The Use of the Catalysis Approach in the Construction of a Framework in a N-tier Architecture

Sergio A. Tanaka
UNIFIL, Computer Science Dept.
Londrina - PR, Brazil, 86020-000
tanaka@filadelfia.br

Ruy T. Nishimura
UNIFIL, Computer Science Dept.
Londrina - PR, Brazil, 86020-000
ruynishimura@filadelfia.br

and

Carlos J. M. Olguín
UEM, Informatics Dept.
Maringá - PR, Brazil, 87020-900
olguin@din.uem.br

Abstract

This paper presents a practical case of use of the Catalysis Approach in the Construction of a Framework in an N-tier Architecture. This framework was used to develop the Values Quotation component of PLATIN – Londrina Platform of Information Technology. The development of the component was made by the Software Factory team with the support of the Software Engineer Laboratory of UNIFIL – University Center Filadélfia. It served as a study case to show the use of recent techniques for the development of components using distributed objects in the tiers architecture style, and to demonstrate how these techniques can complement each other.

The other result of this work is related to the production of a reusable component – values quotation – which can be used in other applications such as cost estimation of materials and equipment price.

Keywords: Component-based development, N-tier Architecture, distributed objects, patterns, frameworks, RUP.

1 Introduction

Software engineering objectives are centered in the quality improvement of both the developing process and the software product. Another important goal is the reduction of efforts and costs involved in software production. In this context reuse is an important technique, as it allows software construction from well specified and tested components using frameworks and models (templates). The use of a component-based development technique in an N-tier architecture, in conjunction with distributed objects technologies, increase reusability, productivity, and interoperability.

An object-oriented framework is frequently characterized by a concrete and an abstract group of classes which collaborate among each other to provide the skeleton for an application. Catalysis [4] gives a wider-range concept of frameworks denominated “models frameworks”. In this approach, frameworks are projected in a higher level of abstraction defining the architecture of a generic application to be imported, with substitutions and inclusions, generating, in this way, specific applications.

The Component-Based Development (CBD) is a technique for software development in which one component is added, linked, or aggregated to another to constitute a bigger system. These components could be composed by just one business class or by a whole system with interfaces, architectures, and business models. A component represents a software artifact with a well-defined interface. A component’s interface is represented by one or several types of behavior provided by the component [4].

The N-tier architecture used in the Component-Based Development has many advantages and benefits the development process. In this architecture the business rules are installed in the server, allowing the client’s machine to have the minimum possible processing (thin client). In contrast, in the traditional architecture (Client/Server), the business rules are installed within the client’s machine, demanding machines with bigger processing capacity (fat client).

Expressed in terms of business modeling, the software development process is a business process; the Rational Unified Process (RUP) is a generic business process for object-oriented software engineering. It describes a family of related software engineering processes which share a common structure and process architecture. It provides a disciplined approach to assigning tasks and responsibilities within a organizational development. Its goal is to ensure the production of high-quality software which meets the needs of its end-users, within a predictable schedule and budget. The RUP captures many of the best practices in modern software development in a form that can be tailorable for a wide range of projects and organizations.

In Catalysis, component-based development is a development approach in which all artifacts — from the executable code to interface specifications, architectures, and business models, scaling from complete applications and systems down to individual components can be built by assembling, adapting, and “wiring” together existing components into a variety of different configurations. Catalysis provides strong support for: modeling business components, interface centric design, component architectures and the reuse of design and model components, in addition to code.

This article presents the construction of a framework in an N-tier architecture using the Catalysis approach and the application of this framework in a Values Quotation component. The decision to use the Catalysis approach was made because Catalysis possesses well defined techniques to integrate components and frameworks (see Section 4.6). Despite Catalysis being sufficiently complete, in this work we used RUP [9] as the development process of the Values Quotation component because RUP applies the iterative process during all the development (i.e. conception, elaboration, construction and transition).

This paper is organized as follows. Section 2 presents the analysis of the values quotation domain. The software architecture proposed in this work for the development of components is shown in Section 3. This architecture includes technologies such as distributed objects, frameworks, and patterns. Section 4 presents the design and prototyping of the Values Quotation component. In Section 5 the validation of the component development is presented and finally, the conclusions of the work are presented in Section 6.

2 Analysis of the Values Quotation Domain

In this section the concepts related to the values quotation domain used as a study case for the construction of the framework and the Values Quotation component are presented. Through this component, it will be possible to make the product price surveys with suppliers to calculate the present value of each quoted product based on prices, conditions, and ways of payment offered by suppliers. It will also be possible to analyze the quotations and to negotiate prices and payment conditions with the suppliers.

In order to utilize the component the user will have to register suppliers, products, and groups of products. The products to be quoted can be imported from a text file or informed by the operator of the system. In the importation process, the user will have to select the layout that describes the text file to allow the system to recognize all the fields.

The quotation will be composed of products of the same group; suppliers can be related to one or more groups of products. For each quotation, a set of Suppliers related to groups of products will be selected. These suppliers will receive a quotation request by fax or e-mail. The request will inform the page address containing the form to be filled with the quoted prices for the products. This form, will automatically update the quoted prices in the database when submitted to the component. Based on the prices and other information sent by the suppliers, the system will generate a spreadsheet including the best price for each product, the best price of each quotation group, and the value of the last purchase for each product. The purchaser will be able to generate another spreadsheet in order to negotiate with one or more suppliers. After that, the purchaser will select a product of each supplier or group to place the order.

Based on the specifications presented in this section, it is possible to understand the proposed domain. In the following section, the software architecture, which considers the technologies used in the construction of the framework for the Values Quotation domain, is presented.

3 Software Architecture

The systematic reuse of all aspects resulting from the development process, such as ideas, concepts, requirements, and designs acquired or constructed during all the phases of the software process, is an important principle that should be incorporated to software development methods.

In this way, research in the field of software architecture, software patterns, and frameworks is being carried out through and evaluated in order to allow the reuse and recycling of existing solutions, in the form of ideas, concepts, projects, and similar products applicable to the software development process, in different levels of abstraction.

In this section, concepts of architectural styles, software patterns and frameworks and the development process used in this work are presented.

3.1 Patterns and Frameworks

Patterns can be used to describe the purpose of a framework and, consequently, to allow its use by the application programmers who do not need to know how it works internally. Patterns can even show many of the inserted details of design in one framework. Thus, as pointed out by [8], patterns can be used to document frameworks. The documentation of a framework describes its purpose, how to use it and its detailed design. The use of patterns for documentation meets these purposes.

A framework can be called a “component” when it has the extensibility feature. It is important to notice that the application of a framework can improve a component, if its extensibility capacity has been fully explored.

3.2 The Architecture Definition of a Component

The architecture of a component is an abstraction of a system which describes the design structure and relationships, as well as administrative rules, or principles which are, or could be, used by many designs [4].

This architecture uses three types of patterns: horizontal (specification of the component), vertical (implementation of the component) and connectors (objects of the component). The architecture diagram of the component specification contains its specifications and interfaces. The architecture implementation of the component shows the dependence existing between the components and the architecture object of the component specifies the component instances which will be accessed [2].

In order to develop software products with quality, low cost, and productivity, an interoperable component was implemented. This could be possible through the use of the multi-tier technique. The first thing that must be defined when using this technique is the architectural style of the software. In our work the adopted style was the use of packages with their correlated dependence. These packages represent the client, the business rules, and the database tier respectively. In the client tier, the graphical interfaces are defined. The use of distributed object technologies for the implementation of the business rules tier allows the choice between different solutions for the other tier. This enables us to deal which technology changes. Finally, in the last tier, a database manager system is implemented.

3.3 The Development Process

The development process was based on RUP with the utilization of Catalysis specific techniques for the construction of components and frameworks. RUP is a software engineering process, which provides a disciplined approach to assigning tasks and responsibilities within a development organization. Its goal is to ensure the production of high-quality software that meets the needs of its end users within a predictable schedule and budget. The figure 1 shows the overall architecture of the RUP. The RUP has two dimensions: 1) the horizontal axis represents time and shows the lifecycle aspects of the process as it unfolds; 2) the vertical axis represents disciplines, which group activities logically by nature.

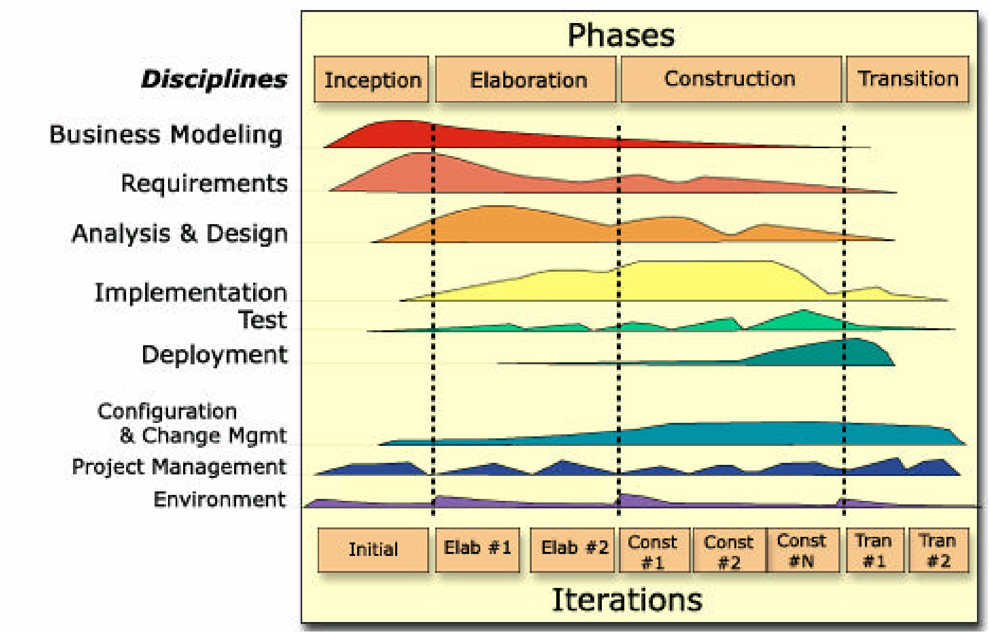


Figure 1. Rational Unified Process [9].

The Catalysis development process, allows to develop artifacts in a structured, and evolutionary way. The process is nonlinear, iterative, with emphasis on architecture and consistency in the frameworks and components developments.

For the design of the Values Quotation Framework described in section 4, we decided to use the Catalysis approach because it applies well defined techniques to integrate components and frameworks (see Sections 4.3, 4.4 and 4.5) and the UML components in the section 4.6. As mentioned, despite Catalysis being sufficiently complete, we used RUP [9] as the development process of the Values Quotation component in this work because RUP applies the iterative process during all the development (i.e., conception, elaboration, construction and transition). All the disciplines defined by RUP (see figure 1), with the exception of Business Modeling and Requirements, were used in this work. The reason for not to considering the disciplines above mentioned was that they were already documented when the work started. Although the development and support disciplines were used in this work we emphasized the Design Analysis through the use of an object-oriented technology. Thus, for specification, visualization, construction and documentation processes the UML modeling language was used.

In this way, only the parts of the project composed by the use case diagram at project level, the framework model for CORBA architecture, the component diagram, the application of the framework model, and the application's unfolding for the development of the Values Quotation component are presented.

Therefore, it can be said that the RUP complement the Catalysis approach and UML Components in relation to the development process. Besides of this, Catalysis also provides the iterative process and use almost all the UML diagrams.

4 Design of the Values Quotation Framework

The standard used for the design of the Values Quotation framework was the UML language [1]. The CASE tool used to specify the diagrams was Rational Rose [13]. The implementation of the component prototype was made using the Java programming language [11].

4.1 The use of Frameworks in CBD

Frameworks and components are related, as it is possible to use frameworks for components construction, and components for frameworks construction [6].

The methodology used in the development of the Values Quotation framework was the Catalysis approach for its consistency in the frameworks and components development, mainly for the fact that the approach utilizes UML as standard in its development.

The choice of the Catalysis approach resulted from the comparison between this approach with some other methods for frameworks development [6]. The compared methods were the “Example-Driven Project” [8], the “Project Driven by Hot Spot” [12] and the “Taligent Approach” [14]. Among these, the frameworks development process, proposed by the Taligent boarding [14], is interesting, intuitive, and emphasizes the idea of constructing small frameworks. However, it is not supported by a consistent notation. In this way, well defined processes, such as the recently published, Catalysis Approach [4], the Unified Process Model [7], and the UML Components [2] are more complete. Both processes are supported by UML notation [1]. The Catalysis method also allows the reuse of patterns and the extensibility of packages diagrams, besides allowing the interaction between diagrams (see Figure 3). Another Catalysis feature is its directed approach to behavioral characteristics of the domain, which makes a bigger detailing of the actions possible.

4.2 Use Case Specification

Use case diagrams graphically depict the behavior of the system (use cases). These diagrams present a high level view of how the system is used as viewed from an outsider’s (actor’s) perspective. A use case diagram may depict all or some of the use cases of a system. The use case can be described as a specific way of using the system from a user’s (actor’s) perspective. Figure 2 shows the use case diagram of the Values Quotation component. In this paper, in order to give an example of use case implementation, will be shown the “ControlSupplier” use case.

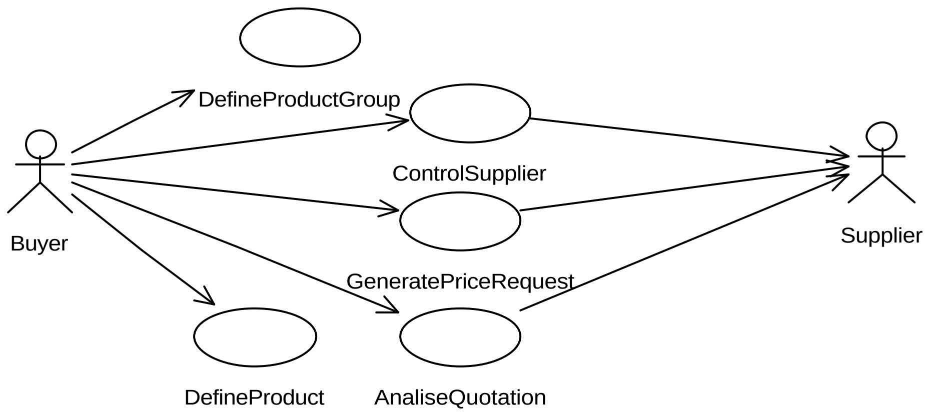


Figure 2. Use Case diagram of the Values Quotation component.

4.3 The Framework Model

The framework model using CORBA patterns is presented in figure 3. The Software Engineer is responsible for the definition of the architecture. The Distributed Object Designer has an inheritance with the Software Engineer, being able to define the structure of the framework CORBA. The Business Designer, however, is responsible for the definition of the business rules and its graphical interfaces. The functionalities represented through ellipses have an arrow of dependence with the package of the model framework. The packages represented with stereotypes <<Framework>> represent applications reused for the domain previously mentioned.

The framework model types are: <FrISupplierMgr>, <FrISupplierMgrOperations>, <FrSupplierMgrPOA>, <FrISupplier>, <FrISupplierOperations> and <FrSupplierPOA>. The last one has the stream of the control execution and the definition of the framework behavior. These types can be substituted in the application of this model. This application is similar to the extensibility of the classes used in the Object Oriented Approach.

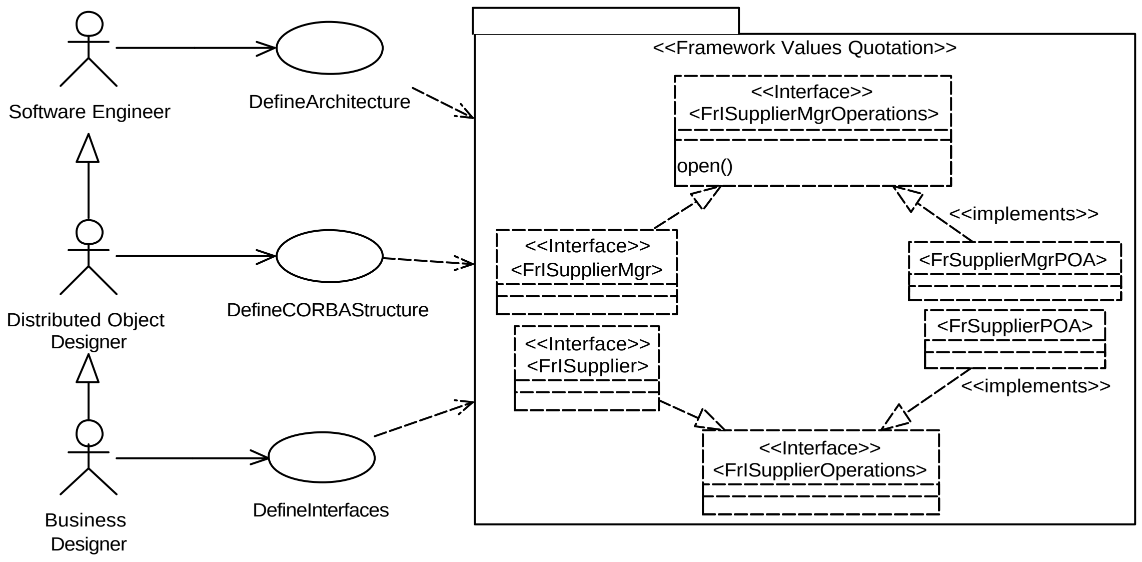


Figure 3. Framework Model of the Values Quotation component.

4.4 Application of the Values Quotation Framework Model

CORBA is a framework of distributed objects proposed by OMG (Object Management Group) [11]. The essence of the CORBA architecture is ORB (Object Request Broker). ORB acts as the broker on which objects interacts with other objects (local or remote). The object interacts with ORB through a POA (Portable Object Adapter) or through the interface ORB.

The CORBA Framework Model, which results from the application of the Framework Model defined in section 4.3, is presented in this section. The framework application shown in figure 4 is an importation of framework models with substitutions. Substitutions are usually represented by arrows of dependences. When substituting the types of the framework model in the Values Quotation component, the following result was obtained: the types <FrISupplierMgr>, <FrISupplierMgrOperations>, <FrSupplierMgrPOA>, <FrISupplier>, <FrISupplierOperations> and <FrSupplierPOA> were substituted by the classes “ISupplierMgr”, “ISupplierMgrOperations”, “SupplierMgrPOA”, “ISupplier”, “ISupplierOperations” and “SupplierPOA”, respectively.

The next step of the Catalysis approach is to implement the component. This implementation process, called “unfolding” by Catalysis, is presented in the next section.

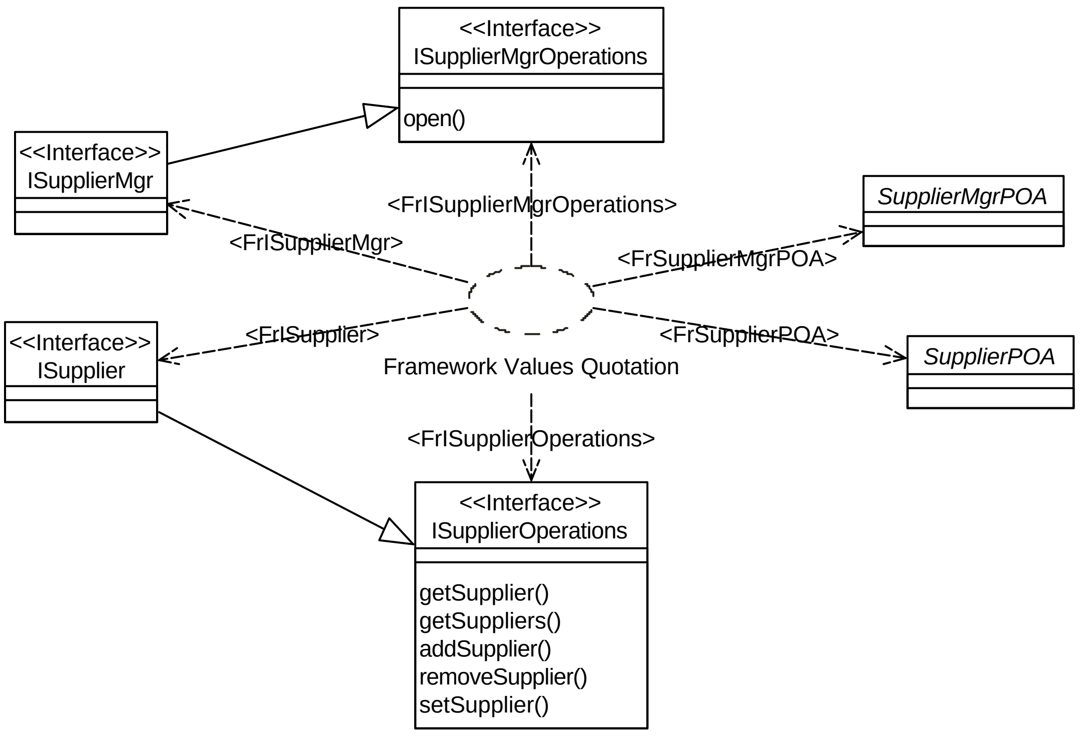


Figure 4. Application of the Values Quotation Framework Model.

4.5 Unfolding of the Component's Application based on the Framework Model

The framework application can be unfolded to get the complete model of the application. The unfolding, presented in figure 5, is the representation of the importation result, including the substitutions in the importation context of the package with the appropriate substituted elements. The Framework is unfolded to supply a version of the content, which is specialized, based on the substitutions.

The Values Quotation component was implemented using the CORBA standard. The development of the component was done using the VisiBroker tool [15] and the Java programming language [11]. The first step was to specify the interfaces for each object of the CORBA server. The specification was made with IDL (Interface Definition Language) which is an independent language used to specify the operations of an object and how these operations could be invoked. Once the CORBA interfaces are definied, it is necessary to convert them into one determined programming language. In the case of the Values Quotation component the chosen language was Java.

In CORBA, client applications are implemented through the execution of the following steps: 1) starting ORB services; 2) connecting the client application with the objects of the server and 3) calling the methods of these objects. The ORB is a bus that connects the CORBA client application with server objects.

For the implementation of the server code (i.e. business rules) the first thing that must be done is to derivate SupplierMgrPOA and SupplierPOA from POA abstract class. The derivated classes are implemented by the classes SupplierMgrImpl and SupplierImpl. Before the implementation of the SupplierMgrPOA and SupplierPOA, their related interface methods (i.e. ISupplierMgrOperations and ISupplierOperations) must be implemented. The SupplierMgrImpl class has a relationship with the SupplierSrv class which has the responsibility to activate the ORB server. The main method of the SupplierSrv class instantiates the servant objects (i.e. SupplierMgrImpl and SupplierImpl), and then activates these objects, as well as the POA manager, and the POA object.

The SupplierClie class acts as an interface between the user interface and the SupplierMgr component, called Glue Components (see Section 4.6).

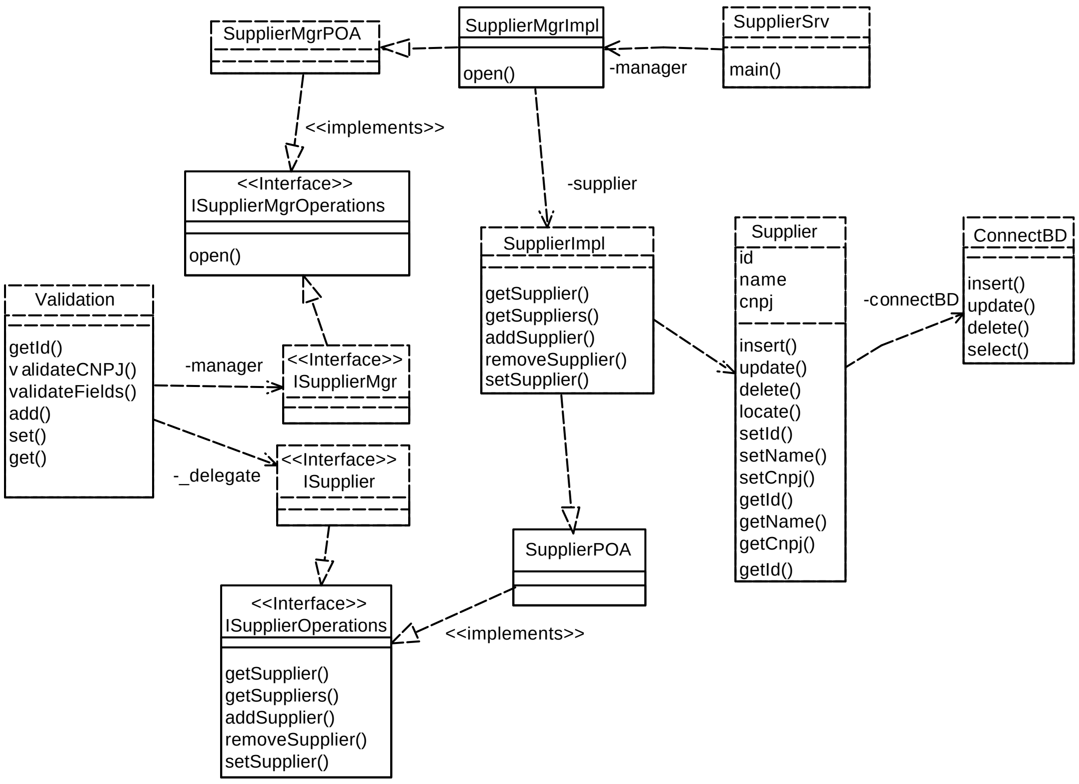


Figure 5. Unfolding of the Component's Application based on the Framework Model.

4.6 Component Diagram

A component can become related to other components through its interfaces. The implementation of a component is selected at design time (i.e., coding time, compilation, plug-in) or run time (i.e., dynamic plug-in, execution, and reflexive) [4]. The connection between heterogeneous components can be done through the development of another component call "Glue". Glue is an intermediary component which allows communication between the connected components [4].

Components cannot be modified by users; therefore, components must be projected to allow communication with other components. This can be done by messages swapping between its instances. This swapping of messages is possible through well-defined interfaces. The component diagram shows a high-level view of this architecture [16].

From the framework model, a component diagram was constructed for a better interpretation of the interfaces and the connections between the components (figure 6).

The Java programming language and CORBA were used to implement the component and the interfaces previously mentioned. Java explicitly represents the definition of classes, interfaces, and methods. This feature is called by the Catalysis approach as reflexive.

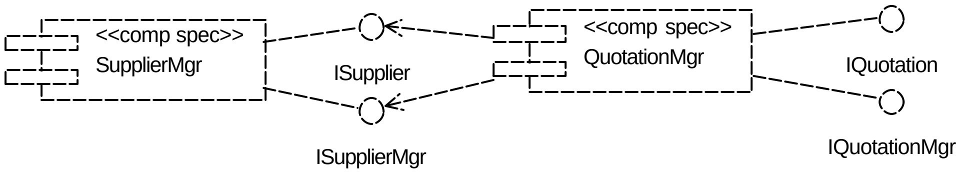


Figure 6. Diagram of the Values Quotation component.

Figure 7 presents the implementation diagram (application) of the SupplierMgr component which implements the stereotyped component <<comp spec>> and its ISupplierMgr and ISupplier interfaces, which are CORBA ones.

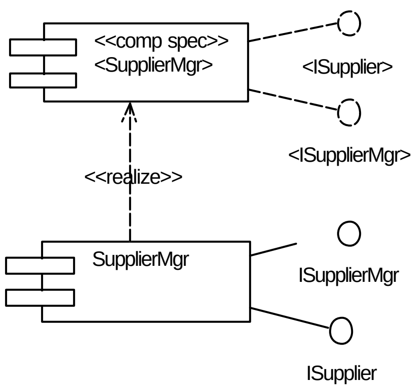
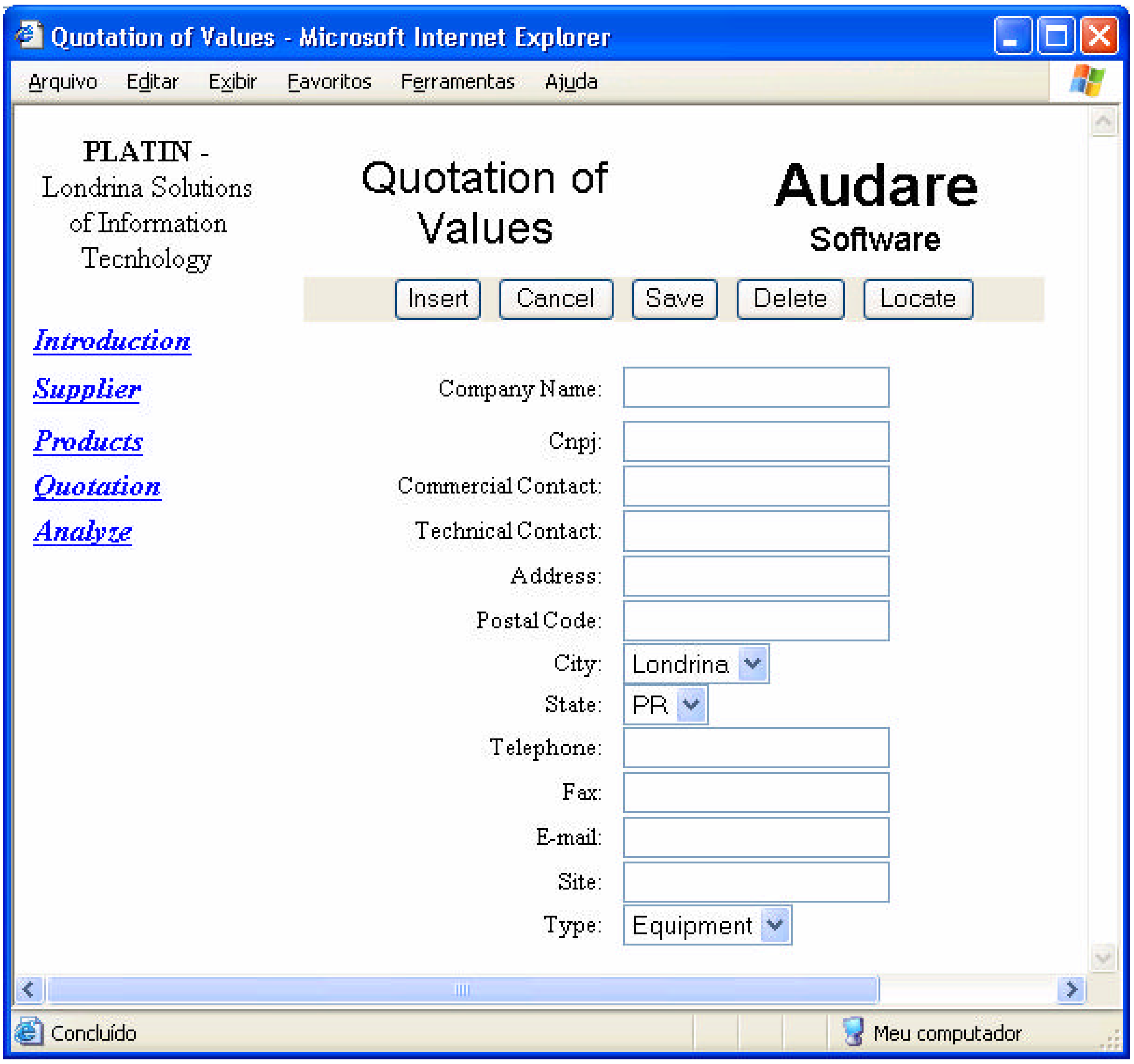


Figure 7. Application of the SupplierMgr Component.

5 Validation of the Component Development

To validate the development of the component with frameworks, a prototype, in Java language, was developed. For the data storage, the Oracle 8 Database Manager System was used [10]. Figure 8 shows one of the interfaces resulting from the implementation of the Values Quotation application that uses the component. The application contains graphical interfaces in which the Buyer executes actions related to the quotation price task. These actions were defined during the use case specification (see Section 4.2). Figure 8 presents the suppliers form, developed for the web, which allows to include, update, exclude and locate a specific supplier. These functions can be done using the buttons shown in the upper part of the form.

The documentation of the Values Quotation component was made using the design standard proposed by Gamma [5]. This standard was chosen due to its proximity with the design implementation.



PLATIN - Londrina Solutions of Information Tecnhology

Quotation of Values

Audare Software

Introduction
Supplier
Products
Quotation
Analyze

Company Name:	
Cnpj:	
Commercial Contact:	
Technical Contact:	
Address:	
Postal Code:	
City:	Londrina
State:	PR
Telephone:	
Fax:	
E-mail:	
Site:	
Type:	Equipment

Figure 8. Graphical interface of the Values Quotation application.

6 Conclusions

One contribution of this study was to show that the Catalysis approach and RUP can be used together and that they have peculiarities which complement each other, allowing a higher quality of the final product in the construction of components and frameworks. These conclusions were achieved through the construction of the Values Quotation framework, presented in this paper. Another contribution was the construction of frameworks for CORBA architecture, at design level, for domains where they can potentially be reused. The study made in the domain of Values Quotation served as a basis for the development of other frameworks and components in diverse areas. Besides the construction of the framework for CORBA architecture, it was observed that several frameworks could be defined for applications being developed.

The use of the Catalysis approach had its emphasis on the development of components and frameworks and for the consistency in the definition of its documentation. The use of CASE tools in the components development is essential. All the modeling was made using the Rational Rose CASE tool, for its capacity to update the modeling and the source code with ease.

The UML Components contemplates RUP, Catalysis and UML itself in its process of development specification based on components that allowed a greater consistency and clarity in the definition of component architecture.

CORBA standard and Java language were adopted for being multi-platforms. The software multi-tiers architecture makes the user interface lighter, and with the use of CORBA in the business rule tier this interface can be developed in different programming languages, such as those for dedicated applications as well as for Web applications.

The application of CORBA services offers the possibility to achieve higher quality levels in CBD. Another fact to be considered is the practical application of the existing design patterns, in this way magnifying the reuse level.

Acknowledgements

Sergio A. Tanaka and Ruy T. Nishimura would like to thank UNIFIL for the given structure of the Software Engineering Laboratory and ADETEC / PLATIN for the infrastructure.

References

[1] Booch, G., Rumbaugh, J. and Jacobson, I. UML – User's Guide. São Paulo: Ed. Campus. 2000.

[2] Cheesman, J. and Daniels, J. UML Components – A Simple Process for Specifying Component-Based Software. Addison-Wesley Publishing Company. 2001.

[3] Chung, P. E., Huang, Y. and Yajnik, S. DCOM and CORBA Side by Side, Step by Step, and Layer by Layer. Available in: http://www.cs.wustl.edu/~schmidt/submit/Paper.html. 2001.

[4] D'Souza, D. and Wills, A. Objects, Components and Frameworks with UML – The Catalysis Approach. Addison-Wesley Publishing Company. 1999.

[5] Gamma, E. et al. Design Patterns. Elements of Reusable Object-Oriented Software. Addison-Wesley Publishing Company. 1995.

[6] Gimenes, I. M. S., Tanaka, S. A. and Palazzo, J. M. O. An Object Oriented Framework for Task Scheduling. In: Proceedings of TOOLS EUROPE 2000. France. IEEE Computer Society Press. 2000.

[7] Jacobson, I., Booch, G. and Rumbaugh, J. The Unified Software Development Process. USA: Addison-Wesley Publishing Company. 1998.

[8] Johnson, R. Documenting Frameworks Using Patterns. SIGPLAN NOTICES. Vol. 27, No. 10, (October 1992).

[9] Kruchten, P. The Rational Unified Process: an Introduction. Second Edition. USA: Addison-Wesley Publishing Company. 2000.

[10] ORACLE Corporation. Oracle Database Family. Available in: http://www.oracle.com/ip/deploy/database /8i/index.html?persed.html. 2001.

[11] Orfali, R. et al. Client/Server Programming with JAVA and CORBA. Willey. 1998.

[12] Pree, W. Design Patterns for Object-Oriented Software Development. USA: Addison-Wesley Publishing Company. 1995.

[13] RATIONAL Software Corporation. RATIONAL Documentation. Available in: http://www.rational.com /support/ documentation/index.jsp. 2001.

[14] Taligent Inc., Apple Computer Inc., IBM Corporation and Hewlett-Packard Company. TALIGENT White Paper. Building Object-Oriented Frameworks. 1996. Available in: http://www.taligent.com.

[15] Borland Software Corporation. Borland VisiBroker – Programmer´s Guide. Available in: http://www.borland.com. 2002.

[16] Wills, A. C. Designing Components Kit and Architectures with Catalysis. TriReme International Ltd. 1998. Available in: http://www.trireme.com. 2001.