

Analysis Modeling Guidelines

Document Name: Analysis Modeling Guidelines
Version: 1.1
Date: September 24, 2001

Table of Contents

1	Introduction	2
1.1	Scope	2
1.2	Document Organization.....	2
1.3	Definitions, Acronyms, and Abbreviations	2
1.4	References	5
2	Domain Modeling Guidelines	6
2.1	Introduction	6
2.2	Development Guidelines	6
2.3	Class Level Guidelines	6
2.4	Attribute Guidelines	6
2.5	Association Guidelines	6
3	State Diagrams.....	8
3.1	Introduction	8
3.2	Development Guidelines	8
	Appendix A – Sample Domain Model	Error! Bookmark not defined.
	Appendix B – Sample State Diagram.....	0

Analysis Modeling Guidelines

1 Introduction

An Analysis Model is a set of diagrams that help stakeholders understand the problem domain defined by the Requirements model. The purpose of this document is to identify the various analysis diagrams and to provide guidelines for developing and reviewing them. The audience includes domain (subject matter) experts, architects, systems analysts, developers, and testers.

The Analysis Model diagrams are represented graphically using an industry standard notation known as Unified Modeling Language (UML). Expertise in UML is not required for the purposes of these guidelines. However, some UML will be presented to help illustrate the analysis modeling guidelines. When UML is presented, it will be done with full explanation.

The Analysis Model is usually started in the Elaboration Phase of the project, and it is refined iteratively in conjunction with the development of the use cases or related artifacts. Put another way, the Analysis Model contains elements that can only be traced to the use cases from the current or prior iterations.

The Analysis Model should be reviewed in terms of its usefulness to the Domain Experts and Technical Staff.

1.1 Scope

The development of an Analysis Model is a recommended best practice of the Rational Unified Process (RUP). Unfortunately, RUP provides little guidance as to what kind of information needs to be captured. Compounding this problem is the fact that outside RUP, there is little agreement as well. As a result, there is a multitude of analysis modeling styles.

1.2 Document Organization

This document is organized as follows:

Section	Description
1	This section
2	Domain Models
3	State Diagrams

1.3 Definitions, Acronyms, and Abbreviations

Term	Definition
Abstraction	<p>An abstraction is a term that is used to describe a concept by using just enough detail so that it can be understood. Put another way, it is “elimination of the irrelevant and amplification of the essential”</p> <p>Example: A television set is an abstraction (at least to homeowners). As homeowners, we are interested in such details such as screen size, HDTV or not, stereo or mono, etc. By contrast, a TV repairperson might have a completely different abstraction, such as tuner strength, number of capacitors, etc.</p>
Analysis Model	A set of artifacts that captures, through various view of the system the conceptual classes, behaviors and states from the problem domain.

Term	Definition
	<p>Note: many of the artifacts in the Analysis Model are also used in the Design model, however, their respective audiences are quite different. The former provides value to both Business and Engineering staff while the latter provides value solely to the Engineering staff.</p>
Aggregation Relationship	<p>A special type of whole-part relationship that says that if the parent class goes away, it's children classes do not.</p> <p>Example: In a library, if the library is closed, the books “live on”, and can be moved to another library.</p> <p>By contrast, see the definition for a Composition Relationship.</p> <p>See the sample domain model (presented later) for additional information.</p>
Association Class	<p>An association relationship has a name, multiplicities and optionally roles. If additional information is needed, i.e. attributes and operations, it is placed in an association class.</p>
Association Relationship	<p>An association communicates that classes are in some way related to one another.</p> <p>Note: An association has many variants, such as whole-part and inheritance. See below for additional information.</p> <p>Example: A store is associated with one more departments.</p>
Association Multiplicity	<p>Indicates the number of objects that may participate in an association relationship.</p>
Association Name	<p>To give an association meaning, it is labeled with a term that makes sense to domain experts. See the sample domain model for additional information.</p>
Bi-directional Association	<p>By default, an association is bi-directional. Unfortunately, the UML is not specific as to what this means. A bi-directional association can communicate one of two things:</p> <ol style="list-style-type: none"> 1) Each class in the association can navigate through the other class in the association OR 2) The association is unspecified. <p>Example: A loan is associated with one to many borrowers. This means that the loan is responsible for knowing who its borrowers are, and each borrower is responsible for knowing the loan(s) that it has.</p> <p>The opposite of a bi-directional association is a navigable association (see below). Which example, the bi-directional one or the navigable one is correct? The answer: it depends on what the domain experts say.</p>
Class	<p>A class is a representation of an abstraction. It contains data and operations on that data.</p> <p>Example: for a loan processing system, candidate classes might be Borrower, Property, Loan Product, etc.</p>
Conceptual Class	<p>A class that represents an abstraction from the <i>problem</i> domain.</p> <p>By contrast, a Design Class represents an abstraction from the <i>design</i> domain.</p>
Composition Relationship	<p>A special type of whole-part relationship that says that if the parent class goes away, it's children do also.</p> <p>Example: For an airplane, if the plane goes away (is destroyed), so are the wings.</p>

Term	Definition
	<p>By contrast, see the definition for an Aggregation Relationship.</p> <p>See the sample domain model in Appendix A for additional information.</p>
Design Model	<p>A set of artifacts that captures, through various views of the system the software components that satisfy the requirements from the problem domain.</p> <p>Note: many of the artifacts in the Design Model are also used in the Analysis Model, however, their respective audiences are quite different. The former provides value to both Business and Engineering staff while the latter provides value solely to the Engineering staff.</p>
Design Class	<p>A class that represents an abstraction from the <i>design</i> domain, which may or may not correlate exactly to the conceptual classes in the problem domain.</p> <p>By contrast, a Conceptual Class represents an abstraction from the <i>problem</i> domain.</p>
Domain Model	<p>A Domain Model captures the conceptual classes and their associations. It is captured graphically using UML.</p>
Inheritance Relationship	<p>A type of association that communicates that one class is a specialization of another class.</p> <p>Example: A loan could be specialized by a consumer loan and a commercial loan. In this context, the loan is called a base class, and the commercial/consumer loans are called the inherited (or specialized) subclasses. Each subclass contains some of the data and operations of the base class, along with additional data and/or operations as well.</p>
Multiplicity	<p>Multiplicity communicates how many classes participate in an association.</p> <p>Example: A library has one to many books and any given book is associated with one and only one library.</p>
Navigable Association	<p>An association that communicates that one class is responsible for navigating through another class, but not the other way around. It is used when the modeler feels that it is necessary to refine an association, which by default is bi-directional.</p> <p>Example: A loan is associated with one to many borrowers. The loan is responsible for knowing who its borrowers are, but each borrower is not responsible for knowing the loan(s) that it has.</p> <p>Note: contrast this example with a bi-directional association. Which version is correct? The answer depends on what the domain experts say.</p>
OMG	<p>Object Management Group. This is a vendor-neutral object-oriented standards organization that has ownership of the UML specification.</p>
Parent-Child Relationship	<p>Relationship that indicates inheritances: Parents also can be called ancestor or base class and children can be called descendants or derived classes.</p> <p>Example: A Salesperson class could conceivably be inherited by 2 sub-classes: Commissioned Salesperson and Non-Commissioned Salesperson.</p>
Role	<p>In addition to assigning a name to an association, a modeler could also assign it a role. A role could be assigned to either end of an association.</p> <p>Example: A Person class could have an association to itself. One end of the association would be assigned the role of manager while the other end would be assigned the role of worker.</p>
RUP	<p>Rational Unified Process. RUP is a software development process based on best</p>

Term	Definition
	practices such as iterative development, use case driven, and architecture centric.
Specialization-Generalization Relationship	Another term used to express an Inheritance relationship.
State Diagram	Captures the behavioral changes that occur in a conceptual class in response to significant events.
UML	Unified Modeling Language. An industry standard modeling convention that when followed properly allows business users and engineering staff to communicate using a consistent set of diagrams.
Whole-Part Relationship	Another term used to express a Parent-Child Relationship.

1.4 References

- Rational Unified Process, version 5.5. Available online at <http://magellan.gmacrfc.com/rup/>. This is the official source for RUP. As good as it is for this purpose, it unfortunately provides little guidance to domain modeling standards add conventions.
- Fowler, Martin, “UML Distilled: A Brief Guide to the Standard Object Modeling Language”, Addison Wesley, 2nd edition, 2000. This book is very well written and it provides practical guidance to the most widely used UML diagrams.
- Larman, Craig, “Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and the Unified Process, Addison Wesley, 2nd edition, 2002. Described as perhaps the best book on OO design. Also very well written.
- Martin, Robert, “Designing Object-Oriented C++ Applications Using C++”, Prentice Hall, 1995.
- Rosenberg, Doug and Scott, Kendall, “Applying Use-Case Driven Modeling with UML”, Addison Wesley, 2001. The authors describe a methodology called ICONIX, a process similar to RUP but with more emphasis on domain models than on use cases. Provides some useful domain modeling guidelines, but falls short everywhere else. Not recommended.

2 Domain Modeling Guidelines

2.1 Introduction

The Domain Model is a set of one or more diagrams that capture the most important types of concepts, such as business entities or business events for the system being developed.

This section provides general domain modeling guidelines. Readers may find it helpful to refer to the sample Domain Model provided in Appendix A.

2.2 Development Guidelines

- The Domain Model captures the most important conceptual classes and their associations. By convention, it refrains from using technical detail or other information that is not deemed essential to its purpose.
- Once developed, the Domain Model remains basically static. That's because if it is done right, i.e. it captures the important abstractions, which rarely change over time. By contrast, a Design Model is more dynamic, and is more apt to change over time.
- Once the Domain Model is approved for all of the use cases, the decision to update it requires careful consideration. Namely, a change to the Domain Model might be indicative of a major scope change to the project.

2.3 Class Level Guidelines

- Only conceptual classes that are essential to the overall understanding of the domain are captured. By definition, this means that design-time and implementation classes are not captured.
- Conceptual classes are extracted from nouns in the use cases. Other sources for class names include the vision document, glossary, etc.
- Classes are named as singular nouns, or noun pairs. Example: Sale, Sale LineItem.
- Unless domain experts absolutely demand it, no operations are to be captured.
- Visibility of attributes and operations is not a concern.

2.4 Attribute Guidelines

- Only essential data (attributes) are captured for each conceptual class.
- If an attribute cannot be expressed as a simple number or a string of characters, express it as a Conceptual Class. Exceptions are common classes that occur frequently such as Phone Number, Date or DateTime, etc.
- If the attribute has a unit, add the unit to the model. Example: If there is an attribute called *price*, one might want to ask what the currency is. To avoid this confusion, add another attribute called *unit*. Another alternative is to create a new conceptual class and call it *Money*.

2.5 Association Guidelines

- Only capture the associations that are relevant to the current and/or prior set of use cases. For example, a Bank Customer can have many ATM cards. For an ATM system, this is not an association that needs to be captured, but for a Bank Customer Sales Tracking system, this association might need to be captured.
- All associations must be named.
- Each end-point of an association must be given a multiplicity.
- If the model contains navigability, generalization, whole-part, composition, or aggregation relationships, be prepared to justify their existence from the point of view that their use enriches model semantics versus the increased model clutter.

- Associations are relationships expressed in the use cases or perhaps the metadata repository. Other sources for associations include the vision document, conceptual blueprint, etc.
- If the model contains roles, be prepared to justify their existence from the point of view that their use enriches model semantics versus increase model cluttering.
- Associations are named as type, verb, or verb pairs. Example: Contains, Has, Paid, Paid-By.
- By convention, associations are read from left-to-right or top to bottom. If this not the case, use the arrowhead symbol.

3 State Diagrams

3.1 Introduction

Given the sheer volume, not every conceptual class is required to have an associated state diagram. Only those conceptual classes critical to an understanding of the problem domain are required. There are no hard and fast rules that dictate which conceptual classes are critical and which ones are not. The degree of coverage is to be determined by consensus between Domain Experts, System Analysts, and Architects.

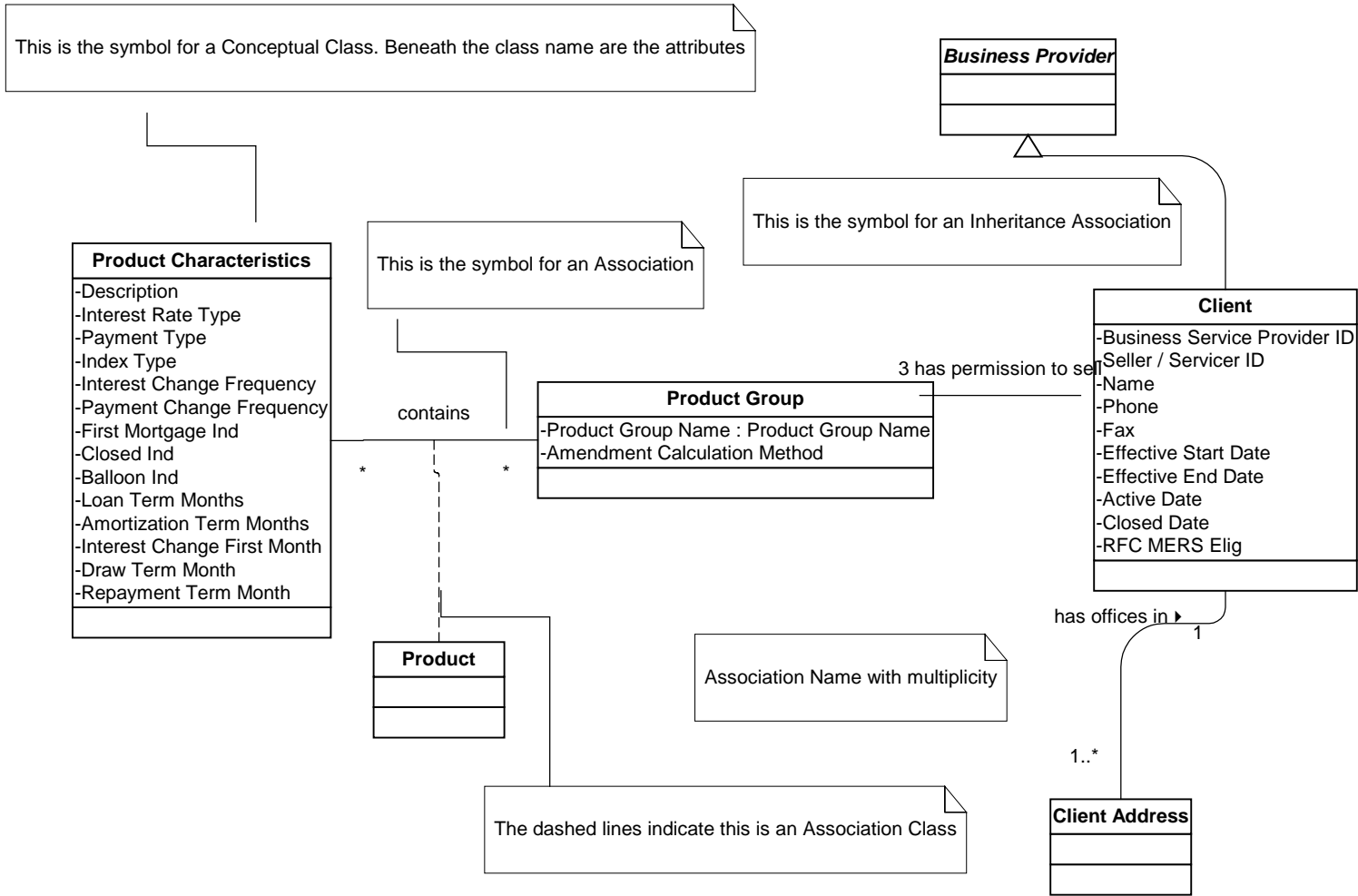
This section provides general diagramming guidelines. Readers may find it helpful to refer to the sample diagram provided in Appendix B.

3.2 Development Guidelines

- At the Analysis level, state diagrams capture only the most important states and transitions. What defines important is of course subjective, but in general, the states and transitions must trace to events and/or states from the problem domain.
- There must be exactly one and only one start state.
- There must be exactly one and only one final state.
- With the exception of the start state and the final state, each intermediate state must have at least one inbound transition and at least one outbound transition.
- State names, in order of preference, are as follows:
 - Descriptive names, i.e. active/inactive, on/off, overdrawn/delinquent, etc.
 - Past tense states, i.e. opened/closed, activated/de-activated, etc.
 - Present tense states, i.e. opening/closing, etc.
- If the diagram contains advanced concepts such as nested states and guarded states, be prepared to justify their existence.
- Use transitions to self (see appendix B) only if their existence can be justified.
- Each transition must be labeled with the events that cause it, with the exception of automatic transitions.
- For each state in the diagram, there must be at least one sequence of events that will lead to it from the start state.
- For each state in the diagram, there must be at least one sequence of events that will lead from it to the end state.
- There must be exactly one transition from the start state to exactly one non-end state.
- Event names are present tense verbs or verb phrases: they never describe the transition.

Sample Domain Model

Client



Appendix B – Sample State Diagram

Sample State Diagram for a Checking Account

