A Framework based on a Pattern Language for Business Resource Management

Rosana T. Vaccare Braga[1]
Paulo Cesar Masiero[2]

ICMC-Universidade de São Paulo
C. P. 668 – 13560-970 – São Carlos – SP – Brazil
{rtvb,masiero}@icmc.sc.usp.br

Abstract

A framework based on a pattern language for Business Resource Management is under construction to be used for developing systems in the information systems domain. Business resources include assets and services, as products and repairs. By management we mean their trade, rental and maintenance. A pattern language has been developed for aiding in the analysis of systems for business resource management. It includes fourteen patterns for identifying the resource, quantifying it, renting, trading and maintaining it, reserve it, quote the trade, quote the maintenance, check its delivery, allowing several items to be dealt with in the same transaction, paying for the transaction, identifying the transaction executor, and identifying maintenance tasks and parts. One of the patterns is included, corresponding to trading the resource. A class model shows its structure (classes and relationships), and an application example shows the roles played by each participant class. An example of the pattern language usage for a car repair shop system is presented, together with an object model that shows the patterns used and the roles played by their participant classes. The present development stage of the framework is mentioned.

Keywords*: analysis patterns, pattern languages, frameworks, business systems, business resources, reuse*

[1] Financial support from FAPESP grant 98/13588-4
[2] Financial support from CNPq and FAPESP

1 - Introduction

Software Patterns document a relation between a certain context, a design problem, and a solution, describing a decision point in the development of an application [3]. Its main purpose is to promote reuse at different abstraction levels: from analysis to design and implementation of software systems. Pattern languages consist of related patterns organized as a tree or graph structure, representing the sequence of decisions in time leading to the complete design, so that a pattern language becomes a method that guides the development process [3]. Integration of design patterns to form pattern languages is considered a challenging and time consuming activity [8], but provides the greatest payoff for pattern based software development.

Frameworks are reusable designs of all or part of a software system described by a set of abstract classes and the way instances of those classes collaborate [7]. A bi-directional relationship exists between frameworks and pattern languages for a specific application domain [3]. A framework allows the implementation of an application designed following a pattern language and a pattern language offers the rules for the use of framework elements [5] and for its extension.

Various analysis patterns have been presented recently, including patterns for business systems [1, 2, 6, 13]. Several frameworks are available to support system infrastructure areas, as operating systems, user interfaces and communications [3]. However, frameworks to support enterprise applications that begins to appear are usually proprietary, as for example, IBM San Francisco [12] and Enterprise Resource Planning [3]. Such frameworks are becoming strategic assets for organizations across all business sectors.

In this doctoral research, we present a pattern language [10, 11] and a framework to deal with business resource management applications, a particular domain of information systems. The pattern language resulted from the experience of more than ten years of systems development practice for medium and small business in this domain. It is composed of several patterns, some of which are specific applications of recurring patterns proposed in the literature [1, 2, 6]. However, our pattern language stays on a higher abstraction level than those patterns, as it is applicable to a more specific domain and contains the semantics inherent to a family of applications in the domain. It provides inexperienced developers with substantial material to develop new systems, because it guides them during system analysis, providing alternative solutions as necessary.

Based on this pattern language, we are now building a framework for systems in the business resource management domain. The goal of the framework is to provide the developer with most of the code that he or she needs to instantiate a particular application of this domain.

The paper is organized as follows. In Section 2 an overview of the pattern language is provided. In Section 3 it is shown how the pattern language can be applied to a car repair shop system. In Section 4 the present implementation of the framework is mentioned. In Section 5 the conclusions are presented.

2 - The Business Resource Management Pattern Language

Our pattern language was designed to help software engineers to develop applications in which business transactions such as resource rental, trade or maintenance need to be logged. Thus, the problem domain includes applications like video rentals, library services, medical

attendance, hotel operation, retail stores, and repair shops, among others. Business systems that involve financial aspects, like banks and insurance companies, are not covered by the present pattern language. Resource rental focuses primarily on the satisfaction of a certain temporary need of a product or service like a videotape or a physician time. Resource trade focuses on the transference of property of a product, as for example a product sale. Resource maintenance focuses on the maintenance of a certain product, using labor and parts to perform it, as in an electric appliance repair shop.

The language is formed by fourteen patterns, whose use depends on the characteristics of the application. Figure 1 shows the dependencies among the patterns and the order in which they are generally applied, denoting the main patterns with a thicker line. The patterns are grouped according to their purpose, as illustrated in Figure 1: Group 1 patterns are basically concerned with the identification of the business resources; Group 2 patterns deal with the business transactions performed by the system; and Group 3 patterns take care of details associated to most business transactions. To apply the pattern language for a particular system, the user has to follow the patterns of Figure 1, beginning with pattern 1 and using the instructions contained in the "Following Patterns" section to proceed. The result is the system class diagram.

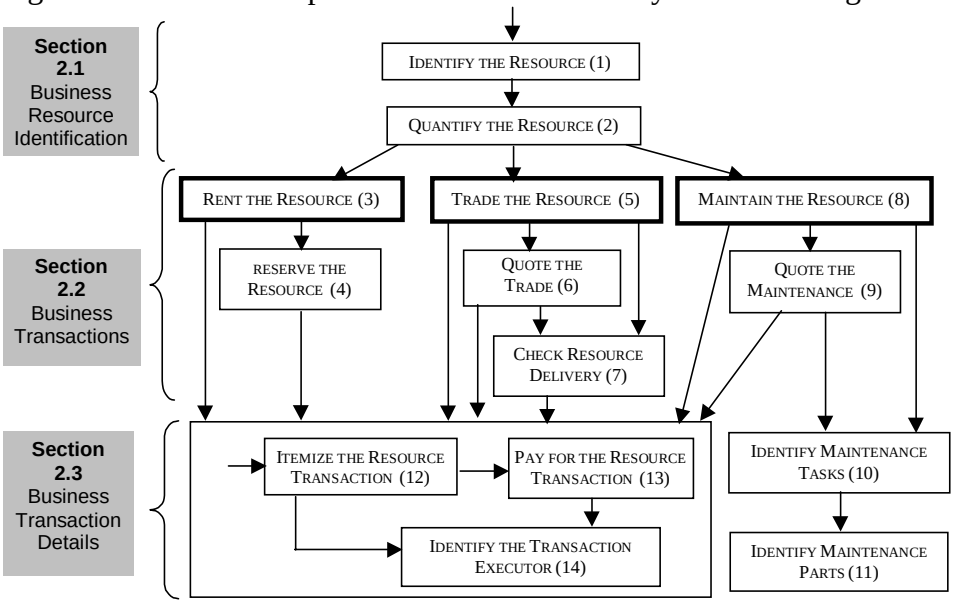

Figure 1: Dependencies among patterns

Figure 2 shows in detail one of the patterns of the pattern language, using UML (Unified Modeling Language) [4] to express its structure. The full pattern language can be found elsewhere [10, 11].

3 - An Application Example

Applying the Pattern Language for modeling a car repair shop system, patterns 1, 2, 5, 11, 12, 13 and 14 were used for the Repair sub-system, where "car" is the resource being managed, and patterns 1, 2, 4, 10, 11, 12, 7 and 10 were used for the Purchase sub-system, where "part" is the resource being managed. The object model produced is shown in Figure 3. The tags show the role played by the class it points to. They contain labels like "P#n: role", where "n" is the pattern number and "role" is the role played by the class in that pattern. For example, in Figure 3, class

Pattern 4: TRADE THE RESOURCE

Context

Your application deals with trade of resources, which may involve resources sold and/or purchased. You have already identified and quantified these resources. Resource trading may be thought of as a resource property transference, in which a resource owned by one party becomes owned by another party. In a sale, if the resource is not available in stock, then the customer can fill in a request that will be granted when possible. In a purchase a request is made to the supplier who delivers the resource within a certain period.

Problem

How do you manage the resource trades made by your application?

Forces

- It is essential to log trade information, because it can be used to generate important reports on resources demand and organization gains (most systems in this domain are concerned with profits).
- The additional storage space and processing time required to log trade information has to be balanced by against possible gains in system functionality when evaluating costs versus benefits.

Structure

Figure 12 shows the TRADE THE RESOURCE pattern.

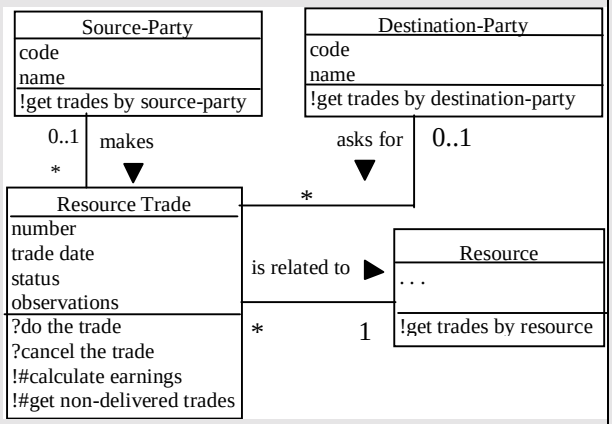

Figure 12: TRADE THE RESOURCE pattern

Participants

Resource Trade: represents all the details involved on trading the resource. The attribute `status` denotes the trade stage: pending, partially fulfilled, or fully fulfilled. When the MEASURABLE RESOURCE sub-pattern has been applied earlier, then an attribute `quantity` is added to denote a non-unitary resource trade.

Resource: as described in previous patterns.

Source-Party: represents the original resource owner, for example, in the case of a sale it is the organization department or branch that sells the resource, and in the case of a purchase it is the supplier organization. This class is optional for small sale systems where there are no departments or branches.

Destination-Party: represents the final resource owner, for example, in the case of a sale it is the customer buying it, and in the case of a purchase it is the organization department or branch buying the resource. This class is optional for small purchase systems where there are no departments or branches.

Example

Figure 13 shows an instantiation of the TRADE THE RESOURCE pattern for an Inventory Control system.

Following patterns

A trade is followed by a delivery and can be preceded by a quotation. So, now, try to use the patterns QUOTE THE TRADE (7) and CHECK RESOURCE DELIVERY (9). Also look at Section 2.3 patterns, which are useful for modeling other trade details.

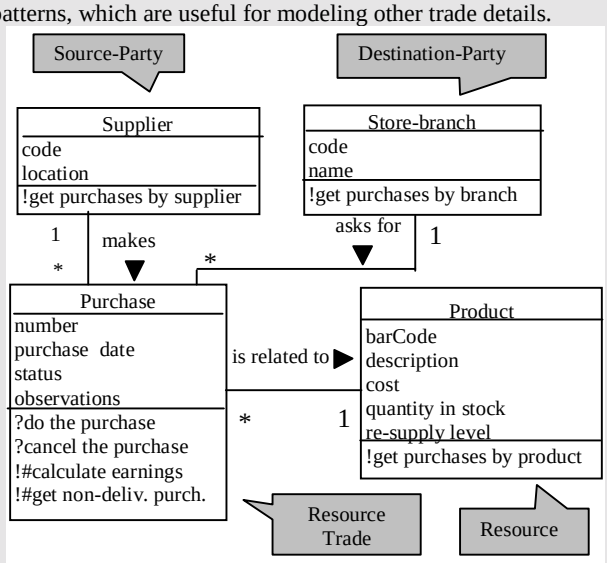

Figure 13: Instantiation of the TRADE THE RESOURCE pattern

Figure 2: Example of a pattern: TRADE THE RESOURCE [11]

"Purchase" represents the "Resource Trade" participant of pattern 4 ("Trade the Resource"), shown in sub-Figure 12 of Figure 2.

4 – The Framework

A framework is being developed based on the pattern language. After identifying the possible hot-spots, i.e., variable aspects of the application domain, the framework class hierarchy

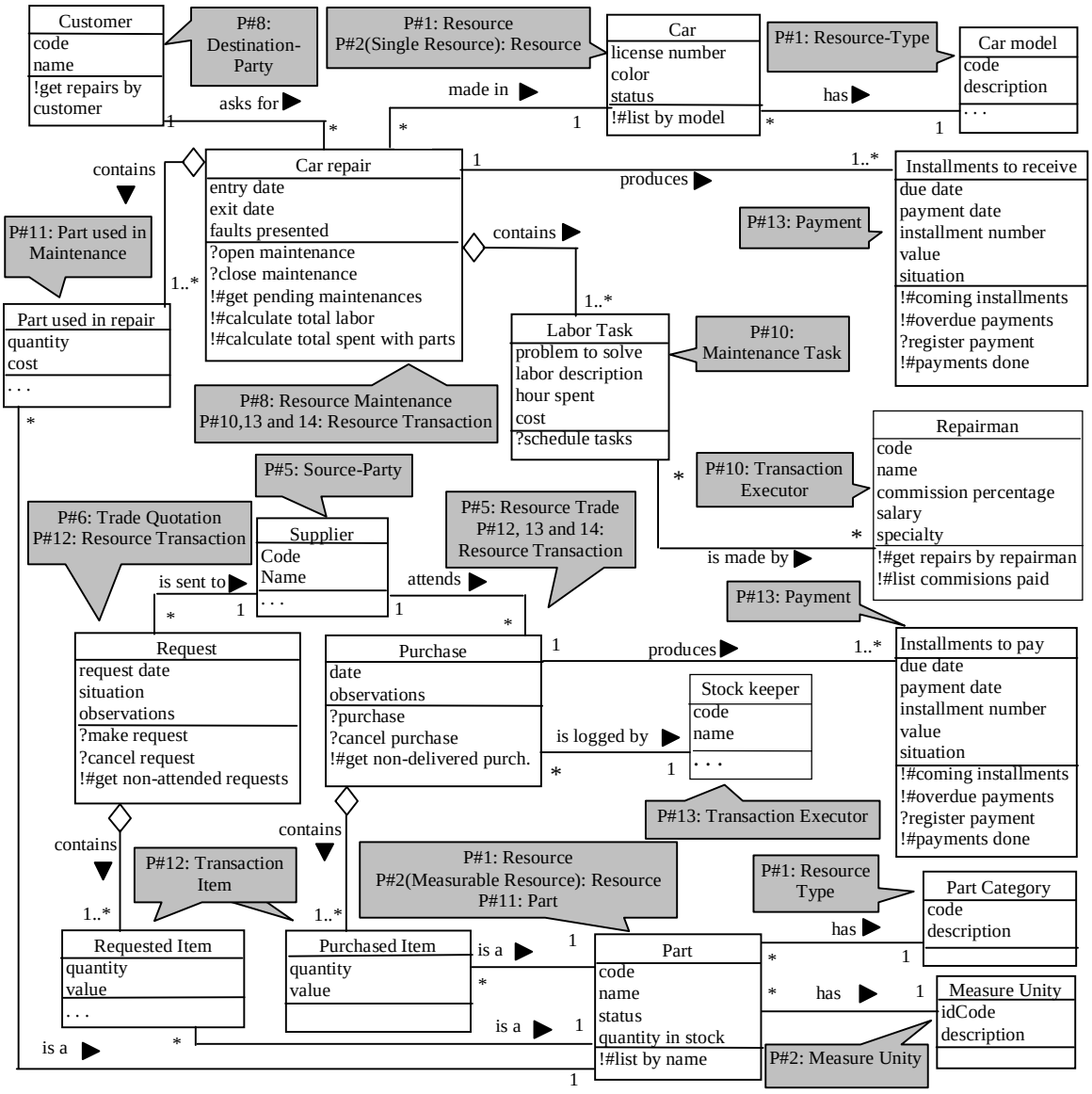

Figure 3: Application of the Pattern Language to a simple Car Repair Shop

is being designed and classes and methods are being implemented, using VisualWorks [9]. We are following the same order of the pattern language to implement the framework, in an evolutionary life cycle. Initially, we have implemented the classes of patterns 1 and 2, obtaining a little framework for registering business resources. We have instantiated it to resources such as products, books, cars, etc. Next, we have implemented pattern 3 and instantiated the framework for simple applications such as video rental, car rental and product rental. At present, we are implementing pattern 4.

5 - Conclusions

The application of the pattern language eases the analysis of new cases, supplying a guideline for it to be more disciplined, with the assurance that the main aspects of the system are

covered. More experiments are planned using newly graduated students with no professional practice and with other domain experts, in order to get some feedback to improve the pattern language usability, quality and completeness, as well as to evaluate gains in reusability and productivity. These experiments will compare the effort to develop a system from scratch, that is, using existing modeling approaches such as UML, Fusion, etc., and using our pattern language.

The framework being implemented is of type white-box, i.e., its instantiation is done by creating subclasses of certain abstract classes. To know which classes to subclass, a documentation of the framework was done, including a cookbook and object models corresponding to each implemented pattern. The pattern language is used in parallel with the cookbook – actually, they are mutually complementary, as the developer follows the pattern language to obtain the application object model and uses the cookbook to define which classes to instantiate. Next, we plan to do an experiment to transform this part of the framework into a black-box one. This will ease its usage, as the developer will be able to choose the components of his/her application in order to obtain most of the design and code necessary for the implementation.

We are also investigating the relation between pattern languages and frameworks – more specifically, we are interested in how a pattern language can be used to help in the framework design and instantiation. Intermediate results show that some sections of the patterns are a rich source for finding the framework hot-spots, and that wizards can be built for the framework instantiation, following the same order of the pattern language's patterns.

References

[1] Boyd, L. *Business Patterns of Association Objects*. In "Martin, R.C.; Riehle, D.; Buschmann, F. (eds) Pattern Languages of Program Design 3, Addison-Wesley, 1998", p. 395-408.

[2] Coad, P.; North, D.; Mayfield, M. *Object Models: Strategies, Patterns and Applications*, Yourdon Press, 2nd edition, 1997.

[3] Fayad, M. E.; Johnson R. E. *Domain-Specific Application Frameworks – Frameworks Experience By Industry*, Wiley, 2000.

[4] Eriksson, H-E; Penker, M. *UML Toolkit*, Wiley Computer Publishing, 1998.

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

[6] Johnson, R. ; Woolf, B. *Type Object*. In "Martin, R.C.; Riehle, D.; Buschmann, F. (eds.) Pattern Languages of Program Design 3, Addison-Wesley, 1998", p. 47-65.

[7] Roberts, D.; Johnson, R. E. *Evolving Frameworks*: *A Pattern Language for Developing Object-Oriented Frameworks*, In "Martin, R.C.; Riehle, D.; Buschmann, F. (eds) Pattern Languages of Program Design 3, Addison-Wesley, 1998", p. 471-486.

[8] Schmidt, D. C.; Fayad, M.; Johnson, R. E. (guest editors). *Software Patterns*. Communications of the ACM, V. 39, nº10, p. 36-39, October 1996.

[9] Cincom Systems, Inc. *VisualWorks 5i.1*, http://www.cincom.com/visualworks/

[10] Braga, R.T.V.; Germano, F.S.R.; Masiero, P.C. *A Pattern Language for Business Resource Management*, proc.of 6th Pattern Language of Programs Conference (PLoP'99), Monticello-IL, EUA, v.7, p. 1-34, Aug 99.

[11] Braga, R.T.V.; Germano, F.S.R.; Masiero, P.C. *A Pattern Language for Business Resource Management Systems*, submitted to the Journal of Brazilian Computer Society, 20p., 2000. Available for FTP at: http://www.icmc.sc.usp.br/~rtvb/pat_lang_jbcs.zip.

[12] Fayad, M. E. , Schmidt, D. C. Johnson, R. (eds.) *Implementing Application Frameworks: Object-Oriented Frameworks at Work*, John Wiley & Sons, 1999.

[13] Fowler, M. *Analysis Patterns*. Addison-Wesley, 1997