Designing Business Process-Based Software: A Detailed Example

White paper, March 1999

ENSEMSBLE SYSTEMS INC.
545-5600 Parkwood Way
Richmond, BC, Canada, V6V 2M2
Tel: (604) 231.9510, Fax: (604) 231.9545
www.ensemble-systems.com

Ensemble Systems Inc.

Designing Business Process-Based Software: A Detailed Example

Contents

Ensemble Systems Inc.

Designing Business Process-Based Software: A Detailed Example

White paper, March 1999

Abstract

This document presents a pragmatic strategy for the development and management of a business system project. It has been designed to provide guidance to the business process analyst as well as to the software architect. From requirements analysis to software implementation the approach is illustrated with detailed examples. Several different software tools are used to accomplish the project. While the usage of all the tools is addressed, the paper focuses on the role played by Ensemble's Rose Business Process Link modeling software.

Keywords: business process modeling, UML, activity diagrams, round-trip engineering.

1. Introduction

Understanding the construction and the functioning of a business process is a necessary precondition for any business transformation project. For this reason, in the past decades the problem of modeling business processes was addressed by various approaches, modeling techniques, and tools (e.g. Object-Oriented Analysis and Design, the Object Management's Group's Unified Modeling Language (UML), the Workflow Management Coalition's Workflow Reference Model (WRM), Use Case formalism, Business Process Reengineering (BPR) techniques, CASE tools, etc).

All of these methods and tools have something in common: they are trying to solve a critical problem, one of communication. People with very different backgrounds take part in the development of software solutions supporting business processes: business managers, workflow specialists, software consultants, designers, and software developers. It is necessary for all parties to understand what the business process model is communicating, during the entire lifecycle of the project. Having a common language, understanding the structure and behavior of a business process, formulating the requirements in an unambiguous manner, mapping the business requirements to the software components, interpreting the business rules correctly, presenting the software solution to the users of the future system, all are communication-related factors which determine the project's success or failure.

The UML 1.1 defines a standard notation for object-oriented systems that was officially adopted in 1997 by the Object Management Group (OMG). Since then it has gained a wide acceptance by various professional communities. The UML is defined as a general-purpose visual modeling language but includes extensions for both Business Modeling and Software Development Processes. These extensions will also be included in UML 1.3, which is expected to be completed and adopted in 1999 [UML 1.3].

Ensemble's Rose Business Process Link (RBPL) is a tool whose purpose is to enhance the communication between business and software professionals. RBPL answers the challenge of modeling a business process in an accurate but simple manner, and turning it into a technical software specification that is both precise and easy to understand. It is an analysis and design tool that allows representing structural and functional aspects of both business processes and software systems. The place of RBPL in the software engineering process is close to the business domain. It helps the designer to accelerate the mapping process between the business terminology and workflow concepts, and the rich but sometimes complicated notation of the UML standard. RBPL provides Rational Rose with a starting model populated with actors, classes, and use cases.

2. Runners Inc. Project

A typical scenario of business process transformation was chosen to illustrate the use of RBPL. As presented below (§ 2.1), the Runners Inc. project requires developing a software system for an existing business process that is functioning in a traditional way using paper support only. The different phases of the project are depicted in Fig. 1. The designer starts building the business process model by using RBPL. The project requirements are analyzed and translated into model elements: business activities, actors, activity diagrams, etc. Involving both business and software teams, the model is updated in an iterative manner with other useful process details. When the model of the existing business process is completed the designer steps into the implementation design. New elements are added to the RBPL model: use cases, packages, classes, other activity diagrams, etc. Finally, RBPL produces a set of deliverables: the RBPL model, the Rose model starting point, GUI views, and documentation in Microsoft Word format. Once the main actors, classes and use cases have been identified and exported to the Rose model, the project continues using other software tools such as Rational Rose (Rational Software), Rose JBuilder Link (Ensemble Systems Inc.), and JBuilder2

(Borland), iteratively via round-trip engineering (RTE). The deliverables of these activities are the Rose model and documentation, the JBuilder project, and the Java classes. MS Access (Microsoft) is used to create and maintain the Runners Inc. database, and the Java Runtime Environment (Sun) is used to execute the application.

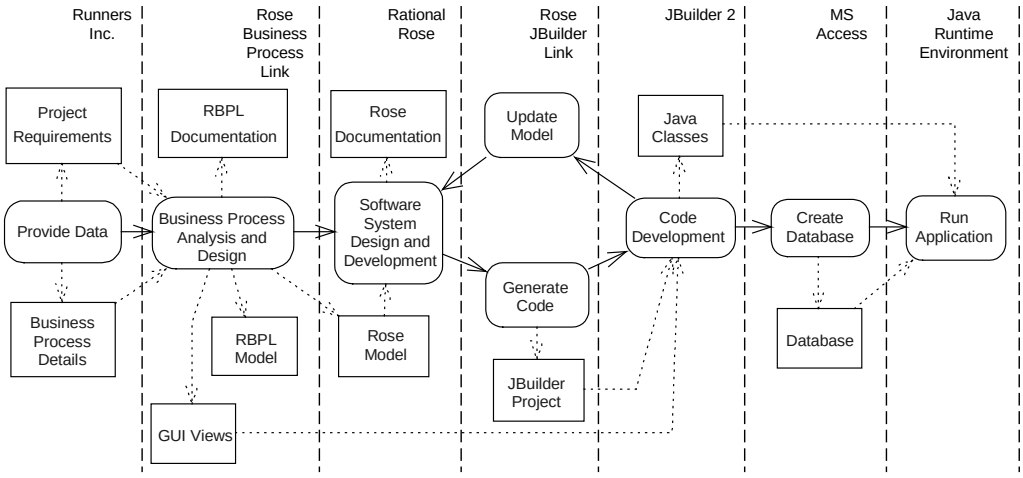

Figure 1. "Runners Inc. Project" activity diagram

2.1 Runners Inc. Requirements

As it is often the case, the requirements are presented in plain text. They are the following:

> **Context:** *Runners Inc. is a distributor of shoes for jogging enthusiasts. Orders are received by telephone, and they are shipped by mail. They carry two styles of shoe, "Trotter" and "Sprinter". Each shoe is available in Men's sizes 6 through 12, and Ladies' 6 through 12. They normally carry sufficient inventory to fill all orders. They currently handle all orders on paper, but have now grown large enough that some degree of automation is required.*
>
> ***Objective:*** *Through a short brainstorming session, it is determined that the most important problem to address is that orders are being delayed or lost due to misplaced paperwork. Therefore, support is required for order entry and for shipping, with checks to ensure that all orders are filled in a timely fashion. Other areas, such as support for inventory management, integration with the accounting system, etc., can be addressed at a later date.*
>
> **Constraints:** *The system should use data storage techniques that support scalability and are easily extended and maintained. However, short-term volumes are expected to remain relatively low (10 orders per day).*

3. Rose Business Process Link

Business processes are complex. The modeling task consists of capturing the useful information about the business process. As the business process is often rich and sophisticated, one might expect the model to be quite complex. The task of RBPL is to make this complexity manageable. For this purpose, RBPL uses four main categories of elements to model a business process: a glossary, feature views, user interface views, and activity diagrams. The usage and the advantages of these model elements are described below.

Because of the complexity of the business processes, it is almost impossible to capture their entire behavior in a single diagram. To overcome this problem, in the UML there are five distinct types of diagrams defined: *use case diagrams*, *activity diagrams*, *statechart diagrams*, *sequence diagrams*, and *collaboration diagrams*. *Use case diagrams* are mainly used to model the context of the system and the software requirements. *Activity diagrams* are used for workflow and operation modeling [Booch 98].

Use cases, as defined by the UML standard, are system-oriented. They focus on interactions between humans and software systems, or between sub-systems, as opposite to interactions between humans, or humans and organizations, which is often the case in business processes [Hurlbut 98]. Business processes are also characterized by flows of goal-oriented business activities. They are activity-based processes, i.e. processes in which the activities have deliverables or other business objects as post-conditions [Berkem 98], [Hruby 97].

RBPL emphasizes the use of *activity diagrams* for business process modeling, and suggests the use of *use cases* only where a software system is involved. However, *activity diagrams* will be also used to describe the software system behavior. If one finds it useful to make a distinction between *business activities* and *actions*, and *software activities* and *actions*, the first may be stereotyped with the stereotypes «*business*» and «*business action*» respectively.

The standard UML activity diagram notation includes: initial and final states, activity states, object flow states, branch and merge, fork and join, input and output events, transitions and object flows, and swimlanes [Booch 98], [Rumbaugh 98]. Besides these, RBPL allows one to represent other UML icons on an activity diagram: actors, classes, packages, use cases, dependencies, and views of GUI designs (see the Appendix).

RBPL adds an extension to the UML standard called *feature*, represented as a triangle. A *feature* is meant to represent a single, task-related capability of a system (a requirement whose context is not specified yet). The context in which the *feature* is used depends on its association with an *activity diagram* or *use case* [RBPL 99].

3.1 "Project Requirements" Activity Diagram

During the requirements analysis, the model of the business process starts being built incrementally. At this stage, different entities of the business process are identified and defined. The requirements of the Runners Inc. project (§ 2.1) are represented using a business-oriented activity diagram in Fig. 2. In a workflow in which a number of different people, functional areas, and subsystems are involved, it is sometimes difficult to keep track of who is responsible for what. In order to organize the responsibilities inside Runners Inc., a first step may be to partition the activity graph in four main categories of model elements using swimlanes: actors, *"business"* activities, *"software"* activities, and business objects.

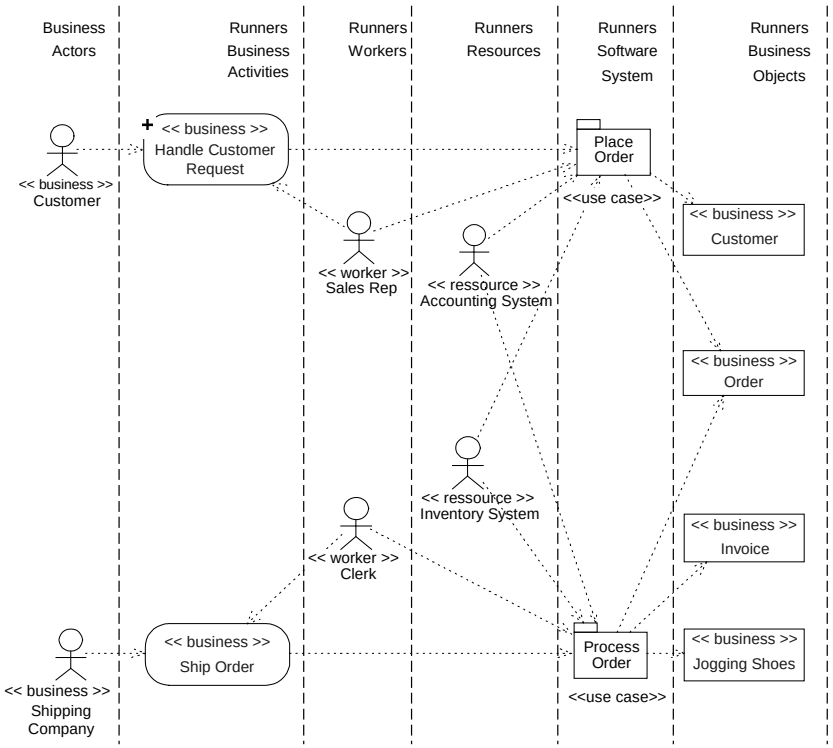

Figure 2. "Business Requirements" activity diagram

Three types of actors are then identified and represented as swimlanes too: "Business Actors", "Runners Workers", and "Runners Resources". The second step is to define the actors:

Business Actors:

- *Customer* – a person who orders products from Runners Inc.
- *Shipping Company* – DHL, FedEx, UPS, and so on.

Runners Workers:

- *Clerk* – an employee of Runners Inc. who retrieves new orders, packages, labels, and ships the products.
- *Sales Rep* – an employee of Runners Inc. who processes customer requests.

Runners Resources:

- *Accounting System* – software that keeps the accounts (not addressed yet).
- *Inventory System* – software that keeps the inventory (not addressed yet).

There are two main business activities that appear clearly from the requirements: one is "Handle Customer Request" during which orders are received by phone, and the other is to "Ship Order" during which orders and products are shipped by mail. The goal of the project is to support these two business activities using a software system. This is represented in the diagram using two use case packages: "Place Order" and "Process Order. Four business objects are also defined: "Customer", "Order", "Invoice", "Jogging Shoes".

During requirements analysis it is very important to use a terminology accepted by all the project teams. RBPL provides a Glossary where technical terms from both business and software domains may be defined. These entries are called *Glossary Items* and they can be structured in *Glossary Packages* as follows:

Acronyms:

- *c.i.f* – abbreviation for: cost, insurance & freight (includes insurance and shipping).

Business Terms:

- *cold call* – to telephone a prospect without previous contact.

Software Terms:

- *JDBC-ODBC* – a bridge that provides access via ODBC drivers to databases from Java programs.

3.2 "Business Process" Activity Diagram

The requirements (§ 2.1) specify that only a part of the whole Runners Inc. business process is the subject of the project, and that the other business activities of the company might be addressed in the future. The scope of the project is translated in the RBPL model as an activity diagram (Fig. 3). This is an important view because it represents the top diagram of the Runners Inc. business process that shows the context in which the present project takes place.

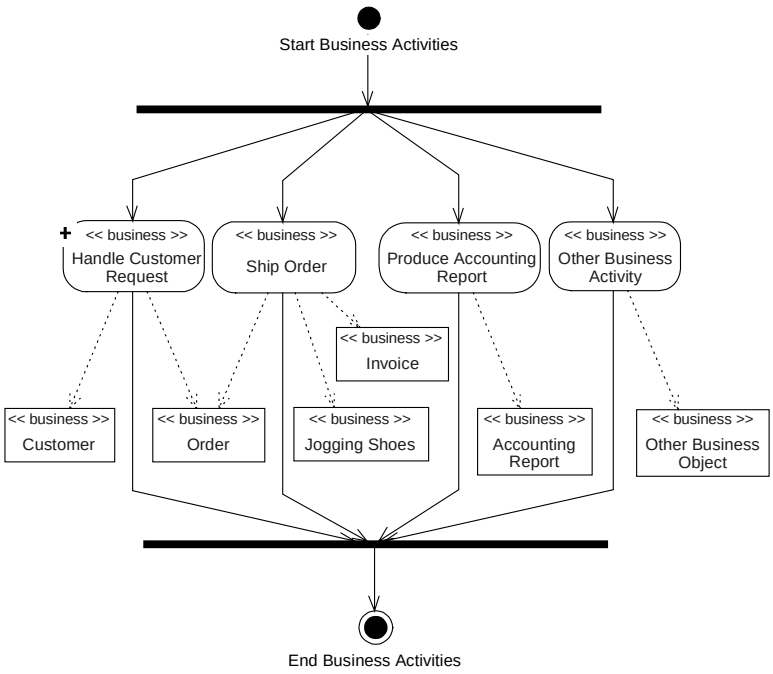

Figure 3. "Business Process" activity diagram

This diagram bounds the current development and will be used as a reference for further developments. If the business process is reengineered (which is not the case in this example), this activity diagram would then be modified to reflect the changes.

3.3. "Handle Customer Request" Activity Diagram

Plain text representation of use cases can be written in a formal or less formal style. A useful technique used to structure and document use cases is that of templates [Cockburn 98], [Schneider 98]. Because there are similarities in modeling

software systems using use cases and business processes using activity diagrams, the use of templates can easily be extended to activity diagrams. RBPL provides the facility to link such text file documents to the activity diagram specification in the RBPL model. By supporting hyperlinks, RBPL allows the user to complete the business process model with information in various formats from the local machine or via URL. An example of such a file that details a business activity using a template is given in Fig. 4. The "Handle Customer Request" activity takes place by phone between two actors: the "Customer" and the "Sales Rep":

BUSINESS ACTIVITY: Handle Customer Request

Goal:	describes the process of handling a customer call.
Actors:	Sales Rep Customer
Preconditions:	the Sales Rep is logged into the system and waiting for a call.
Trigger Condition:	the Customer calls.
Success End Condition:	the call ends.
Failed End Condition:	none
Post-condition:	the Sales Rep waits for another call.

PRIMARY SCENARIO

1. The Sales Rep welcomes the Customer.
2. While the customer has a request:
    2.1. The Sales Rep identifies the requested service;
    2.2. The Sales Rep performs the requested business activity:
        a) take a new Order;
        b) get the status of an existing Order;
        c) cancel an existing Order;
        d) solve a complaint about an existing Order;
        e) answer to a special Customer demand;
        f) present the products.

   end
3. The Sales Rep thanks the Customer.
4. The business activity ends.

SECONDARY SCENARIOS

1. Wrong number, no request.

RELATED INFORMATION

Priority:	top
Performance Target:	20 minutes
Frequency:	15/day

SCHEDULE

Due Date:	version 1.0

Figure 4. "Handle Customer Request" specification

The "Handle Customer Request" business activity is also represented graphically in Fig. 5. There is a main *"do while"* loop and a decision block with guarded output transitions that identify the requested service. The whole business activity is decomposed into activities and actions. Those business activities that are in turn detailed using activity diagrams are marked with a "+" sign. The user can navigate up and down the activity diagram hierarchy. Furthermore, dynamic visualizations of the activity diagram are possible using the storyboard mechanism provided by RBPL.

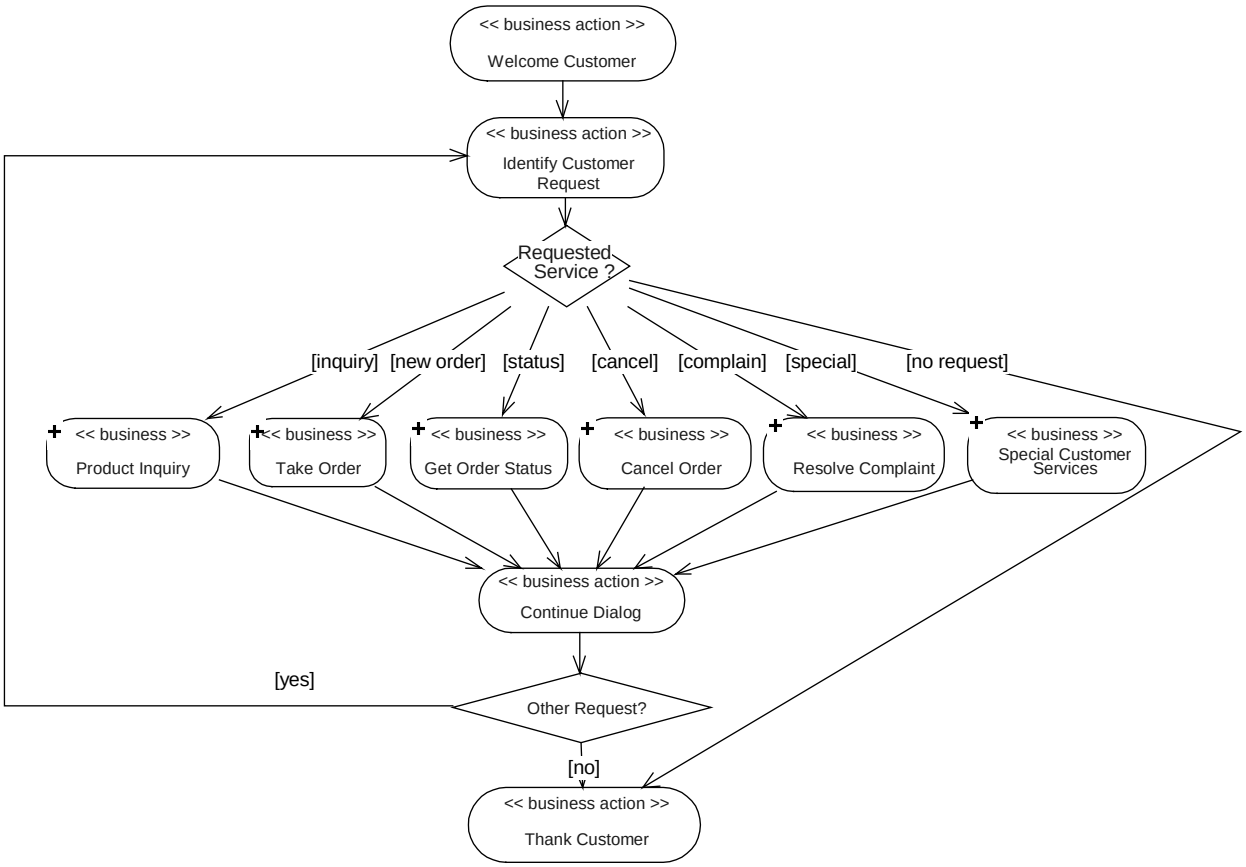

Figure 5. "Handle Customer Request" activity diagram

3.4. "Take Order" Activity Diagram

The "Take Order" business activity from Fig. 5 is decomposed in Fig. 6. The first goal of this new activity diagram is to identify the use cases of the Runners Inc. software system, represented in the rightmost swimlane, using the workflow of the business process, represented in the leftmost swimlane. In the middle swimlane, the "Sales Rep" actor is represented twice only for graphic considerations. The discussion takes place between the "Sales Rep" and the "Customer", who is omitted from the diagram. This business activity is presented the level of detail where it is self-explanatory.

The "Sales Rep" is interacting with the software system via graphical user interfaces (GUI). The second goal of the "Take Order" activity diagram is to design the GUI views of the future software system at this early stage of the project. RBPL provides for this purpose a graphical library which contains commonly used symbols such as: button, control, edit box, static control text, check box, list box, radio button, combo box, horizontal and vertical scroll bars, etc.

Based on the use cases previously identified and on the data manually collected by the Sales Rep during the phone call, four interfaces are ergonomically designed to answer to the end-user needs. Two of them are presented in Fig. 7. The "Add New Customer" dialog is opened from the "Take Order Main Dialog" using the "New Customer" button, and is closed when its "Quit" button is pushed.

Once designed, the GUI views become part of the RBPL model as "Dialog" elements, and part of the software system requirements. These views will be used to detail the use cases and to assist in implementation.

A *feature* is typically used when a capability or need is identified, but not yet described as a *use case*. This is the case in Fig. 6 with the "Phonetic Matching" feature attached to the "Search Customer by Business Name" use case. This is a functionality that should be developed in the future.

For the sake of brevity, from this point on, only the "Add New Customer" business activity will be detailed and followed to the end of the project.

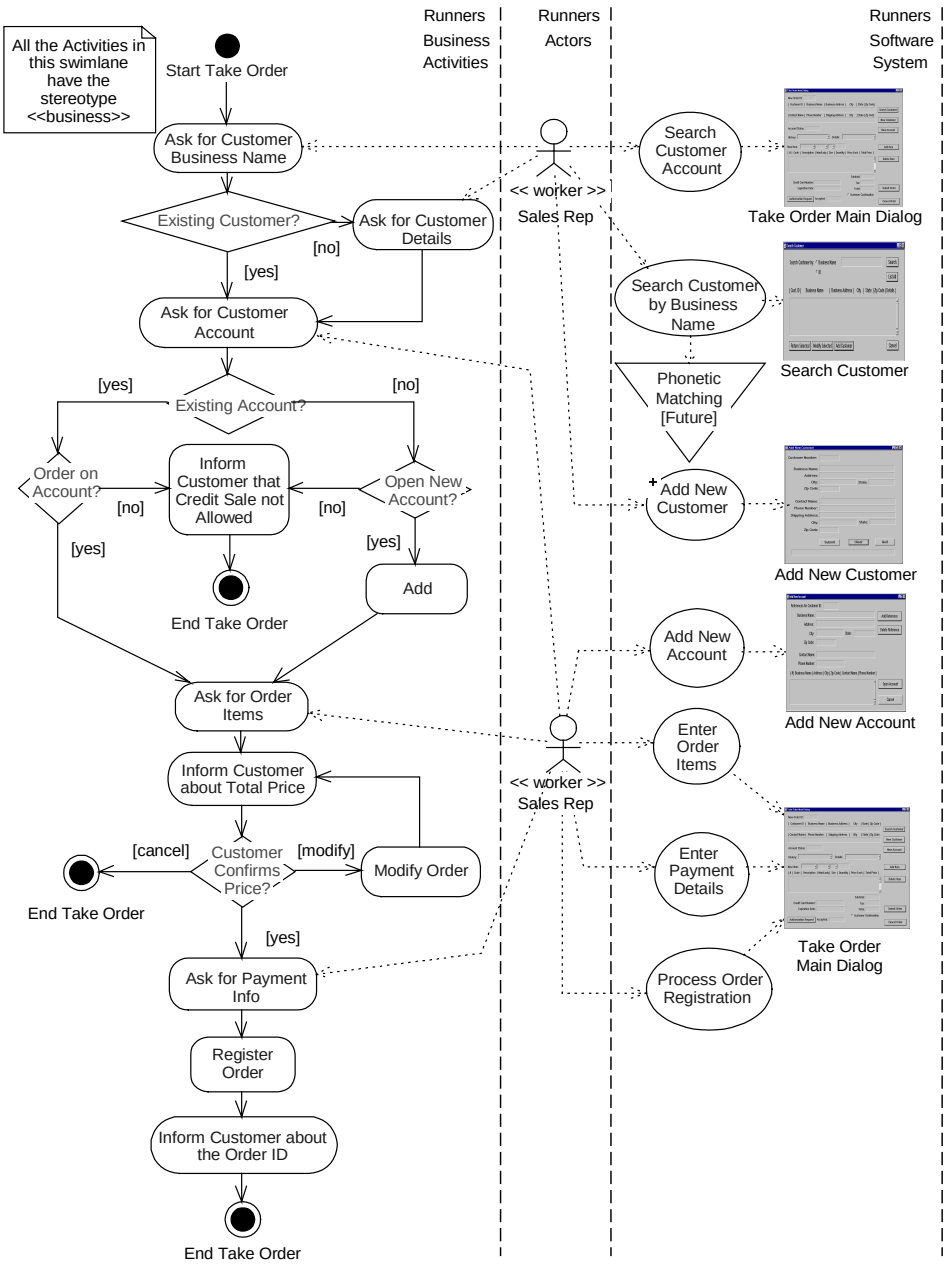

Figure 6. "Take Order" activity diagram

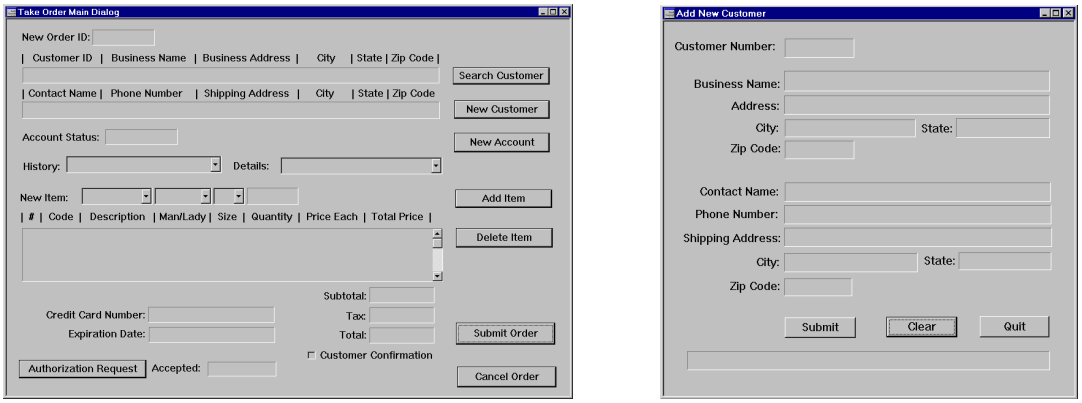

Figure 7. User Interface Views: a) "Take Order Main Dialog" and b) "Add New Customer"

3.5. "Add New Customer" Activity Diagram

The "Add New Customer" activity diagram (Fig. 8) documents the use case "Add New Customer" identified in Fig. 6. The purpose of this diagram is to assist in software design and for this reason it is formed by "software" activities. It is based on the requirements expressed in the "Add New Customer" dialog (Fig. 7.b), and it is supported by a class package (Fig. 9). It is relatively straightforward to identify and define the four classes that form the "Add New Customer System" package. The "AddNewCustomer_Controller" class uses the "AddNewCustomer_Dialog" class to collect the "Customer" details. An auxiliary "Address" class is used to record either business or shipping address details.

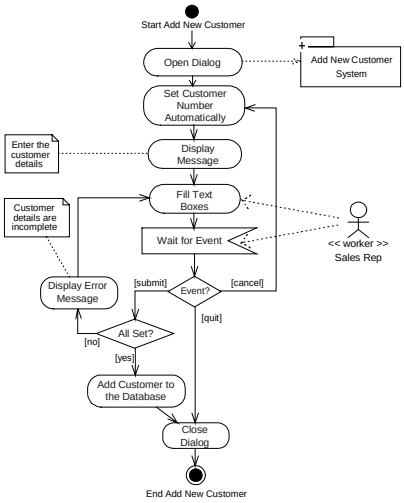

Figure 8. "Add New Customer" activity diagram

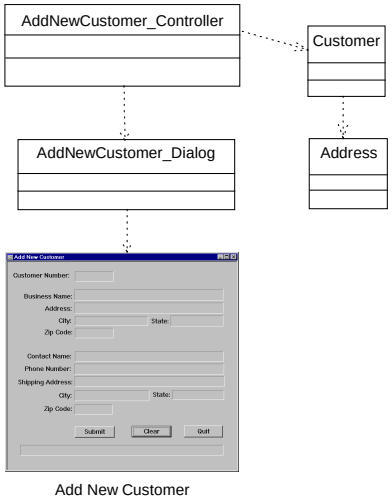

Figure 9. "Add New Customer" system

Finally, the use cases, the actors and the classes are exported from the RBPL model to a Rose model (Fig. 10).

Figure 10. RBPL model exported to Rational Rose

The "Analyze and Design" activity assigned in Fig. 1 to RBPL and detailed above was developed using a top-down approach. In the RBPL model, the diagram hierarchy is navigable in both directions. A meta-diagram is presented as Fig. 10 in order to show how the resulting activity diagrams are linked inside the model.

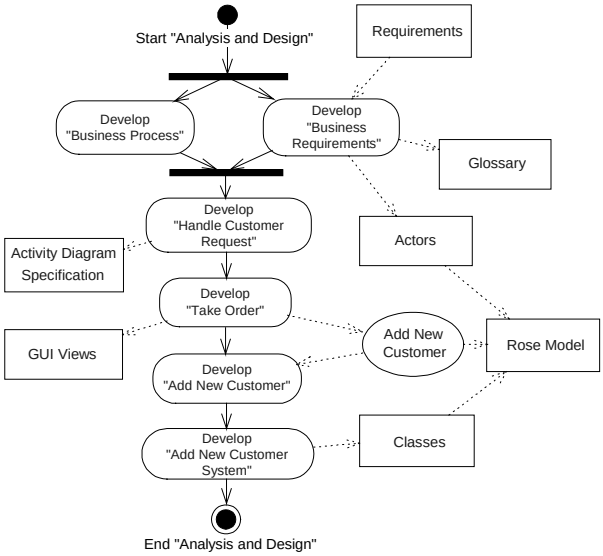

Figure 10. "Analyze and Design"

4. Round-Trip Engineering

The Rose model exported from RBPL is the starting point for a round-trip engineering process that involves Rational Rose, Ensemble's Rose JBuilder Link (Fig. 11), and Borland's JBuilder2. Through an iterative process, the Rose model and the JBuilder project are updated and synchronized.

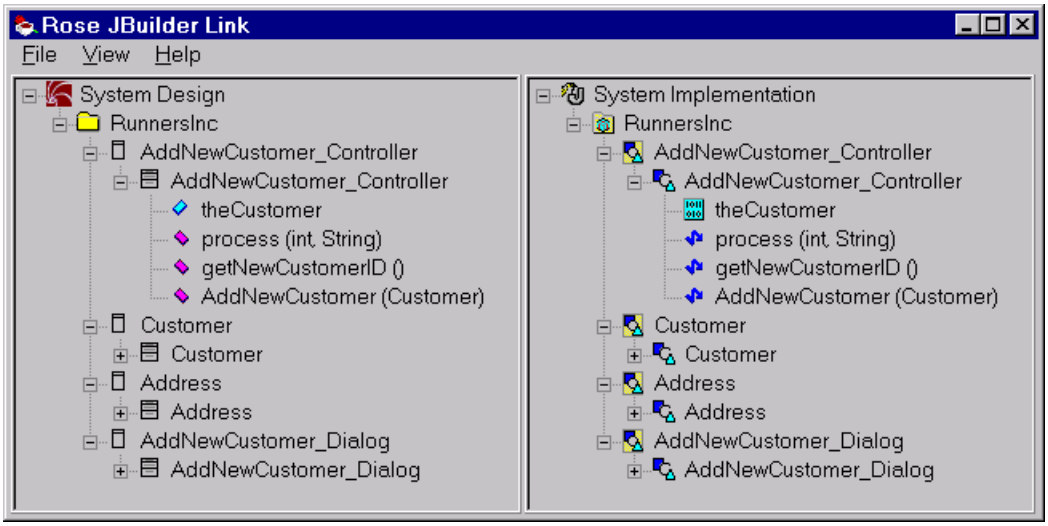

Figure 11. Rose JBuilder Link

The project is developed using a two-tier client-server model for database access. The "AddNewCustomer_Controller" class talks directly to a relational MS Access database via the JDBC-ODBC bridge [Hamilton 98]. The controller sends SQL statements and receives "ResultSet" objects from the Runners Inc. database. The actions of the controller are event-driven ("Submit", "Clear", and "Quit") from the "AddNewCustomer_Dialog" class which is a standalone runnable applet (i.e. it has both *init()* and *main()* methods and thus it can be run alone as a Java application without being called from an HTML page). The dialog provides a simple data entry form that contains no information concerning the business objects (here the customer details). The communication with the controller is done using Java Strings. The Rose model main class diagram is presented in Fig. 12 and the sequence diagram that details the "Submit" event (which is part of the "Add New Customer" use case) is shown in Fig. 13.

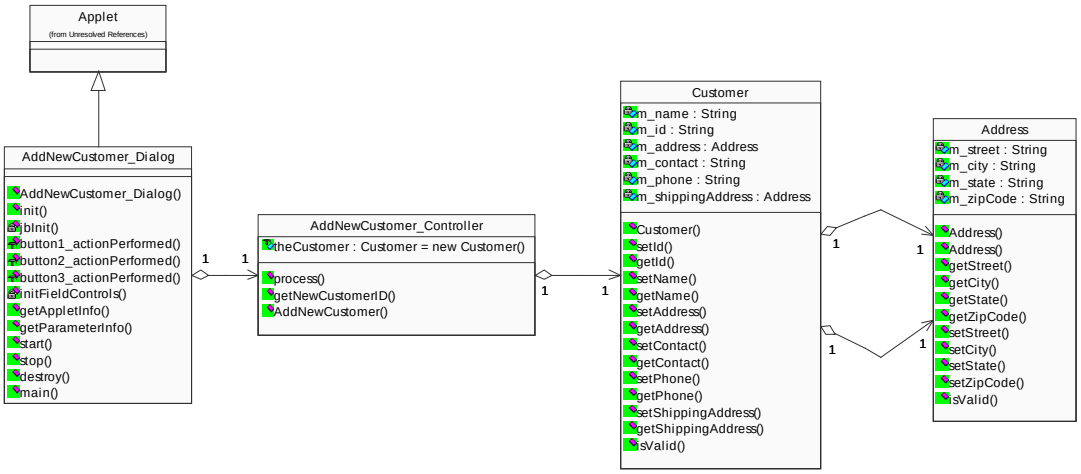

Figure 12. Runners Inc. Rose model

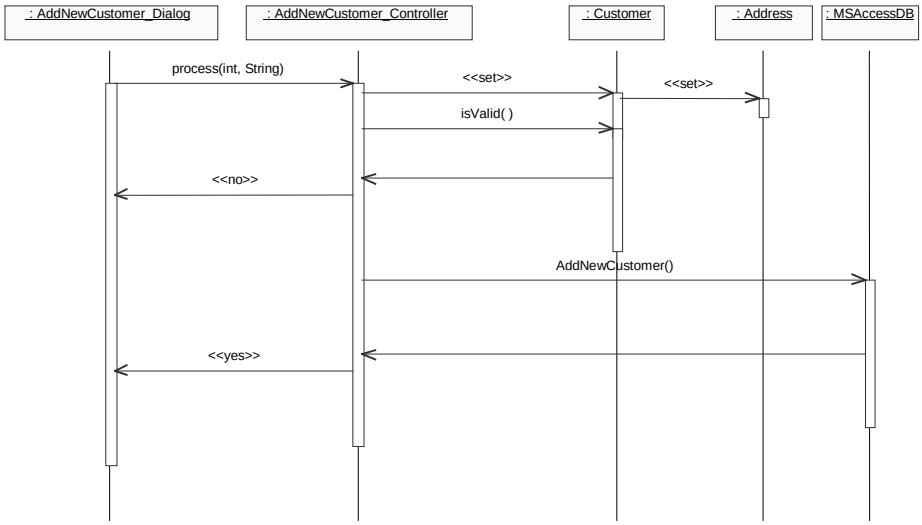

Figure 13. "Add New Customer" use case: "Submit"event sequence diagram

5. Runners Inc. Database

The Runners Inc database is created using Microsoft Access 97. It contains the same data fields formerly identified in the "Add New Customer" user interface view (Fig. 7.b), and then implemented in the "Customer" class (Fig. 12). In order to keep track of the customer record history, a supplementary data field, "CustomerStatus", was added.

The RunnersIncDatabase.mdb should be properly installed using the Microsoft Access Driver from the "ODBC Data Source Administrator" of the ODBC32 application. For more details consult the following reference **[**Milne 98].

CustomerId	BusinessName	BusinessAddress	BusinessCity	BusinessState	BusinessZipCode
8	Early Birds Joggers	8900 Nest Plaza	Ducksville	OR	7W7 1X1

ContactName	PhoneNumber	ShipAddress	ShipCity	ShipState	ShipZipCode	CustomerStatus
John Carefoot	1-800-345-6754	8900 Nest Plaza	Ducksville	OR	7W7 1X1	New Customer

Figure 14. "Runners Inc. Database" entry

6. Java Runtime Environment

In order to avoid applet security issues and simplify the setup process, the Java classes could be run either as an applet from the JBuilder IDE, or as a Java application using the Java Runtime Environment [JRE 1.1.7]. For this purpose, a *jar* file containing the Runners Inc. classes, Borland's JBCL (JavaBeans Component Library) runtime classes, and ObjectSpace's JGL (Generic Collection Library for Java), is created using the JBuilder Deployment Wizard [Jensen 98]. The executable jre.exe can be used from a batch file to start the Java VM and execute the Runners Inc. application. The applet dialog is presented in Fig. 12.

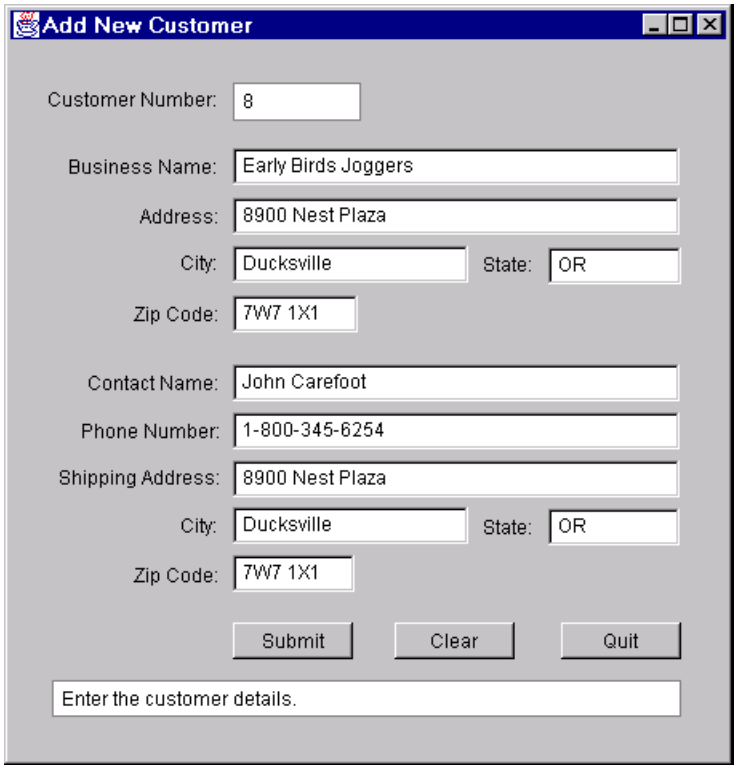

Figure 12. "Add New Customer" applet

7. Additional Information

The RBPL and Rose models together with the complete Java code are available on the Ensemble Systems Inc. web site at: www.ensemble-systems.com. Fully functional 30-day trials of Rose Business Process Link and Rose JBuilder Link are also available. For more information or to download a demo version of Rational Rose 98, visit: www.rational.com.

8. Bibliography

[Berkem 98] Berkem, B., "Formalizing a 'bridge' from the UML's Activity Diagram toward Use Cases", OOPSLA'98 Workshop on Formalizing UML, Vancouver, Canada, 1998, http://www.oblog.pt/workshop_program.htm

[Booch 98] Booch, G., Jacobson, I., Rumbaugh, J., *"The Unified Modeling Language User Guide"*, Addison-Wesley Pub Co., 1998, ISBN: 0201571684, http://www.amazon.com/exec/obidos/ASIN/0201571684/argoumlhomepage/002-5829840-5207043

[Cockburn 98] Cockburn, A., *"Basic Use Case Template"*, 1998, http://members.aol.com/acockburn/papers/uctempla.htm

[Hamilton 98] Hamilton, G., Cattell, R., Fisher, M., *"JDBC™ Database Access with Java™. A Tutorial and Annotated Reference"*, Addison-Wesley Pub Co., 1998, ISBN: 0201309955, http://www.amazon.com/exec/obidos/ASIN/0201309955/o/qid=922561735/sr=2-2/002-5829840-5207043

[Hruby 97] Hruby, P., *"The Object Model for a Product Based Development Process"*, ECOOP '97 Workshop on Modeling Software Processes and Artifacts, Jyväskylä, Finland, 1997, http://www.navision.com/services/methodology/methodologists_site.asp

[Hurlbut 98] Hurlbut, R.R, *"Managing Domain Architecture Evolution Through Adaptive Use Case and Business Rule Models"*, Ph.D. Thesis, Illinois Institute of Technology, 1998, http://www.iit.edu/~rhurlbut/research.html

[Jensen 98] Jensen, C., Stone, B., Anderson, L., *"JBuilder Essentials"*, Osborne McGraw-Hill; ISBN: 0078822238, 1998, http://www.amazon.com/exec/obidos/ASIN/0078822238/qid%3D922583109/002-5829840-5207043

[JRE 1.1.7] Java Runtime Environment, JRE 1.1.7B, http://java.sun.com/products/jdk/1.1/jre/download-jre-windows.html

[Milne 98] Milne, P., Andrews, M., "The Jtable Class Is DB-Aware", The Swing Connection, 1998, http://java.sun.com/products/jfc/tsc/archive/tech_topics_arch/db/db.html

[RBPL 99] Ensemble Suite, Technical documentation, http://www.ensemble-systems.com/downloads.html

[Rumbaugh 98] Rumbaugh, J., Jacobson, I., Booch, G., *"The Unified Modeling Language Reference Manual"*, Addison-Wesley Pub Co., 1998, ISBN: 020130998X, http://www.amazon.com/exec/obidos/ASIN/020130998X/ref=sim_books/002-5829840-5207043

[Schneider 98] Schneider, G., Winters, J. P., Jacobson, I., *"Applying Use Cases: A Practical Guide"*, Addison-Wesley Pub Co., 1998, ISBN: 0201309815, http://www.amazon.com/exec/obidos/ASIN/0201309815/ref=sim_books/002-5829840-5207043

[UML 1.3] OMG UML V1.3 alpha R2, January 1999, http://www.rational.com/uml/index.jtmpl

9. Appendix

RBPL Activity Diagram Notation