports the organization and management of business analysis patterns and their retrieval, evaluation and adaptation. The prototyping tool developed according to the proposed framework is detailed in the following sections.

Tool Architecture

By offering a set of editors, tools and a repository, the RIBAP tool prototype supports every aspect of the application to reuse business analysis patterns. RIBAP is organised as shown in Figure 3, and it consists of:

- a set of ***reuse tools***, namely, a classifier, editors, a retriever and a browser permitting manipulation and reuse of business analysis patterns. These tools share a common interface;
- ***base services***, which are a set of high level functions for reuse tools to access the repository; and
- ***repository*** storing business analysis patterns that belong to a language or to a catalogue of a classification scheme.

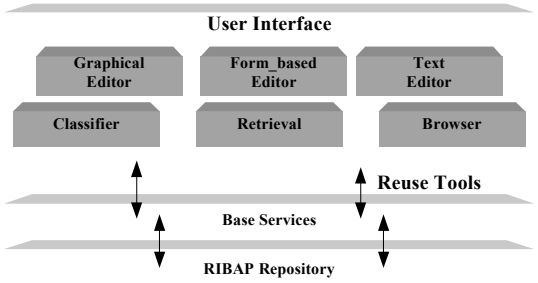

Figure 3. The RIBAP Architecture

The RIBAP Repository consists of the business analysis pattern, the classification schema and the catalogue/language organisation repository. Examples of business analysis patterns and related models (UML, EKD and free natural language) are saved in the business analysis patterns base. The classification schema as well as the relationships between the classification and the analysis is saved in the *classification base*. The language organisation is saved in the *language organisation base.* Furthermore, the relationships between patterns that belong to a catalogue are stored in the *business analysis pattern relationship base*. The *classifier* tool allows the *RIBAP repository* to be updated and its contents to be listed. The *form_based* as well as the *text editor* or the *graphical editor* is used whenever the classifier is called to update the business analysis pattern base.

The *user interface* allows the designer to submit a query conducted and controlled by the r*etrieval* tool to retrieve examples of business analysis patterns that are stored in the *business analysis pattern base***.** The *retrieval* fetches the business analysis pattern(s) that totally or partially satisfy the query and the business analysis pattern(s) description(s) is/(are) presented to the designer. The business analysis pattern(s) description(s) is/(are) presented to the designer in order to provide him with the information he needs to know under what circumstances the business analysis pattern is relevant to reuse. Once a business analysis pattern is selected, the justifier is presented and the solution can be visualised via one of the *editors* in accordance with the body type. At this point, the designer can decide whether the solution is suitable or whether it must be adapted or rejected.

By using the browser, the *user interface* allows browsing through the business analysis pattern repository. All the business analysis patterns, which are organised by languages and catalogues, are presented to the designer and the reuse process occurs when a business analysis pattern is selected.

The tool is aimed at two kinds of users, namely, the reuse engineer and the designer. The reuse engineer's job is to create a basis of knowledge at both a classification and business analysis pattern level. The designer's job is to interact with the tool in order to find business analysis patterns that can be reused in his work (either adapted or "as-is").

The Implementation

The implementation of RIBAP mainly focused on the process concerning the reuse of business analysis patterns.

RIBAP – User Interface

The RIBAP prototype tool is described in this section as the set of UI components that describe RIBAP's functionality. RIBAP provides its designers (users) with a simple, yet effective user interface. The prototype features several windows. Each window allows direct and visual manipulation of various repository objects, which include a template-level user interface.

Once the RIBAP tool is activated, the designer is presented with the initial, main screen (depicted in Figure 2). The screen is divided in two main sections. The top section contains menus and buttons that perform several actions. The menu-driven dialogue may be used to activate the different functions of the RIBAP tool. In order to accelerate the speed of use of the RIBAP functions, a list of the most commonly used buttons is displayed just below the menu area.

The bottom section of the browser tool is split into two sides. The left side, called *Tree view*, contains a view of the contents of the RIBAP repository using standard *Explorer* format. The right side, called *List view,* shows the contents of a catalogue or language in accordance with the active item (catalogue or language) in the *Tree*.

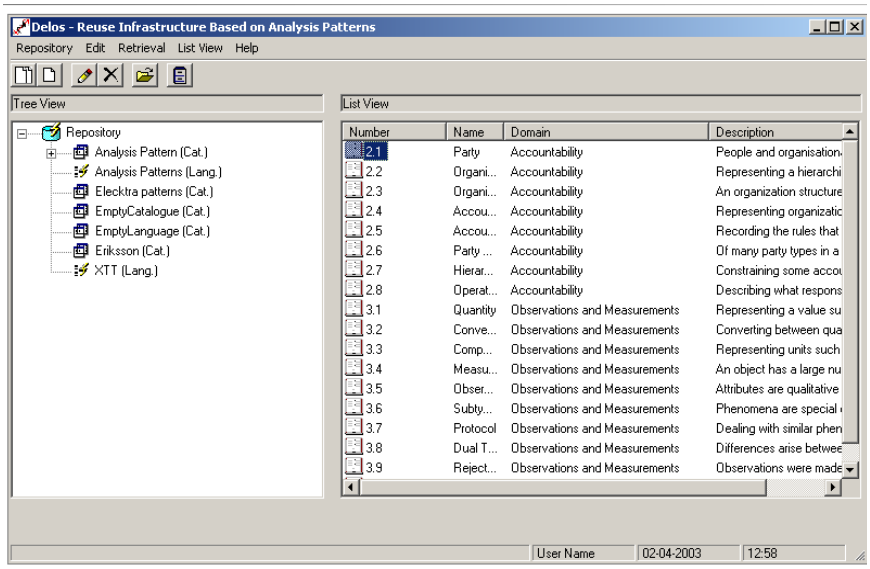

Figure 4. Initial Screen in RIBAP: Visualisation of Catalogue Contents

The RIBAP repository contains business analysis patterns that belong to a catalogue or a language. These are organised according to the RIBAP model. The tree structure shown on the left-hand side of the screen also reflects this organisation. The tree view allows browsing through the RIBAP repository. *List* view shows the contents of both a catalogue and a language. When a catalogue is selected in tree view, the designer can see the details of analyses patterns through the *list* view, i.e., number, name, domain and description (as shown in Figure 4). Furthermore, the designer is able to select business analysis patterns and see their respective details according to the reuse process.

When a language is selected, the designer can see the list of patterns that belong to the language. Whenever a pattern is selected, the designer is also able to see related patterns and the different paths permitted by the language (as shown in Figure 5).

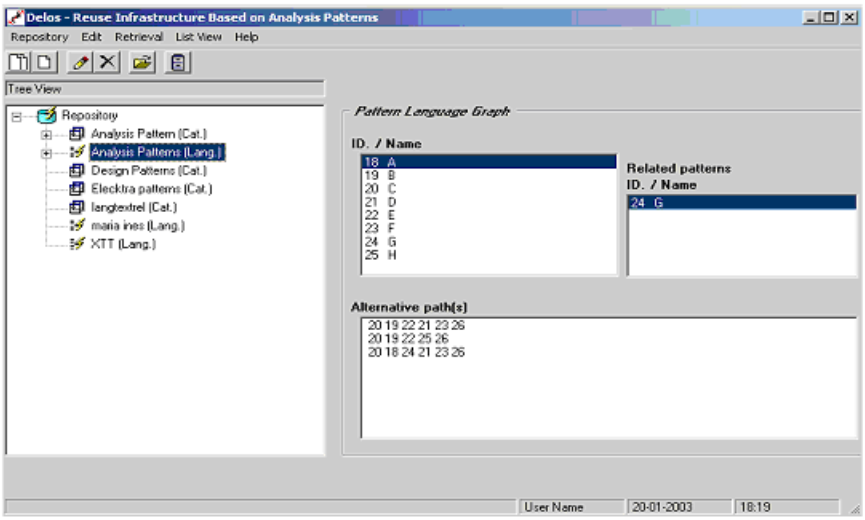

Figure 5. Initial Screen in the RIBAP Prototype: Visualisation of a Language Contents

The business analysis pattern editor and the classifier form are shown in Figure 6. The pattern editor caters for the description of patterns that are developed according to EKD, UML or to free natural language. From the pattern editor the graphical editor or text editor is called to stipulate the body of the business analysis pattern. The classifier is used to classify the business analysis pattern. Terms from the term space are selected with a mouse click. A business analysis pattern is classified and saved in the repository with the help of this/these form(s).

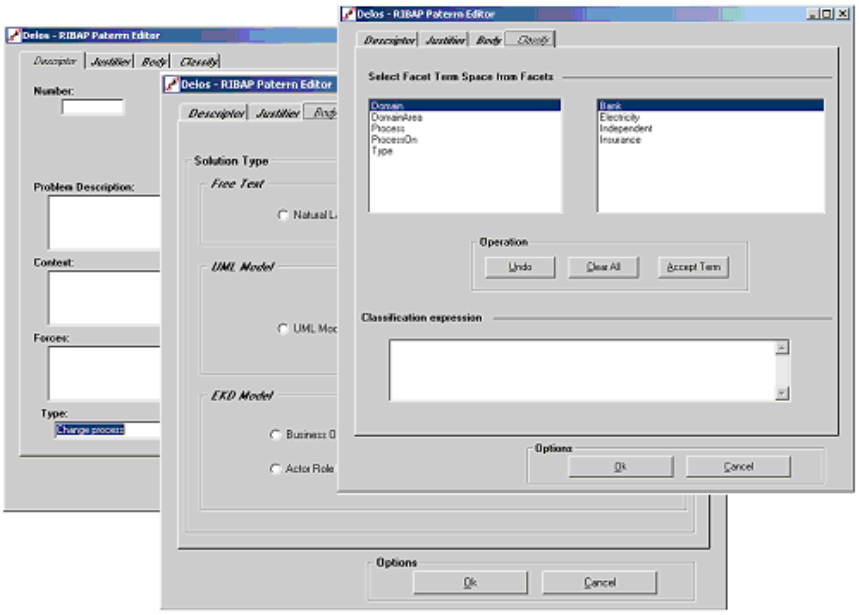

Figure 6. Pattern Editor and Classifier

The retrieval process begins with the query formulation that is supported by the window shown in Figure 7.

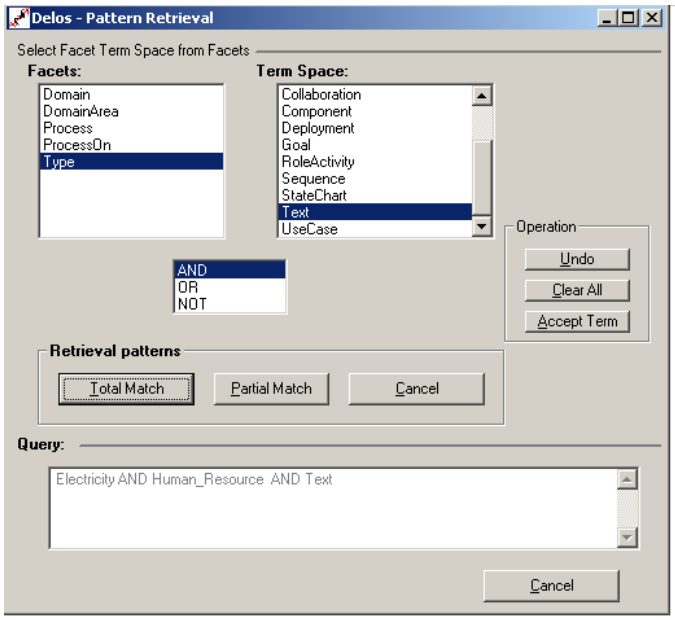

Figure 7. Support to the Query Formulation

RIBAP Performance Evaluation

System performance was measured on a Pentium III, with 130.612 KB of RAM and a 500 MHz CPU, Running Microsoft Windows 2000 Professional.

Several tests were carried out on the tool to find the discrepancies between the prototype and the original objectives. The tests concluded that prototyping achieved the goals. The tests were carried out using EKD and Fowler's catalogue. Two arbitrary languages were created to test the analysis pattern language.

Additional tests were conducted to measure system time from the time of browsing the RIBAP repository and selecting the analysis pattern to the time of query formulation and retrieval of analysis patterns that total or partially match a designer's query. Since the system is only a prototype, measurements merely provide an indication of the worst-case performance.

The time interval between the selection of an analysis pattern and its visualisation is almost zero since the process is instantaneous. The time taken to formulate a query and to retrieve analysis patterns that match the query was dependent on the number of analysis patterns. Two size examples were considered, i.e. tiny (64 analysis patterns) and small (140 analysis patterns).

The test revealed that RIBAP is up to 1.5 times faster when analysis pattern are taken from a tiny repository as opposed to a small one. This time revelation means that retrieval is executed very quickly. Nevertheless, by incorporating algorithms that are more efficient as well as better database systems, system performance could be further improved.

Conclusions

Business analysis patterns are a promising vehicle in requirements engineering reuse, they could be used are a way to: (1) enhancing agreement and a common understanding of a proposed solution amongst a wide group of stakeholders, and (2) enhancing validation of the solution in terms of evaluation solution feasibility. Additionally, a solution model can be used as a starting point for a more detailed RE process.

This paper dealt with the development of a business analysis-pattern reuse framework and focused on the technical aspects of the business analysis pattern description and business analysis pattern retrieval. We identified conceptual challenges solved by the framework. These may all be attributed to the problem of currently describing business analysis patterns and retrieving them. Furthermore, the framework proposed must be satisfied if a reuse tool is to be successful.

The RIBAP tool is a prototype tool; it was (and is being) developed to study the best, most effective and most efficient way for a tool to support reuse of business analysis patterns. RIBAP was used in our research to experiment and evaluate the proposed approach to reuse business analysis patterns. RIBAP has been evaluated with respect to both functionality and user-friendliness. RIBAP's retrieval is already a usable system. It features characteristics that are expected of developed search engines and document indexers. It uses a lexicon to assist the classification of analysis patterns, employs a retrieval method and provides a list of ranked analysis patterns. In addition, the browser permits one to navigate through catalogues and languages and to inspect analysis patterns.

We have focused so far on RIBAP's development in providing the viability of the catalogue/language-base approach to integrate tool support for reuse analysis patterns. Features of the tool resulting from the framework are: (1) the repository of patterns supports languages and catalogues and allows cross-referencing between patterns within languages; (2) the pattern template proposed makes reuse process activities both easier and complete; the information embodied in a business analysis pattern is clustered in such a way that the evolution and revision of activities in the reuse process is supported by required and sufficient information; (3) support different business analysis pattern types (business process patterns and change process patterns using different nomenclatures, namely, UML, EKD and free natural language); (4) ability to specify relationships between patterns belonging to a pattern catalogue or language; (5) Proposes an analysis retrieval model that supports: (a) a classification mechanism to structure patterns in order to facilitate the finding of patterns (b)a representation of the designer's queries (c) a mechanism to retrieve business analysis patterns satisfying a query and ranking them according to the designer's query.

References

Alexander, C., Ishikawa S., Silverstein M., Jacobson M., Fiksdahl-King I., and Angel S. "A Pattern Language", Oxford University Press, New York, 1977.

Banker, R. D., Kauffman, R. J., and Zweig, D. "Repository Evaluation of Software Reuse", IEEE Transactions on Software Engineering (19:4), April 1993, pp. 379-388.

Bellinzona, R., Fugini, M. G., and Pernici, B. "Reusing Specifications in OO Applications", IEEE Software (12:2), March 1995, pp. 65-75.

Berry, D. M. and Lawrence, B., "Requirements engineering", IEEE Software, (15:2), March/April 1998, pp. 26-29

Constantopoulos, P., and Pataki A. "A Browser for Software Reuse", 4th International Conference on Advanced Information

Systems Engineering - CAiSE '92, Manchester, UK, May 1992.

Coplien, J., and Schmidt, D. C. "Pattern Languages of Program Design 2", Addison Wesley, USA, 1995.

Davis, A. M. "Software requirements – objects, functions and states", Prentice Hall, USA 1993.

Ferreira, M. J., and Loucopoulos, P. "Organisation of Analysis Patterns for Effective Reuse", In proceedings of Third International Conference on Enterprise Information Systems - ICEIS 2001, Setúbal, Portugal, July 2001.

Ferreira, M. J., and Loucopoulos, P. "Towards an Effective Analysis Pattern Retrieval", in Proceedings of Tools Eastern Europe 2001, Sofia, Bulgaria, March 2002.

Filippidou, D. "SEED: Scenario-based Experimentations to Evaluating Designs", PhD Thesis, UMIST, UK, 1997.

Fowler, M. "Analysis Patterns: Reusable Objects Models", Addison-Wesley, USA, 1997.

Glass, R. L. "Reuse: What's Wrong With this Picture?", IEEE Software (15:2), March/April 1998, pp. 57-59.

Greiff, W. R. "The Use of Exploratory Data Analysis in Information Retrieval Research", in Advances in Information Retrieval – Recent Research from the Center for Intelligent Information Retrieval, Croft W B (ed), Kluwer Academic Publishers, USA, 2000.

Grosz, G., Semmak, F., Brash, D., and Prekas, N. "Domain Patterns for Constructing Conceptual Specifications", submitted for publication, 1999.

Holbrook, C. H. "A Scenario-based Methodology for Conducting Requirements Elicitation", ACM Sigsoft – Software Engineering Notes (15:1), January 1990, pp. 95-104.

Hsia, P., Samuel, J., Gao, J., and Kung, D. "Formal approach to scenario analysis", IEEE Software, (11:2), March 1994, pp. 33-41.

IEEE "IEEE Guide to Software Requirements Specifications", ANSI/IEEE Standard 830- 1984, Institute of Electrical and Electronics Engineers, New York, 1984.

Jr, V. F. L. "Facet-Based Classification Scheme for Industrial Automation Software Components", in Proceedings of the European Conference on Object Oriented Programming -ECOOP 2001 - Sixth International Workshop on Component-Oriented Programming, Budapest, Hungary, June 2001.

Klimanthianakis, P. "Delos - A Repository Based Environment for Development Network Centric Information System", Marter's, UMIST, Mancester, UK, 1997.

Klimanthianakis, P., and Loucopoulos, P. "A Repository Based Environment for Development Network Centric Information System", in Proceedings of the CaiSE'97, Barcelona, Spain, 1997.

Knight, J., and Kienzle, D. "Reuse of Specifications", in Proceedings of 5th Annual Workshop on Institutionalising Software Reuse, Palo Alto, California, USA, October 1992, pp. 767-775.

Kotonya, G., and Sommerville, I. "Requirements Engineering with Viewpoints", *Software Engineering Journal,* January 1996, pp. 5-18.

Kotonya, G., and Sommerville, I. "Requirements Engineering – Processes and Techniques", John Wiley & Sons, Inc., USA, 1997.

Kramer, J., and Ng, K. "Animation of requirements specifications", Software Practice and Experience (18:8), 1988, pp. 749-774.

Loucopoulos, P. "The Generic Pattern Model", UMIST, Research Report (ELEKTRA project), ELEKTRA/WP5/T5.1/UMIST/1, December 1997.

Loucopoulos, P., and Karakostas, V. "System Requirements Engineering", McGraw –Hill, England, 1995.

Noble, J. "Classifying Relationships Between Object-Oriented Design Patterns", in Proceedings of Australian Software Engineering Conference – ASWEC'98, Adelaide, Australia, November 1998, pp. 98-109.

Pohl, K. "Process-centered requirements engineering", Research Studies Press LTD, England, 1996.

Prekas, N., Loucopoulos, P., Rolland, C., Grosz, G., Semmak, F., and Brash, D. "Developing Patterns as a Mechanism for Assisting the Management of Knowledge in the Context of Conducting Organisational Change", 10th International Conference & Workshop on Database and Expert Systems Applications (DEXA 99), Florence, Italy, 1999.

Rolland, C., and Prakash, N. "From Conceptual Modelling to Requirements Engineering", Technical Report Series 99-11, CREWS, 1999.

Staringer, W. (1994) "Constructing Applications from Reusable Components", IEEE Software (11:5), September, pp. 61-68.

TSE "Special Issue on requirements Engineering", IEEE Transactions on Software Engineering, 1997

William, D., and Kennedy, M. "A Framework for Improving the Requirements Engineering Process Effectiveness", in Proceedings of International Council on Systems Engineering, 1999, Web publication, **http://www.sbu.ac.uk/php-cgiwrap/~scismdbs/search_results.php3**. Available 2003.

Ye, Y., Fischer, G., and Reeves, B. "Integrating Active Information Delivery and Reuse Repository Systems", Software Engineering Notes – Proceedings of the ACM SIGSOFT 8th International Symposium on the Foundations of Software Engineering (FSE-8), San Diego, CA, November 2000, pp. 60-68.

Zave, P. "Classification of Research Efforts in Requirements Engineering", ACM Computing Surveys (29:4), 1997, pp.315-321.

Zimmer, W. "Relationships Between Patterns", in Pattern Languages of Program Design 1, Coplien J and Schmidt (eds), Addison Wesley, USA, 1994.