Business Process Modeling with UML.

Conference Paper · January 2001

Source: DBLP

4 authors:

Nuno Castela
Polytechnic Institute of Castelo Branco
21 PUBLICATIONS **68** CITATIONS

SEE PROFILE

José Tribolet
Inesc-ID
269 PUBLICATIONS **3,339** CITATIONS

SEE PROFILE

Alberto Rodrigues Da Silva
University of Lisbon
255 PUBLICATIONS **1,965** CITATIONS

SEE PROFILE

Arminda G Lopes
Polytechnic Institute of Castelo Branco
56 PUBLICATIONS **145** CITATIONS

SEE PROFILE

Some of the authors of this publication are also working on these related projects:

Project   Call For Paper: Special Issue: Requirements Engineering: Practice and Research View project

Project   RSLingo View project

BUSINESS PROCESS MODELING WITH UML

Author: Craig Dewalt

Instructor: Karthik Shyamsunder

for Johns Hopkins University

7 December 1999

CONTENTS

LIST OF ILLUSTRATIONS

ABSTRACT

The Unified Modeling Language (UML) is a powerful notation for building software blueprints. This report shows UML diagrams of business processes that are used for analyzing information instead of building software blueprints. The diagrams capture business process information for a fictional internet company.   Since the diagrams show both general and detailed information, they demonstrate that UML is capable of displaying various kinds of information. Flexible, easy to comprehend, and easy to build are traits that make UML diagrams a superior choice for business process modeling.

KEYWORDS

UML INTRODUCTION

The Unified Modeling Language (UML) is a standard language for writing software blueprints that premiered with version 1.1 in 1997. Three prominent object-oriented programming professionals, Gray Booch, Ivar Jacobsen, and James Rumbaugh are the principle authors of UML. UML establishes a collection of graphical symbols as well as semantics to support and define these symbols. This collection can be broken down into three kinds of building blocks: things, relationships, diagrams. Things are the abstractions that are first-class citizens in a model; relationships ties these things together; diagrams group interesting collections of things. There are nine different kinds of diagrams in UML: class, object, use case, sequence, collaboration, statechart, activity, component, and deployment (Booch, Jacobsen, Rumbaugh, The Unified Modeling Language Users Guide 13-18).

UML traces is origins to the software development field. Many of the definitions for UML artifacts and notation define rules and semantics based on the assumption that the purpose is building or designing software elements. An element is an atomic constituent of a model (Booch, Jacobsen, Rumbaugh, The Unified Modeling Language Users Guide 461). What if, however, the purpose was not to build models in support of a software system, but instead to map out business processes? Is UML still expressive enough so that it is useful in capturing information that is not going to be used to produce software?

One early indication that UML is well suited for this task is that many of the concepts in UML previously existed in object-oriented programming. One part of object-oriented programming is object-oriented analysis. Object-oriented analysis is a method

of analysis that examines requirements from the perspective of the classes and objects found in the vocabulary of the problem domain (Booch 39). An object is something you can do things to (Booch 516). A class is a set of objects that share a common structure and a common behavior (Booch 512). The significance of object-oriented analysis is to build objects that directly represent things in the physical world. Other techniques try to capture information and translate that information into some kind of container that is well suited for constructing software but may not exist outside of the software. Object-oriented analysis equates a real entity to a corresponding software entity as closely as possible.

Since UML is built upon many of the principles of object-oriented programming including object-oriented analysis, it is likely that UML is also effective at representing real-world entities. One limitation to building UML diagrams that are not going to be directly used to build a software system is that many definitions in UML include references to software systems. Some degree of interpretation is therefore necessary in order to use UML outside its intended purpose.

BUSINESS PROCESS MODELING

A model is a representation of a set of components of a process, system, or subject area (United States General Accounting Office 66). A business process is a set of activities that takes an object as an input and adds value to it in order to meet the requirements (What is a Business Model Page). A business process model is therefore a set of components that shows a set of activities. The purpose of creating a business process model is to better understand, analyze, improve, or replace a business process. Modeling is not limited to visual representations since it is possible to create models in a variety of different formats. A digital sound file that simulates noise is an example of a model that can only be understood by listening to, as opposed to looking at the model. This report will include an additional constraint on the definition of a model. This constraint is that the model must be some kind of graphical representation of activities within a process.

Often the business process model is used in conjunction with business process re-engineering. Business process reengineering is a systematic, disciplined improvement approach that critically examines, rethinks, and redesigns mission-delivery processes in order to achieve dramatic improvements in performance in areas important to customers and stakeholders (United States General Accounting Office 65).   Developing models is one of the first steps in business process re-engineering. Other steps include taking a fresh look at the objectives of the process (Business Process Improvement Workbook 9). Business process reengineering is beyond the scope of this report. This report is limited to creating business process models with UML; a logical next step would be to use the models to support reengineering activities.

The paradox of modeling is deciding how much detailed information is necessary to include in the model in order to gain an understanding of the topic being modeled. Haphazardly choosing information to place in the model is not a recommended technique because it is easy to ignore pertinent information. The consequence of this action is that the model will not accurately represent what was being modeled. Therefore some logical way of showing a subset of information is necessary in order to ensure that the model is accurate. Abstraction provides a mechanism to enable this kind of information filtering. A good abstraction emphasizes details that are significant to the reader or user and suppresses details that are diversionary (qtd. in Booch 41). Creating an effective abstraction is not easy to accomplish because deciding which details are important is a fairly subjective decision.

There are a large number of established techniques that support business process modeling. These techniques include Process Mapping, Simulation, Role Activity Diagrams, Integration Definition for Function Modeling (IDEF), Fishbone Diagrams, and Flowcharting, as well as many others. Nonetheless one of the most popular techniques is the "cocktail napkin" approach. This approach encompasses any informal technique that is made up spontaneously even if it borrows from other techniques.

Regardless of the technique, there is one goal that is common with all these techniques is to provide a pictorial representation of objects that perform some function and yield a result. This picture can then be used for a significant effort such as business process re-engineering or software engineering, or it can simply be used to explain a particular topic and then be discarded. Whether it is developed using a well documented technique and an advanced Computer Aided Software Engineering (CASE) tool or if it is

scribbled on the back of a cocktail napkin, as long as the technique assists someone in

comprehending a subject, it has served its purpose.

IDEF0 MODELS

One way to gain a better understanding of how UML can be used to capture business processes is to first consider notations that existed before UML, such as IDEF. IDEF models were most popular in government information technology projects in the 1980's (Barrett). IDEF is a set of standards developed by the United States Air Force Program for Integrated Computer Aided Manufacturing (ICAM) in an effort to increase manufacturing productivity. IDEF0 models are a way to capture a static view of a process or relationship (IDEF Home Page). The IDEF0 standard release in 1981 is designed to represents functions of a manufacturing system. The modeling portion of IDEF0 uses Input Control Output Mechanisms (ICOMs). The basic semantics for ICOMs are boxes and lines, sometimes called arrows, as in example 1.

EXAMPLE 1
IDEF0 BASIC SEMANTICS
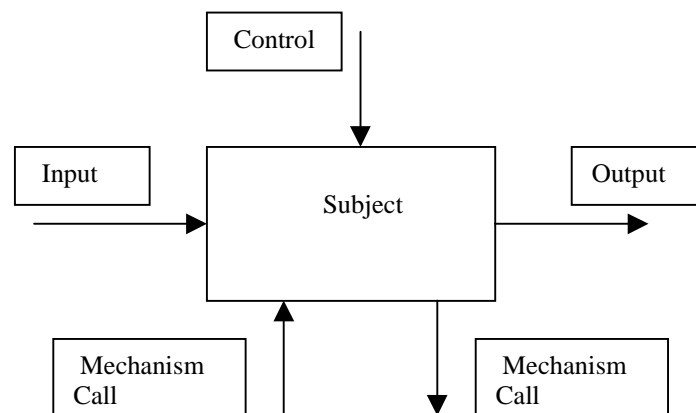
Boxes are named with a verb phrase and represent activities, actions, tasks, or other action-oriented parts of systems. Lines are named with a noun phrase and can be inputs, controls, outputs, mechanisms, or calls. IDEF models follow a rigid and

structured nature to capture information because the models are built by successfully increasing details in "child" levels that are mapped to higher levels (IDEF Home Page).

A comparison of IDEF0 models to UML is not necessarily a valid comparison. IDEF0 attempts to represent the physical world within its notation while UML defines a notation that is designed to represent the physical world. Also UML diagrams do not have notational constructs that are as restrictive as some of the constructs that apply to IDEF0. For example there are only four kinds of arrows, input, output, control, and mechanism arrows in IDEF0. In contrast, the arrow counterpart in UML, a message, has no corresponding limitations. Because the two diagramming procedures are very different, it is not reasonable to try and justify that one method is better than the other. Stating that UML is better than IDEF0 is like stating a baseball bat is better than a hockey stick. External influences such as experience, position, preference, or time will all affect how a particular technique or tool will be used.

The shortcomings of IDEF0 are indicators that reflect the strengths of UML. While IDEF0 models tend to have some restrictive qualities such as the types of arrows permitted, UML diagrams tend to have fewer limitations. Furthermore, the hierarchy that is essential in IDEF0 models is not required in UML diagrams.

INFOMARKET.COM

Infomarket.com is a fictional company based upon genuine business plans of actual companies. Infomarket.com is an internet company provides an electronic storefront for buying and selling unique research reports. There are two primary customers: buyers and sellers. Buyers are consumers who purchase the reports and sellers are the report authors.

The Infomarket.com Project Manager is a fictional character who will serve as the subject matter expert who will review all Infomarket.com diagrams. The Infomarket.com Project Manager has received an appropriate level of training in order to understand the diagrams, including parts of the diagrams that are not explained. The complete list of UML notation including the diagrams in this report is available at www.rational.com.

The purpose including Infomarket.com is to provide realistic diagrams that demonstrate the topics in this report. The six Infomarket.com diagrams (from broad to narrow in scope) are Example 4 Infomarket.com Packages, Example 3 Seller Services Use Case Diagram, Example 2 Register Account Use Case, Example 5 Register Account Sequence Diagram, Example 6 Register Account Activity Diagram, Example 7 Modified Register Account Activity Diagram. Additionally, there are a variety of steps necessary in order to build the UML diagrams that are not identified.

MODELING PROCESSES IN USE CASE DIAGRAMS

Use case diagrams are an excellent way to display business processes in a visual model because the powerful abstraction granted by the simple notation makes them both easy to build an easy to understand.   A use case is a description of a set of sequences of actions, including variants, that a system performs that yields an observable result of value to an actor (Booch, Jacobsen, Rumbaugh, The Unified Modeling Language Users Guide 468). Example 2 shows a use case for one of the business processes of Infomarket.com**.** A use case diagram shows a set of use cases and actors and their relationships (Booch, Jacobsen, Rumbaugh, The Unified Modeling Language Users Guide 468). Example 3 is a basic use case diagram that shows the basic seller services of Infomarket.com. The Register Account use case in Example 2 is one of the use cases in the use case diagram in Example 3. The simple oval notation that represents a use case on a use case diagram uses abstraction to effectively eliminate a great deal of detailed information. In both examples there is an actor called 'Seller'. An actor is a coherent set of roles that users of use cases play when interacting with the use cases (Booch, Jacobsen, Rumbaugh, The Unified Modeling Language Users Guide 457). An actor's role is defined by what it does in the context of a particular use case. The implication of this definition is that the same person or thing could be listed as a different actor in different use cases, since the actor's role is defined by their actions in each use case. Use case diagrams can be used to convey how a collection of use cases define or demonstrate a processes.

Use cases evolved from Ivar Jacobsens's work in the telecommunications industry in the late 1960's when he used traffic cases and functions for the same purposes that use

cases are used for today (Ivar Jacobsen Interview). Use cases are popular for a number of reasons. Two reasons are that easy to understand and that they are easy to construct. One report identified use cases and use case diagrams as being easily understood by everyone on the software project team, including users and non-technical managers (Literate Modeling Papers Page). Another reason that use cases are very valuable in software engineering projects is that they are truly multipurpose. They can be used to capture and document requirements, they can serve as valid test cases, and support marketing efforts by outlining the most prominent features of a system.

The use case diagrams are most effective when they are combined with use cases (text) that define the use cases in the diagram. Nonetheless the use case diagram by itself is still a beneficial model.

The high-level business processes in Example 3 are Update Personal Data, Request Payment, Use Support Tools, Purchase Advertisement, Register Account. The use cases represent the services that Infomarket.com provides to individuals who will be selling materials through Infomarket.com. The use case diagram was drafted before the details that will make up the use cases were available. Therefore the diagram served the purpose of outlining to the Informarket.com Project Manager the major components of services that Infomarket.com provides to the seller. The diagram can also be used in a presentation to senior management or other interested parties. In such a case the use case diagram would be used simply to provide a snapshot of major components. Many of the other techniques mentioned earlier could be used for the same purpose. Use case diagrams were chosen because they are a very elegant solution. They are elegant because there is not a lot of notation in the diagrams, which makes the use case diagram easy to

comprehend. One of the key outputs of any successful modeling technique is that it

identifies the major functions of a business (Damelio 33). In this example the use case

diagram is used simply to provide an answer to the question "What are the major services

(business processes) that Infomarket.com provides to sellers?"

HIGH LEVEL PROCESSES WITH UML PACKAGES

Placing several use cases in a use case diagram is a good strategy in order to demonstrate the logical connection between the use cases. Too many use cases on the same diagram will cause the diagram to become confusing. While there are no defined limits on the number of use cases that belong on one diagram, any more than 10 use cases on the same diagram could make the diagram confusing.

UML provides a way to prevent confusion by organizing UML elements, including use cases, with packages. A package is a general-purpose mechanism for organizing elements into groups (Booch, Jacobsen, Rumbaugh, <u>The Unified Modeling Language Users Guide</u> 464). Example 4 shows a set of packages that represent the high level business processes of Infomarket.com. Each package contains a group use cases that are logically related. For example the Seller Services package includes Update Personal Data, Request Payment, Use Support Tools, Purchase Advertisement, Register Account. All of these use cases support or define some kind of seller services.

Packages can be used to organize any UML element. Packages can then be placed on a variety of UML diagrams, including use case diagrams. In fact, packages can be thought of as a fancy name for a file folder. The package diagram in example 4 does not offer a tremendous amount of advantages from a modeling perspective. It is simply an organizational tool. Packages are important because they can be used to organize a variety of UML elements and they can be used in all levels of a model.

SEQUENCE DIAGRAMS VERSUS ACTIVITY DIAGRAMS

There are several ways to capture detailed business process information that provide more information in order to supplement use cases. Two of the possibilities within UML are sequence diagrams and activity diagrams. Sequence diagrams are a kind of an interaction diagram that emphasize the time ordering of messages. A message is a specification of a communication between objects that conveys information with the expectation that activity will ensue (Booch, Jacobsen, Rumbaugh, <u>The Unified Modeling Language Users Guide</u> 463). An interaction diagram is a diagram that shows an interaction, consisting of a set of objects and their relationships, including messages that may be dispatched among them. Activity diagrams show the flow from activity to activity while interaction diagrams emphasize the flow from object to object (Booch, Jacobsen, Rumbaugh, <u>The Unified Modeling Language Users Guide</u> 257). Another difference between an activity diagram and a sequence diagram is the time ordering of messages that is present in a sequence diagram but not in an activity diagram. Therefore when you view a sequence diagram, time increases as you proceed from the top to the bottom of the diagram. Activity diagrams do not necessarily include this constraint. The two diagrams also use slightly different notation.

The Infomarket.com Project Manager was provided the opportunity to choose which diagrams (sequence or activity) he would prefer to review for detailed business process information. The diagrams are provided in addition to the use case text that elaborates the use cases in the use case diagram. He was given the sample diagrams in example 5 and 6 for the Register Account Use Case and asked which diagram he would prefer for the remaining use cases. The Infomarket.com Project Manager chose the

activity diagrams instead of sequence diagrams because the activity diagram was easier

for him to understand. Clearly either kind of diagram would be feasible from a technical

perspective. The decision to build activity diagrams was based solely on the reviewer's

preference.

ADVANCED MODELING CONCEPTS

UML provides a wide variety of elements that can be used effectively in modeling. This report only reviewed a small subset of these elements. Notes are an example of a UML element that can be very useful in all nine UML diagrams, as in the sequence diagram in example 5. A note is simply a graphical symbol that renders constraints or comments attached to an element or a collection of diagrams (Booch, Jacobsen, Rumbaugh, The Unified Modeling Language Reference Manual 464). In practice notes are a great mechanism to capture information that does not seem to belong anywhere in particular. Swimlanes are another important construct for modeling business processes. Swimlanes are partitions for organizing responsibilities for activities (Booch, Jacobsen, Rumbaugh, The Unified Modeling Language Reference Manual 461). As an object travels from one partition to the next, responsibility changes to the owner of the new partition. Swimlanes are often used in conjunction with activity diagrams such as example 6.

In UML there is no built in support for critical performance measures in business process reengineering. The four performance measures are cost, quality, service, and speed (United States General Accounting Office 6). UML does provide extensibility mechanisms that can be used to capture information such as performance measures. An extensibility mechanism is one of three mechanisms (stereotypes, tagged values, and constraints) that permit you to extend the UML in controlled ways (Booch, Jacobsen, Rumbaugh, The Unified Modeling Language Users Guide 461). An example of one of these extensibility mechanisms, tagged values, is shown on example 7. The tagged value represented by the text {time = #seconds} provides additional information for each

activity. The tagged value could be any additional data for the model. The tagged values

in example 7 show the time required to complete each activity. The total time required to

complete the Register Account process is listed at the end of the model. The tagged

value was therefore a useful way to customize the standard UML diagram while still

following UML notation.

The disadvantage of adding non-standard information, such as a tagged value, to a

diagram is that your are customizing the UML diagram. The modeler now has the extra

responsibility of providing sufficient explanation for their additions to the diagram.

Using the extensibility mechanisms, as opposed to inventing new symbols, to add more

information to the diagram is therefore a good way to prevent this disadvantage from

causing unnecessary confusion. Most CASE tools only have limited support for such

extensions. An activity diagram such as Example 7 that shows the tagged value of time

would need to have the total time calculation performed manually unless the CASE tool

used to build the diagram could be adapted for this purpose.

CONCLUSION

There are many reasons why UML is an excellent choice for business process modeling. One of the reasons is that companies can train employees in the UML and then task the employees to work on software projects, quality improvement projects, technical documentation projects, or any activity that needs a reliable technique to capture information. From a technical perspective, UML is a superior choice for modeling all kinds of information. It can be used to capture complex information in its very expressive notation, or it can be used to display a limited set of simple concepts in a plain format that is easy for many inexperienced individuals to understand. From a business perspective, UML provides a cost-effective way to rapidly model processes. From a user perspective, UML diagrams are easy to comprehend after an appropriate level of training. Companies that are looking for an inexpensive way to quickly capture and present information could reap great benefits by incorporating the UML to fulfill this need.

EXAMPLE 2
REGISTER ACCOUNT
USE CASE

Name: Register Account

Purpose: Define the steps that a seller must follow in order for the seller to register with Infomarket.com so that they are permitted to sell reports on the Infomarket.com electronic store.

Actors: Seller

Preconditions: The seller understands the business proposition that Infomarket.com offers. The seller understands their responsibilities, and desires to register account information at Infomarket.com.
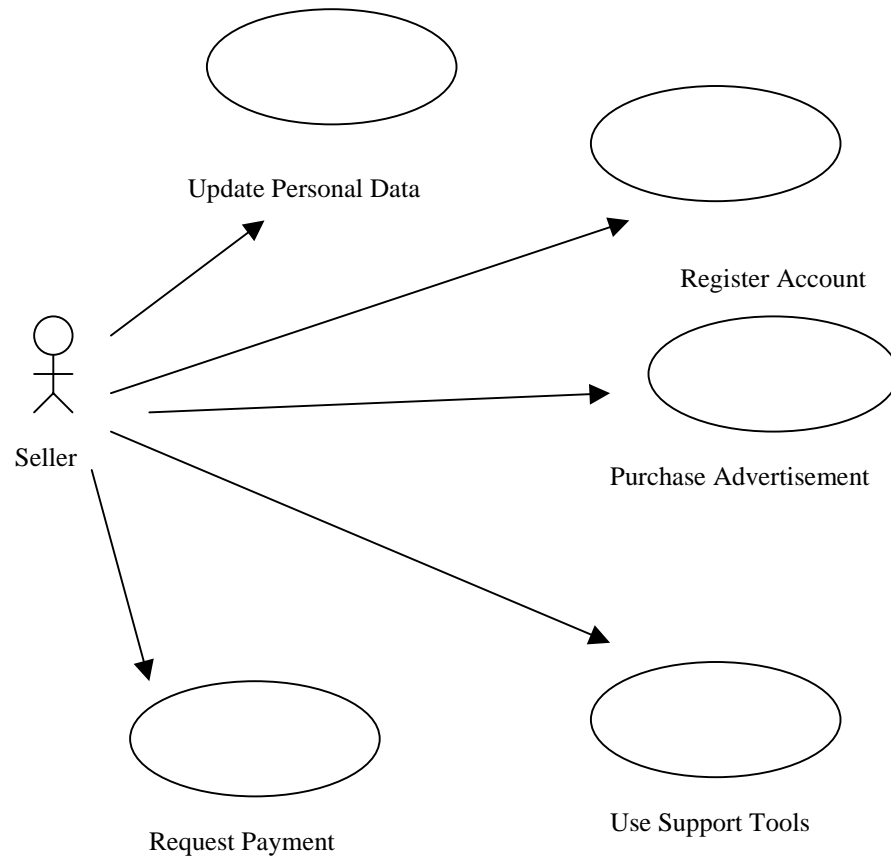
Main Flow:
1. Register Account begins when the seller finds location on website that allows them to register account information.
2. The seller reads the steps necessary to complete registration process.
3. The seller inputs all required personal data.
4. Register Account ends when the seller replies to an email confirmation of receipt of their account information.
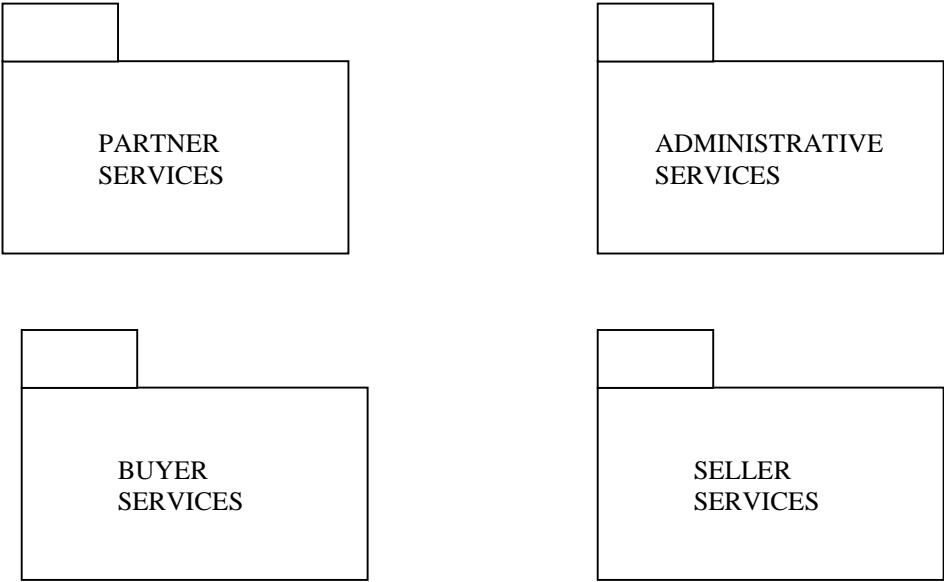
Alternate Flows: None

Postconditions: The seller has completed all the necessary steps in order to be permitted to sell material on Infomarket.com. The seller has received confirmation that the information they submitted has been received by Infomarket.com.

EXAMPLE 3

SELLER SERVICES
USE CASE DIAGRAM

Update Personal Data

Register Account

Purchase Advertisement

Seller

Request Payment

Use Support Tools

EXAMPLE 4
INFOMARKET.COM PACKAGES

PARTNER
SERVICES

ADMINISTRATIVE
SERVICES

BUYER
SERVICES

SELLER
SERVICES
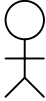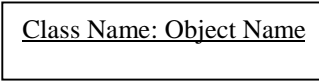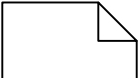
EXAMPLE 5
SEQUENCE DIAGRAM NOTATION

Purpose   The purpose of a sequence diagram is to emphasize time ordering of messages.

SEQUENCE DIAGRAM SYNTAX (SYMBOLS)

Symbol Name	Symbol
*Actor	
Object	Class Name: Object Name
Time	
Lifeline (Object)	
Message	⟶
*Note	

*Actors and notes are found throughout all UML diagrams.

EXAMPLE 5

ACCOUNT REGISTER
SEQUENCE DIAGRAM

Seller

Website Service:
Account Register

Not shown: all return
messages from Register
Object to Seller (normally
in the form of a web
page).

1. Choose FOR AUTHOR

2. Choose OUR MISSION

3. Select BACK BUTTON

4. Choose SELL YOUR
WORK

5. Choose I WANT TO
REGISTER

6. Read steps list

7. Read step1

8. Select CHECKBOX 1&2

9. Select JOIN
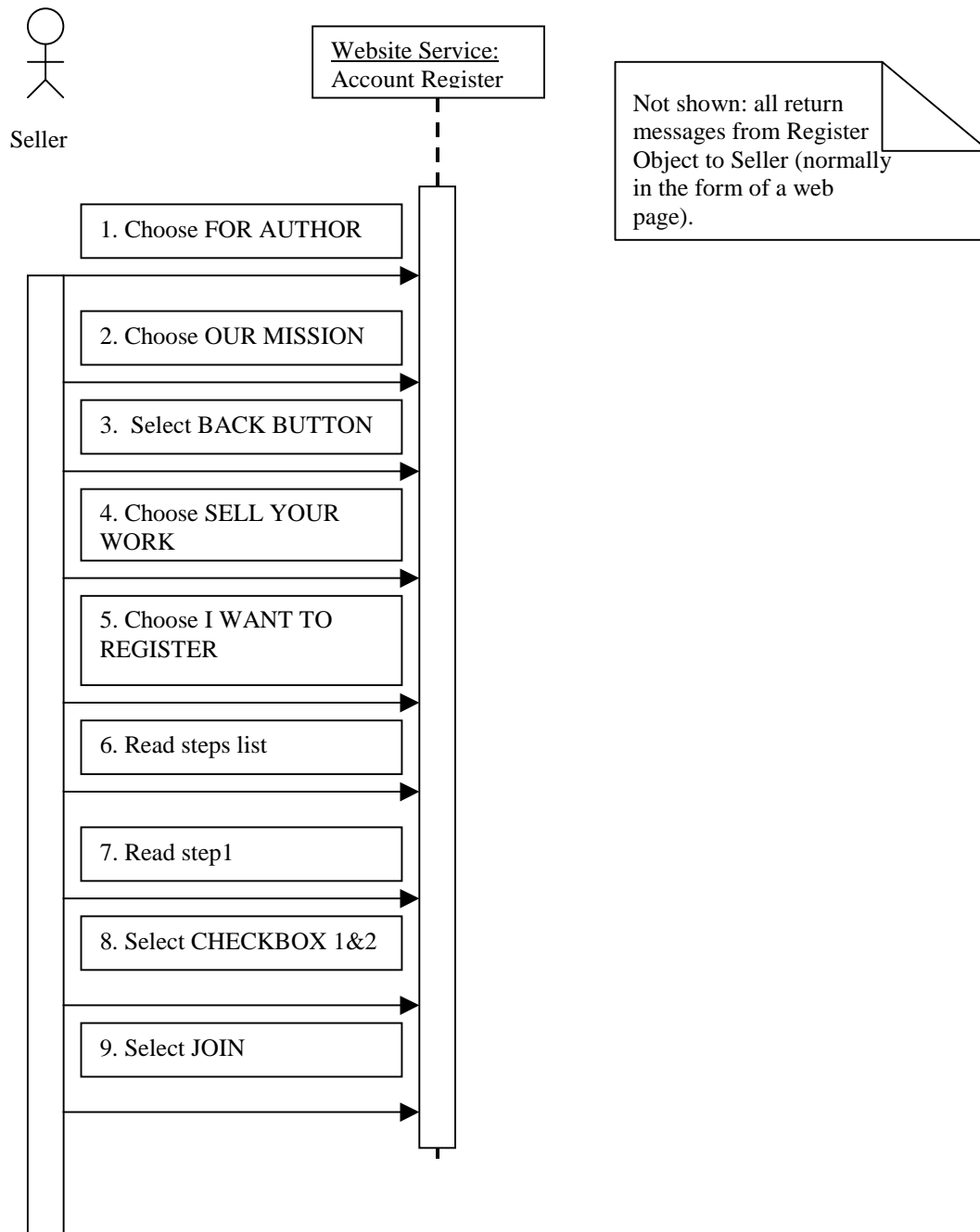
Continued next page

EXAMPLE 5

ACCOUNT REGISTER
SEQUENCE DIAGRAM (continued)
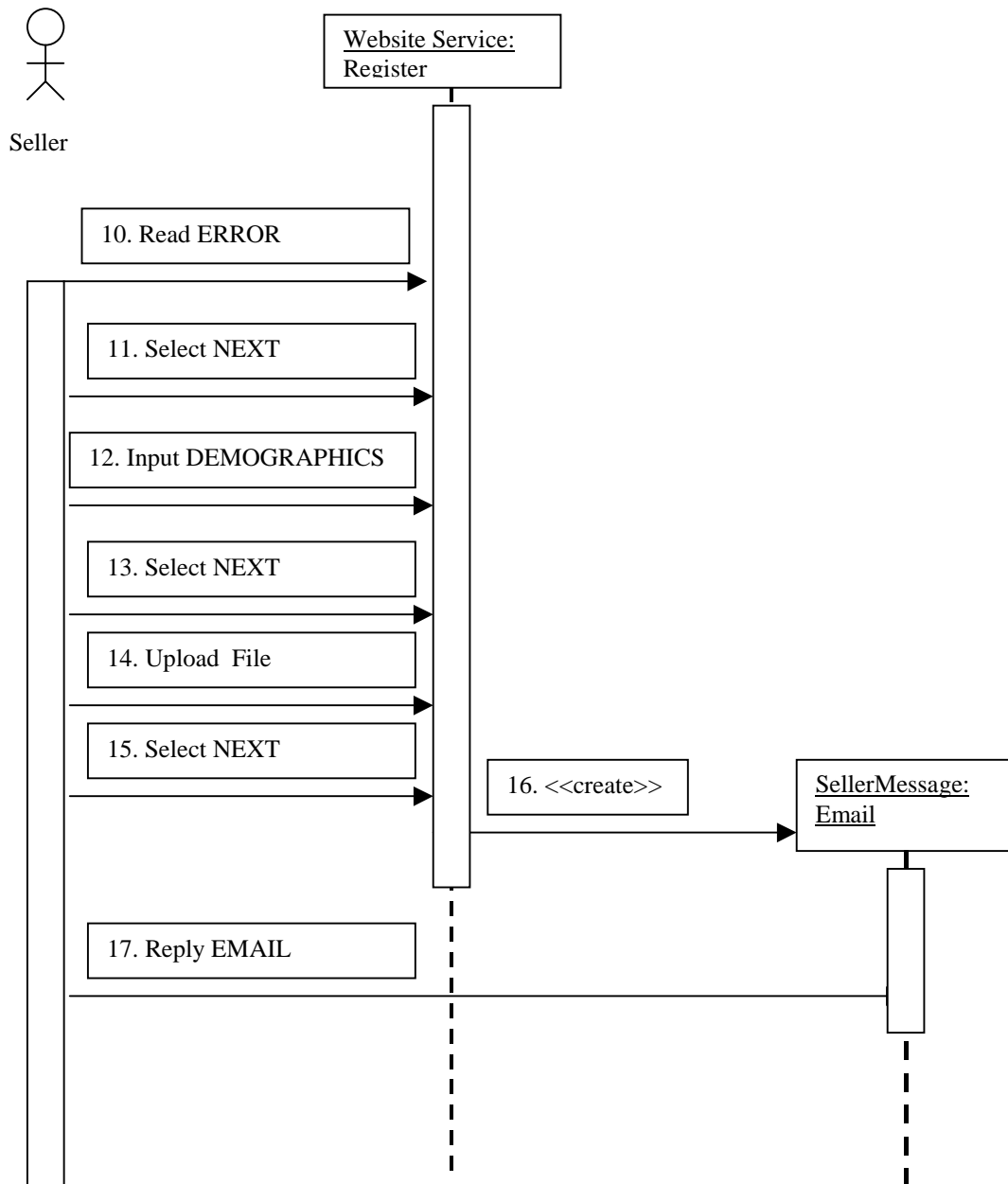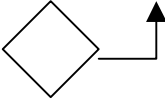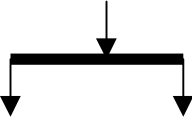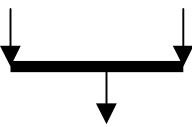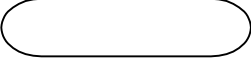
EXAMPLE 6

ACTIVITY DIAGRAM NOTATION

Purpose The purpose of an activity diagram is to emphasize flow of control from activity to activity. The end result of an activity diagram must be some action.

ACTIVITY DIAGRAM SYNTAX (SYMBOLS)

Symbol Name	Symbol
Initial State	●
Final State	◉
Sequential Branch	◇
Concurrent Fork	
Concurrent Join	
Action State	⬭
*Guard Expression	[*some expression*]
**Tagged Value	{ tag name = value }

*Guard Expression are most often used in conjunction with Sequential Branches

**Tagged Values are present on Example 7 only

EXAMPLE 6
REGISTER ACCOUNT
ACTIVITY DIAGRAM*

*Bold indicates text/icon on web page

```
                              ●
                              │
                              ▼
                    ┌──────────────────────┐
                    │ CHOOSE FOR AUTHOR    │
                    └──────────────────────┘
                              │
                    ┏━━━━━━━━━┻━━━━━━━━━┓
                    ▼                   ▼
        ┌──────────────────────┐  ┌──────────────────────┐
        │ CHOOSE OUR MISSION   │  │ CHOOSE SELL YOUR WORK│
        └──────────────────────┘  └──────────────────────┘
                    │                   │
                    ▼                   ▼
    [Hit Back Button]  ◇      ┌──────────────────────────┐
                              │ CHOOSE I WANT TO REGISTER│
         [Exit]               └──────────────────────────┘
                    │                   │
                    ▼                   ▼
                   ◉           ┌──────────────────┐
                              │ READ STEPS LIST  │
                              └──────────────────┘
                                       │
                                       ▼
                              ┌──────────────┐
                              │ READ STEP1   │
                              └──────────────┘
                                       │
                                       ▼
                              ┌────────────────────┐
                              │ SELECT CHECKBOX 1&2│
                              └────────────────────┘
```

SELECT JOIN

READ ERROR

◉

SELECT NEXT

INPUT DEMOGRAPHICS

SELECT NEXT

UPLOAD FILE

SELECT NEXT

REPLY EMAIL

28

◉ Account is registered.

EXAMPLE 7
MODIFIED REGISTER ACCOUNT
ACTIVITY DIAGRAM*

*s = seconds

CHOOSE **FOR AUTHOR**
{time=2s}

CHOOSE **OUR MISSION**
{time=2s}

CHOOSE **SELL YOUR WORK**
{time=2s}

[Hit **Back Button**]

[Exit]

CHOOSE **I WANT TO REGISTER**
{time=2s}

READ **STEPS LIST**
{time=18s}

READ **STEP1**
{time=58s}

SELECT **CHECKBOX 1&2**
{time=6s}

SELECT **JOIN**
{time=2s}

SELECT **NEXT**
{time=2s}

READ ERROR
{time=2s}

INPUT DEMOGRAPHICS
{time=25s}

EXAMPLE 7
MODIFIED REGISTER ACCOUNT
ACTIVITY DIAGRAM (continued)

SELECT **NEXT**
{time=2s}

UPLOAD FILES
{time=50s}

SELECT **NEXT**
{time=2s}

REPLY EMAIL
{time=45s}

Account is registered.
Total time = 207s
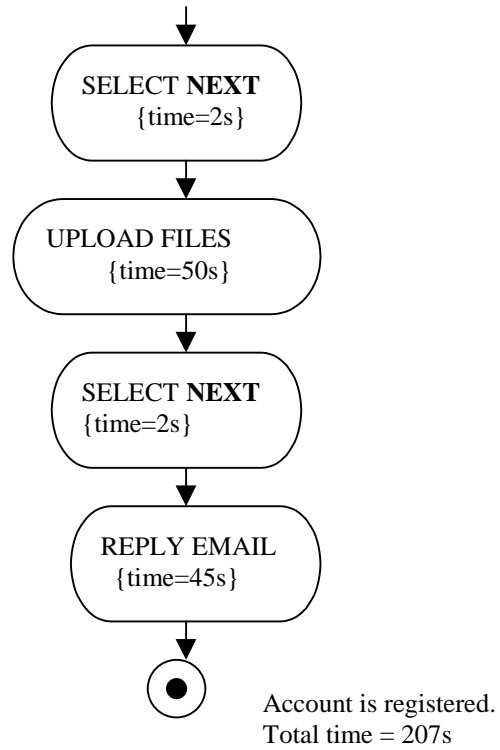
DEFINITIONS

Abstraction	An abstraction emphasizes details that are significant to the reader or user and suppresses details that are diversionary
Actor	A coherent set of roles that users of use cases play when interacting with the use cases.
Activity Diagram	The purpose of an activity diagram is to emphasize flow of control from activity to activity. The end result of an activity diagram must be some action.
Class	A class is a set of objects that share a common structure and behavior.
Element	An element is an atomic constituent of a model.
Extensibility Mechanism	An extensibility mechanism is one of three mechanisms (constraints, tagged values, and stereotypes) that permit you to extend UML in controlled ways.
Interaction Diagram	An interaction diagram is a diagram that shows an interaction, consisting of a set of objects and their relationships, including messages that may be dispatched among them.
IDEF	Integrated Definition Method.
Message	A message is a specification of a communication between objects that conveys information with the expectation that activity will ensue.
Model	A model is a representation of a set of components of a process, system, or subject area.
Object	An object is a concrete manifestation of an abstraction that represents a real-world entity.
Object-Oriented Analysis	Object-oriented analysis is a method of analysis that examines requirements from the perspective of the classes and objects found in the vocabulary of the problem domain.
Package	A package is a general-purpose mechanism for organizing elements into groups.
Sequence Diagram	A sequence diagram is an interaction diagram that shows time ordering of messages.
Swimlane	A swimlane is a partition on activity graphs for organizing responsibilities for activities.
UML	A standard language for writing software blueprints.
Use case	A description of a set of sequences of actions, including variants, that a system performs that yields an observable result of value to an actor.
Use Case Diagram	A use case diagram shows a set of use cases and actors and their relationships.

WORKS CITED

Barrett, Randy. Chasing the BPR Tool Market Page.   Enterprise Reengineering, March,
      1996. <www.dtic.mil/c3i/bprcd/5316.htm>

Business Process Improvement Workbook by H. James Harrington, Erik K. C. Esseling,
      Harm Van Nimwegen. McGraw-Hill. New York, New York 1997.

Damelio, Robert.   The Basics of Processing Mapping. Quality Resources.
      New York, New York 1996.

IDEF Home Page. Knowledge Based Systems. 1999 <http://www.idef.com>

IDEF0/SADT Business Process and Enterprise Modeling by David A. Marca and
      Clement L. McGowan. Eclectic Solutions Corporation San Diego CA. 1988.

Ivar Jacobsen Interview. Analisis-Disegno. 1999
      <http://www.analisis-disegno.com/uml/JacobsenInterview.html>

Literate Modeling Papers Page. Literate Modeling. 1999
      <http://www.literatemodeling.com/papers.htm>

Object-Oriented Analysis and Design with Applications by Grady Booch. Second
      Edition. Addison-Wesley. Santa Clara, CA. 1998.

Rational Home Page. Rational Software Corporation. 1999 <http://www.rational.com>

United States General Accounting Office. Business Process Reengineering Assessment
      Guide. Version 3, May 1997

The Unified Modeling Language Reference Manual by James Rumbaugh, Ivar Jacobsen,
      Grady Booch. Addison-Wesley. Reading, Massachusettes 1999.

The Unified Modeling Language User Guide by James Rumbaugh, Ivar Jacobsen, Grady
      Booch. Addison-Wesley. Reading, Massachusettes 1999.

What is a Business Model Page. Industrial and Financial Systems 1999
      <http://www.ifsab.com/modeling/gen96.htm>