Chapter – 3

Object-Oriented Analysis (OOA) Methodologies

Objectives:

At the end of this Chapter, the students will be able to:

1. Define artifacts of business Model.
2. Define Requirement Engineering and Analyze Domain Engineering.
3. Analyze any scenario in OOAD perspective
4. Discuss different types of software analysis techniques.
5. Understand Behavior model and Requirement model architectures.
6. Understand Use Case Analysis.
7. Compare various Object-Oriented Methodologies.
8. Understands new Unified efforts for the process and system analysis.

Introduction

Maybe you can ask a question - why should we need an analysis for a software project?

There is an analogy: if you want to build a house, you will also need a plans before you make a first dig in the ground. And usually, you want to know how the house will look like in advance and that is why you hire architects and engineers.

Well, with software it is the same.

Before you start software project or before you ask somebody to make it for you, you will need a pretty good idea of what you expect of it. Who will be using it, what functionality should it have, what data will be inputted and what output are expected.

Sometimes, this might be quite challenging to communicate between the customer representatives and supplier employees. Mostly because the first ones are managers and the latter ones are IT professionals. Between those groups, the communication might be difficult. And that is why, all demands, expectations, interfaces, etc. should be identified and formulated into a written specification.

Based on this specification, a detailed analysis should be made by software architects, who will analyze all requirements in detail, identify and solve possible problems and sketch the system concepts. The analysis will also give answer to questions like what tools and technologies should be or is possible to use, what can be the project schedule, make better budget estimations, etc.

3.1 Business Analysis

The **Development of management theory and philosophy** considers the fundamental principles that underlie the formation and operation of a business enterprise; the nature and purpose of a business, for example, is it primarily property or a social institution; its role in society; and the moral obligations that pertain to it. The subject is important to business and management, and is closely related to business ethics and political economy. It is influenced significantly by philosophy, ethics, and economic theory.

One must draw an important distinction between the philosophy of business and business philosophy, which is an appellation that one often hears in the business world.

The purpose of **System Requirements Analysis** is to obtain a thorough and detailed understanding of the business need as defined in Project Origination and captured in the business Case, and to break it down into discrete requirements, which are then clearly defined, reviewed and agreed upon with the Customer Decision-Makers. During System

Objectives

Requirements Analysis, the framework for the application is developed, providing the foundation for all future design and development efforts.

System Requirements Analysis can be a challenging phase, because all of the major
Customers and their interests are brought into the process of determining requirements.

In this chapter we are going to discuss about these issues to help system designer to comprehend the overall system.

3.1.1 What Is Business?

A **business** (also known as **enterprise** or **firm**) is an organization engaged in the trade of goods, services, or both to consumers. Businesses are predominant in capitalist economies, where most of them are privately owned and administered to earn profit to increase the wealth of their owners. Businesses may also be not-for-profit or state-owned.

A business owned by multiple individuals may be referred to as a company, although that term also has a more precise meaning.

The etymology of "business" relates to the state of being busy either as an individual or society as a whole, doing commercially viable and profitable work. The term "business" has at least three usages, depending on the scope — the singular usage to mean a particular organization; the generalized usage to refer to a particular market sector, "the music business" and compound forms such as agribusiness; and the broadest meaning, which encompasses all activity by the community of suppliers of goods and services.

However, the exact definition of business, like much else in the philosophy of business, is a matter of debate and complexity of meanings.

3.1.2 Classifications of Business

- **Agriculture and mining businesses** are concerned with the production of raw material, such as plants or minerals.

- **Financial businesses** include banks and other companies that generate profit through investment and management of capital.
- Information businesses generate profits primarily from the resale of intellectual property and include movie studios, publishers and packaged software companies.
- Manufacturers produce products, from raw materials or component parts, which they then sell at a profit. Companies that make physical goods, such as cars or pipes, are considered manufacturers.
- Real estate businesses generate profit from the selling, renting, and development of properties, homes, and buildings.
- Retailers and distributors act as middle-men in getting goods produced by manufacturers to the intended consumer, generating a profit as a result of providing sales or distribution services. Most consumer-oriented stores and catalog companies are distributors or retailers.
- Service businesses offer intangible goods or services and typically generate a profit by charging for labor or other services provided to government, other businesses, or consumers. Organizations ranging from house decorators to consulting firms, restaurants, and even entertainers are types of service businesses.
- Transportation businesses deliver goods and individuals from location to location, generating a profit on the transportation costs.
- Utilities produce public services such as electricity or sewage treatment, usually under a government charter.

3.2 HUMAN RESOURCES

Human resources is a term used to describe the individuals who make up the workforce of an organization, although it is also applied in labor economics to, for example, business sectors or even whole nations. Human resources is also the name of the function within an organization charged with the overall responsibility for implementing strategies and policies relating to the management of individuals (i.e. the human resources). This function title is often abbreviated to the initials **"HR".**

Human resources is a relatively modern management term, coined as late as the 1960s. The origins of the function arose in organizations that introduced 'welfare management' practices and also in those that adopted the principles of 'scientific management'. From these terms emerged a largely administrative management activity, coordinating a range of worker related processes and becoming known, in time, as the 'personnel function'.

Human resources progressively became the more usual name for this function, in the first instance in the United States as well as multinational or international corporations, reflecting the adoption of a more quantitative as well as strategic approach to workforce management, demanded by corporate management to gain a competitive advantage, utilizing limited skilled and highly skilled workers

3.2.1 Business Analyst (BA)

A Business Analyst (BA) acts as a face of Customer to the Development team, most of the time. A Business Analyst should be credible enough and the team should have absolute faith in him/her.

Development team should be able to ask any question regarding the system and they should believe in the answers that BAs provide. If they start having doubts on the answers BAs provide they may get tempted to develop something that is not needed by the business or spend extra time in clarifying the doubt from various sources.

The development team should trust a BA. Following are the important practices that A BA has to do:

- Interact with the developers regularly and keep asking them if they have any doubts. The idea is not to overdo it as they may get a feeling that you are trying to judge their work. Keep it simple and just make sure that they know you are there if they need any clarifications in the requirements.
- Make sure that the development team runs through the requirements before they start with the implementation. Do it on module-to-module basis, plan with the Project
- Managers and Team Leads. Make sure they keep these sessions as informal as possible and try to make them understand the business pain points rather than teaching them (as they may switch off).
- Encourage the team to approach you for any clarification in the requirements. When they approach you make sure you clarify their issues or get the issues raised to correct person, if you are not the right one.

There are at least four tiers of business analysis:

1. Planning Strategically – The analysis of the organization's strategic business needs
2. Operating/Business Model Analysis – The definition and analysis of the organization's policies and market business approaches.
3. Process Definition and Design – The business process modeling (often developed through process modeling and design).
4. IT/Technical Business Analysis – The interpretation of business rules and requirements for technical systems (generally IT)

3.2.2 System Analyst

It is a good idea to explain the business side to the developers and also let them know about the domain, as you have that knowledge. Have these talks at non-work timings like lunch, coffee or

while traveling. Make sure you don't come out as a person who is bragging about his knowledge but as a person who is genuinely helping. Keep it honest; if you are not comfortable don't try it. A **systems analyst** researches problems, plans solutions, recommends software and systems, and coordinates development to meet business or other requirements. They will be familiar with multiple varieties of programming languages, operating systems, and computer hardware platforms. Because they often write user requests into technical specifications, the systems analysts are the liaisons between vendors and information technology professionals.[1] They may be responsible for developing cost analysis, design considerations, and implementation timelines. A systems analyst may:

- Plan a system flow from the ground up.
- Interact with customers to learn and document requirements that are then used to produce business requirements documents.
- Write technical requirements from a critical phase.
- Interact with designers to understand software limitations.
- Help programmers during system development, ex: provide use cases, flowcharts or even Database design.
- Perform system testing.
- Deploy the completed system.
- Document requirements or contribute to user manuals.
- Whenever a development process is conducted, the system analyst is responsible for designing components and providing that information to the developer

3.3 Domain Analysis

- In software engineering, **domain analysis**, or **product line analysis**, is the process of analyzing related software systems in a domain to find their common and variable parts. It is a model of wider business context for the system. The term was coined in the early 1980s by James Neighbors Domain analysis is the first phase of domain engineering. It is a key method for realizing systematic software reuse.
- Domain analysis produces domain models using methodologies such as domain specific languages, *feature tables*, *facet tables*, *facet templates*, and *generic architectures*, which describe all of the systems in a domain. Several methodologies for domain analysis have been proposed.
- The products, or "artifacts", of a domain analysis are sometimes object-oriented models (e.g. represented with the Unified Modeling Language (UML)) or data models represented with entity-relationship diagrams (ERD). Software developers can use these models as a basis for the implementation of software architectures and applications. This approach to domain analysis is sometimes called model-driven engineering.
- Imagine if everyone on your team was talking a different language. Let's say you're speaking German, your teammate is speaking French, and someone else is speaking Swahili. Every time someone speaks, people glean whatever slivers of meaning they can, and then nod as if they've understood perfectly. They then walk away with a completely wrong interpretation of what the speaker was really trying to say.

- In virtually all IT projects, the problem of miscommunication is rampant, but it's rarely noticed because everybody *thinks* they're speaking the same language. They're not.

.

 **The domain model is a live, collaborative artifact. It is refined and updated throughout the project, so that it always reflects the current understanding of the problem space.**

3.3.1 Top 10 Domain Modeling Guidelines

Top ten principles that can be followed as list of guidelines are:

1. Don't put screens and other GUI-specific classes on your domain model.

2. Don't expect your final class diagrams to precisely match your domain model, but there should be some resemblance between them.

3. Do initial domain model before you write use cases, to avoid name ambiguity?

4. Use the domain model as a project glossary.

5. Don't confuse an object (which represents a single instance) with a database table (Which contains a collection of things)?

6. Don't mistake your domain model for a data model.

7. Organize your classes around key abstractions in the problem domain.

8. Limit your initial domain modeling efforts to a couple of hours.

9. Use generalization (is-a) and aggregation (has-a) relationships to show how the objects relate to each other.

10. Focus on real-world (problem domain) objects.

3.4 Requirement Analysis

3.4.1 What is a Requirement?

- A condition or capability needed by a user to solve a problem or achieve an objective.
- A condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed document.

The set of all requirements forms the basis for subsequent development of the system or system component

3.4.2 What is Requirements Analysis?

Requirements analysis in systems engineering and software engineering, encompasses those tasks that go into determining the needs or conditions to meet for a new or altered product, taking account of the possibly conflicting requirements of the various stakeholders, such as beneficiaries or users.

Requirements analysis is critical to the success of a development project. Requirements must be documented, actionable, measurable, testable, traceable, related to identified business needs or opportunities, and defined to a level of detail sufficient for system design. Requirements can be architectural, structural, behavioral, functional, and nonfunctional.
Conceptually, requirements analysis includes three types of activity:

- Eliciting requirements: the task of communicating with customers and users to determine what their requirements are. This is sometimes also called requirements gathering.
- Analyzing requirements: determining whether the stated requirements are unclear, incomplete, ambiguous, or contradictory, and then resolving these issues.
- Recording requirements: Requirements might be documented in various forms, such as natural-language documents, use cases, user stories, or process specifications.

Requirements analysis can be a long and arduous process during which many delicate psychological skills are involved. New systems change the environment and relationships between people, so it is important to identify all the stakeholders, take into account all their needs and ensure they understand the implications of the new systems. Analysts can employ several techniques to elicit the requirements from the customer. Historically, this has included such things as holding interviews, or holding focus groups (more aptly named in this context as requirements workshops) and creating requirements lists. More modern techniques include prototyping, and use cases. Where necessary, the analyst will employ a combination of these methods to establish the exact requirements of the stakeholders, so that a system that meets the business needs is produced.

3.4.3 Types of Requirements

1. Customer Requirements

Statements of fact and assumptions define that the expectations of the system in terms of mission objectives, environment, constraints, and measures of effectiveness and suitability (MOE/MOS). The customers are those that perform the eight primary functions of systems engineering, with special emphasis on the operator as the key customer. Operational requirements will define the basic need and, at a minimum, answer the questions posed in the following listing.

- Operational distribution or deployment: Where will the system be used?

- Mission profile or scenario: How will the system accomplish its mission objective?
- Performance and related parameters: What are the critical system parameters to accomplish the mission?
- Utilization environments: How are the various system components to be used?
- Effectiveness requirements: How effective or efficient must the system be in performing its mission?
- Operational life cycle: How long will the system be in use by the user?
- Environment: What environments will the system be expected to operate in an effective manner?

2. Architectural Requirements

Architectural requirements explain what has to be done by identifying the necessary system architecture of a system.

3. Structural Requirements

Structural requirements explain what has to be done by identifying the necessary structure of a system.

4. Behavioral Requirements

Behavioral requirements explain what has to be done by identifying the necessary behavior of a system.

5. Functional Requirements

Functional requirements explain what has to be done by identifying the necessary task, action or activity that must be accomplished. Functional requirements analysis will be used as the toplevel functions for functional analysis.

6. Non-functional Requirements

Non-functional requirements are requirements that specify criteria that can be used to judge the operation of a system, rather than specific behaviors.

7. Performance Requirements

The extent to which a mission or function must be executed; generally measured in terms of quantity, quality, coverage, timeliness or readiness. During requirements analysis, performance (how well does it have to be done) requirements will be interactively developed across all identified functions based on system life cycle factors; and characterized in terms of the degree of certainty in their estimate, the degree of criticality to system success, and their relationship to other requirements.

8. Design Requirements

The "build to," "code to," and "buy to" requirements for products and "how to execute" requirements for processes expressed in technical data packages and technical manuals.

9. Derived Requirements

The requirements that are implied are transformed from higher-level requirement. For example, a requirement for long range or high speed may result in a design requirement for low weight.

10. Allocated Requirements

A requirement that is established by dividing or otherwise allocating a high-level requirement into multiple lower-level requirements is called Allocated Requirement.

3.4.4 Requirements Analysis Issues of Stakeholders

User/customer issues

- Users do not understand what they want or users don't have a clear idea of their requirements
- Users will not commit to a set of written requirements
- Users insist on new requirements after the cost and schedule have been fixed
- Communication with users is slow
- Users often do not participate in reviews or are incapable of doing so
- Users are technically unsophisticated
- Users do not understand the development process
- Users do not know about present technology

This may lead to the situation where user requirements keep changing even when system or product development has been started.

Engineer/developer issues

Possible problems caused by engineers and developers during requirements analysis are:

- Technical personnel and end-users may have different vocabularies. Consequently, they may wrongly believe they are in perfect agreement until the finished product is supplied.
- Engineers and developers may try to make the requirements fit an existing system or model, rather than develop a system specific to the needs of the client.
- Analysis may often be carried out by engineers or programmers, rather than personnel with the people skills and the domain knowledge to understand a client's needs properly.

3.4.5 What is Requirements Engineering (RE)?

RE is concerned with identifying the purpose of a software system, and the contexts in which it will be used.

RE acts as the bridge between the real world needs of users, customers and other constituencies, affected by a software system and the capabilities and opportunities afforded by software-intensive technologies.

Systematic requirements analysis is also known as requirements engineering. It is sometimes referred to loosely by names such as requirements gathering, requirements capture, or requirements specification. The term requirements analysis can also be applied specifically to the analysis proper, as opposed to elicitation or documentation of the requirements, for instance. Requirements engineering can be divided into discrete chronological steps:

- Requirements elicitation,
- Requirements analysis and negotiation,
- Requirements specification,
- System modeling,
- Requirements validation,
- Requirements management.

3.4.6 Stakeholder identification

Stakeholders (SH) are people or organizations (legal entities such as companies, standards bodies) that have a valid interest in the system. They may be affected by it either directly or indirectly. A major new emphasis in the 1990s was a focus on the identification of stakeholders. It is increasingly recognized that stakeholders are not limited to the organization employing the analyst.
Other stakeholders include:

- anyone who operates the system (normal and maintenance operators)
- anyone who benefits from the system (functional, political, financial and social beneficiaries)
- Anyone involved in purchasing or procuring the system. In a mass-market product organization, product management, marketing and sometimes sales act as surrogate consumers (mass-market customers) to guide development of the product
- organizations which regulate aspects of the system (financial, safety, and other regulators)
- people or organizations opposed to the system (negative stakeholders; see also Misuse case)
- organizations responsible for systems which interface with the system under design
- those organizations who integrate horizontally with the organization for whom the analyst is designing the system

3.4.7 Stakeholder interviews

Stakeholder interviews are a common technique used in requirement analysis. Though they are generally idiosyncratic in nature and focused upon the perspectives and perceived needs of the stakeholder, very often without larger enterprise or system context, this perspective deficiency has the general advantage of obtaining a much richer understanding of the stakeholder's unique business processes, decision-relevant business rules, and perceived needs. Consequently this technique can serve as a means of obtaining the highly focused knowledge that is often not elicited in Joint Requirements Development sessions, where the stakeholder's attention is compelled to assume a more cross-functional context. Moreover, the in-person nature of the interviews provides a more relaxed environment where lines of thought may be explored at length.

3.4.8 Prototypes

In the mid-1980s, prototyping was seen as the best solution to the requirements analysis problem. Prototypes are Mockups of an application. Mockups allow users to visualize an application that hasn't yet been constructed. Prototypes help users get an idea of what the system will look like, and make it easier for users to make design decisions without waiting for the system to be built. Major improvements in communication between users and developers were often seen with the introduction of prototypes. Early views of applications led to fewer changes later and hence reduced overall costs considerably.

However, over the next decade, while proving a useful technique, prototyping did not solve the requirements problem:

- Managers, once they see a prototype, may have a hard time understanding that the finished design will not be produced for some time.
- Designers often feel compelled to use patched together prototype code in the real system, because they are afraid to 'waste time' starting again.
- Prototypes principally help with design decisions and user interface design. However, they cannot tell you what the requirements originally were.
- Designers and end-users can focus too much on user interface design and too little on producing a system that serves the business process.
- Prototypes work well for user interfaces, screen layout and screen flow but are not much useful for batch or asynchronous processes which may involve complex database updates and/or calculations.

Prototypes can be flat diagrams (often referred to as wireframes) or working applications using synthesized functionality.

3.4.9 Use Case

A use case is a technique for documenting the potential requirements of a new system or software change. Each use case provides one or more *scenarios* that convey how the system should interact with the end-user or another system to achieve a specific business goal. Use cases

typically avoid technical jargon, preferring instead the language of the end-user or domain expert. Use cases are often co-authored by requirements engineers and stakeholders.

Use cases are deceptively simple tools for describing the behavior of software or systems.
A use case contains a textual description of all of the ways which the intended users could work with the software or system. Use cases do not describe any internal workings of the system, nor do they explain how that system will be implemented. They simply show the steps that a user follows to perform a task. All the ways that users interact with a system can be described in this manner.

3.5 Behavioral Model

What is a Behavioral Model?

All behavioral models really do is describe the control structure of a system. This can be things like:

- Sequence of operations
- Object states
- and Object interactions

Furthermore, this modeling layer can also be called Dynamic Modeling. The activity of creating a behavioral model is commonly known as behavioral modeling. As well as this, a system should also only have one behavioral model – much like functional modeling.

How do we represent them?

We represent behavioral models with dynamic diagrams.

For example:

- Sequence Diagrams (Design)
- Communication Diagrams *or* **collaboration diagram**
- State Diagrams *or* **state machine diagram** *or* **state chart**

For consistency we will use communication diagram and state diagram

If we have both a sequence diagram AND communication diagram, then together these are known as interaction diagrams, this is because they both represent how objects interact with one another using messages.

3.6 Requirements Statement for Example ATM System

The software to be designed will control a simulated automated teller machine (ATM) having a magnetic stripe reader for reading an ATM card, a customer console (keyboard and display) for interaction with the customer, a slot for depositing envelopes, a dispenser for cash (in multiples of $20), a printer for printing customer receipts, and a key-operated switch to allow an operator to start or stop the machine. The ATM will communicate with the bank's computer over an

appropriate communication link. (The software on the latter is not part of the requirements for this problem.)

The ATM will service one customer at a time. A customer will be required to insert an ATM card and enter a personal identification number (PIN) - both of which will be sent to the bank for validation as part of each transaction. The customer will then be able to perform one or more transactions. The card will be retained in the machine until the customer indicates that he/she desires no further transactions, at which point it will be returned - except as noted below.

The ATM must be able to provide the following services to the customer:

1. A customer must be able to make a cash withdrawal from any suitable account linked to the card, in multiples of $20.00. Approval must be obtained from the bank before cash is dispensed.
2. A customer must be able to make a deposit to any account linked to the card, consisting of cash and/or checks in an envelope. The customer will enter the amount of the deposit into the ATM, subject to manual verification when the envelope is removed from the machine by an operator. Approval must be obtained from the bank before physically accepting the envelope.
3. A customer must be able to make a transfer of money between any two accounts linked to the card.
4. A customer must be able to make a balance inquiry of any account linked to the card.

A customer must be able to abort a transaction in progress by pressing the Cancel key instead of responding to a request from the machine.

The ATM will communicate each transaction to the bank and obtain verification that it was allowed by the bank. Ordinarily, a transaction will be considered complete by the bank once it has been approved. In the case of a deposit, a second message will be sent to the bank indicating that the customer has deposited the envelope. (If the customer fails to deposit the envelope within the timeout period, or presses cancel instead, no second message will be sent to the bank and the deposit will not be credited to the customer.)

If the bank determines that the customer's PIN is invalid, the customer will be required to re-enter the PIN before a transaction can proceed. If the customer is unable to successfully enter the PIN after three tries, the card will be permanently retained by the machine, and the customer will have to contact the bank to get it back.

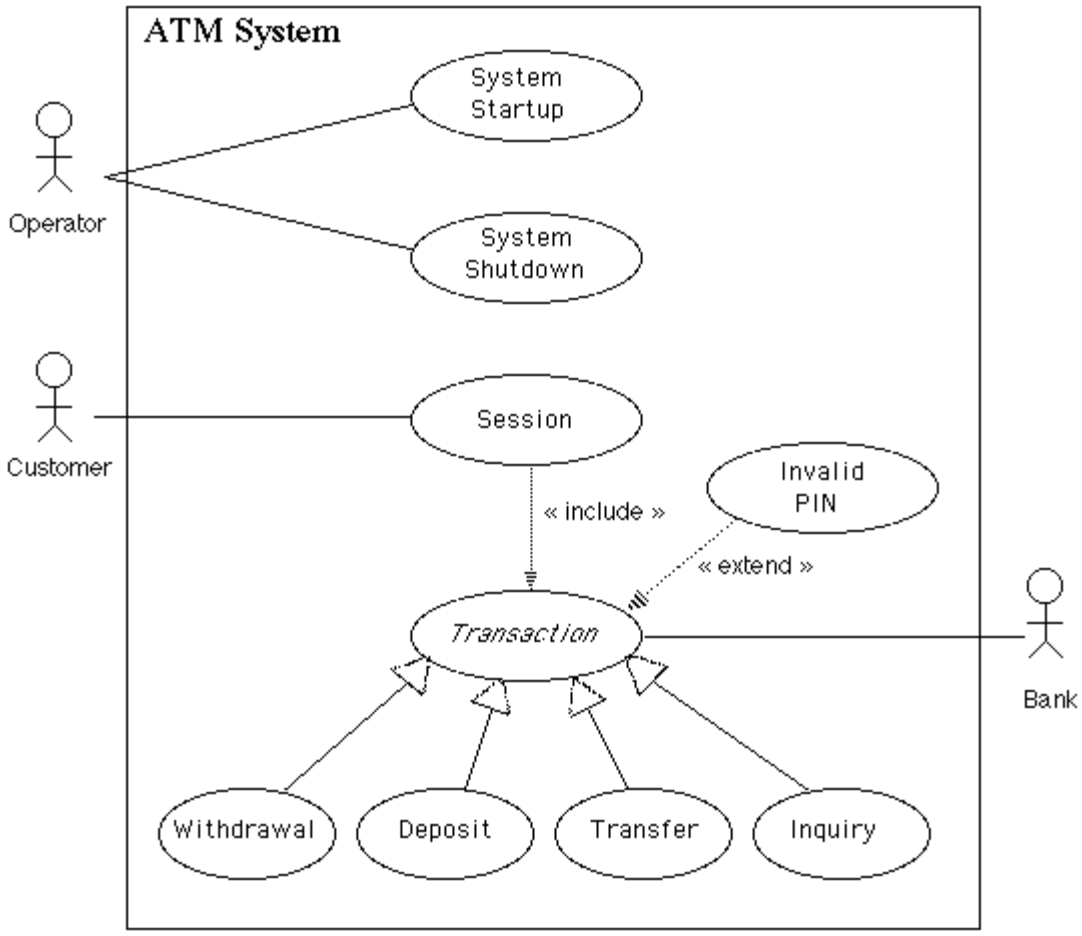
If a transaction fails for any reason other than an invalid PIN, the ATM will display an explanation of the problem, and will then ask the customer whether he/she wants to do another transaction.

The ATM will provide the customer with a printed receipt for each successful transaction, showing the date, time, machine location, type of transaction, account(s), amount, and ending and available balance(s) of the affected account ("to" account for transfers).

The ATM will have a key-operated switch that will allow an operator to start and stop the servicing of customers. After turning the switch to the "on" position, the operator will be required to verify and enter the total cash on hand. The machine can only be turned off when it is not servicing a customer. When the switch is moved to the "off" position, the machine will shut down, so that the operator may remove deposit envelopes and reload the machine with cash, blank receipts, etc.

The ATM will also maintain an internal log of transactions to facilitate resolving ambiguities arising from a hardware failure in the middle of a transaction. Entries will be made in the log when the ATM is started up and shut down, for each message sent to the Bank (along with the response back, if one is expected), for the dispensing of cash, and for the receiving of an envelope. Log entries may contain card numbers and dollar amounts, but for security will *never* contain a PIN.

Use Cases for Example ATM System

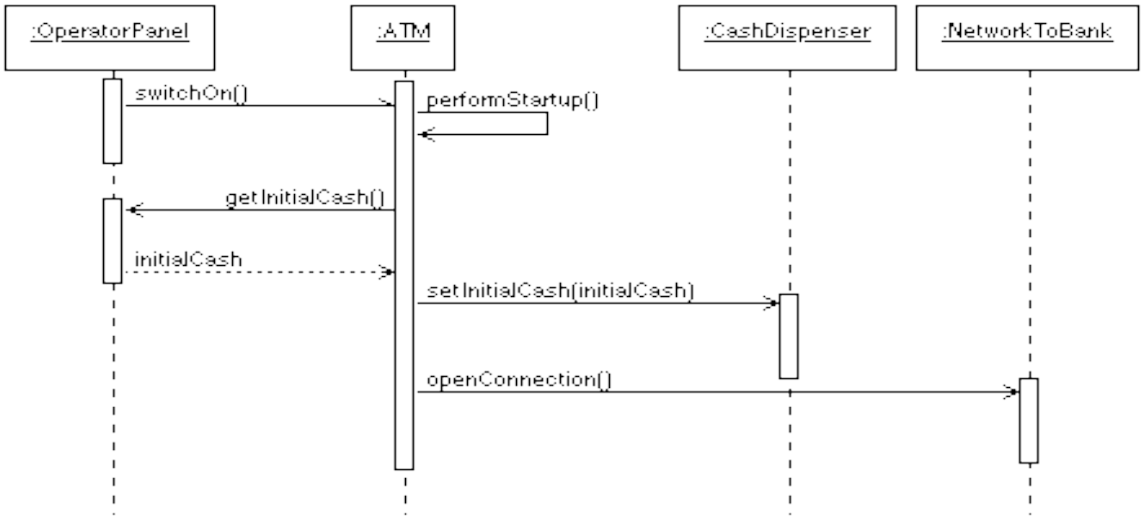


Flows of Events for Individual Use Cases

System Startup Sequence diagram

The system is started up when the operator turns the operator switch to the "on" position. The operator will be asked to enter the amount of money currently in the cash dispenser, and a connection to the bank will be established. Then the servicing of customers can begin.

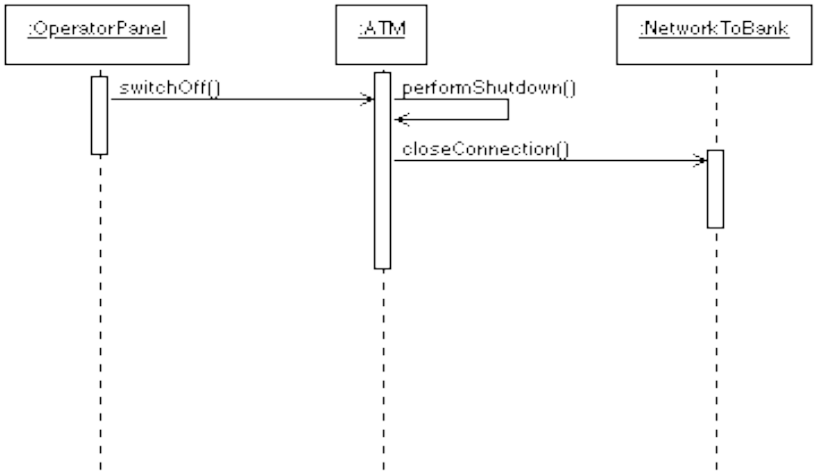
System Startup Sequence Diagram



System Shutdown Sequence diagram

The system is shut down when the operator makes sure that no customer is using the machine, and then turns the operator switch to the "off" position. The connection to the bank will be shut down. Then the operator is free to remove deposited envelopes, replenish cash and paper, etc.

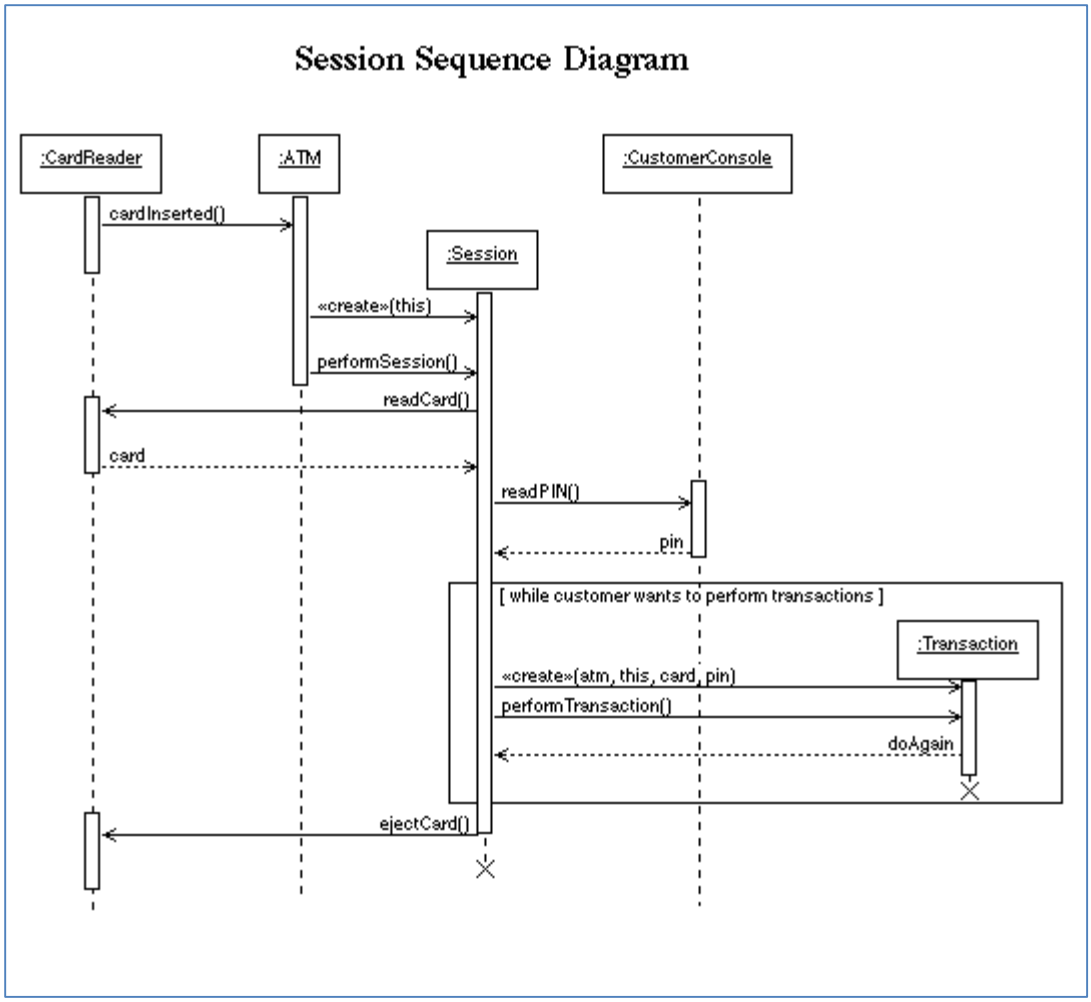
System Shutdown Sequence Diagram



Session Sequence diagram

A session is started when a customer inserts an ATM card into the card reader slot of the machine. The ATM pulls the card into the machine and reads it. (If the reader cannot read the card due to improper insertion or a damaged stripe, the card is ejected, an error screen is displayed, and the session is aborted.) The customer is asked to enter his/her PIN, and is then allowed to perform one or more transactions, choosing from a menu of possible types of transaction in each case. After each transaction, the customer is asked whether he/she would like to perform another. When the customer is through performing transactions, the card is ejected from the machine and the session ends. If a transaction is aborted due to too many invalid PIN entries, the session is also aborted, with the card being retained in the machine.

The customer may abort the session by pressing the Cancel key when entering a PIN or choosing a transaction type.



Transaction Use Case

Note: Transaction is an abstract generalization. Each specific concrete type of transaction implements certain operations in the appropriate way. The flow of events given here describes the behavior common to all types of transaction. The flows of events for the individual types of transaction (withdrawal, deposit, transfer, inquiry) give the features that are specific to that type of transaction.

A transaction use case is started within a session when the customer chooses a transaction type from a menu of options. The customer will be asked to furnish appropriate details (e.g. account(s) involved, amount). The transaction will then be sent to the bank, along with information from the customer's card and the PIN the customer entered.

If the bank approves the transaction, any steps needed to complete the transaction (e.g. dispensing cash or accepting an envelope) will be performed, and then a receipt will be printed. Then the customer will be asked whether he/she wishes to do another transaction.

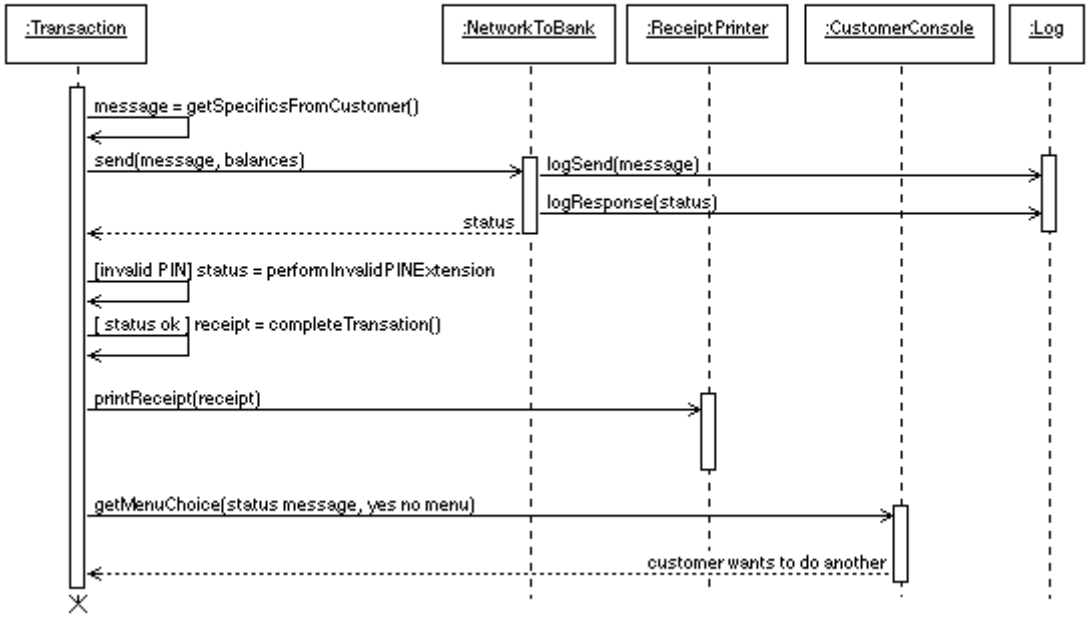
If the bank reports that the customer's PIN is invalid, the Invalid PIN extension will be performed and then an attempt will be made to continue the transaction. If the customer's card is retained due to too many invalid PINs, the transaction will be aborted, and the customer will not be offered the option of doing another.

If a transaction is cancelled by the customer, or fails for any reason other than repeated entries of an invalid PIN, a screen will be displayed informing the customer of the reason for the failure of the transaction, and then the customer will be offered the opportunity to do another.

The customer may cancel a transaction by pressing the Cancel key as described for each individual type of transaction below.

All messages to the bank and responses back are recorded in the ATM's log.

Transaction Sequence Diagram

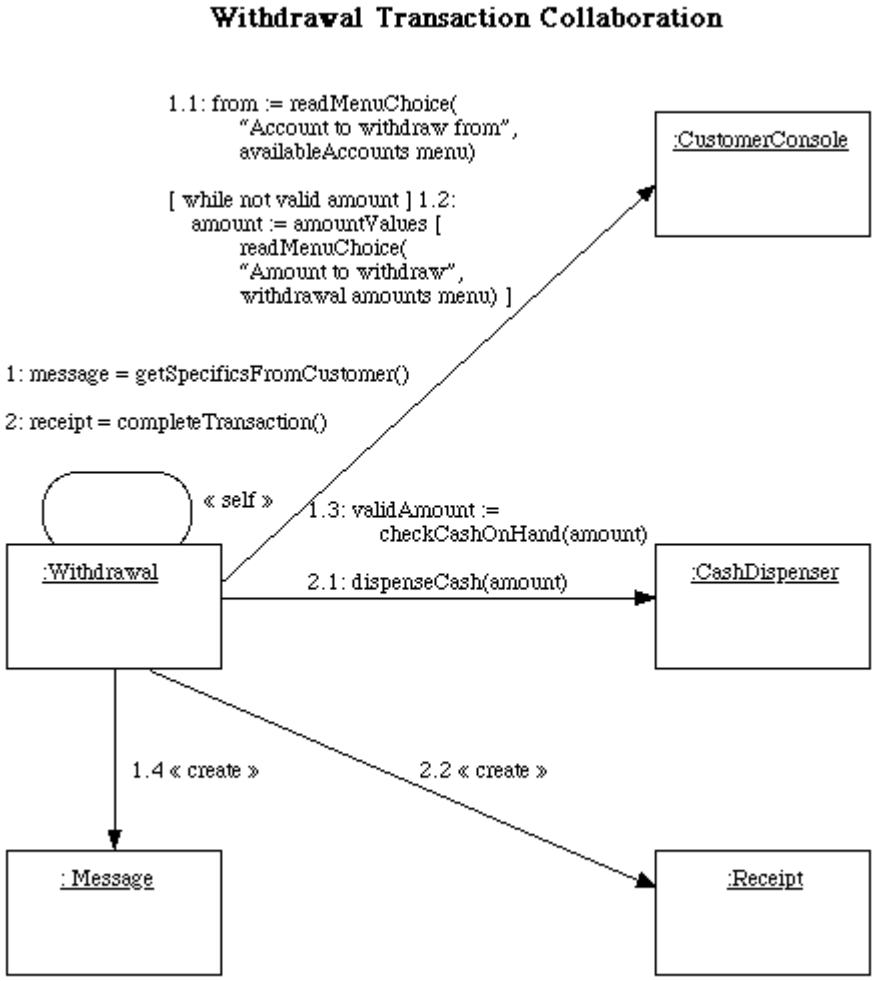


Withdrawal Transaction Use Case

A withdrawal transaction asks the customer to choose a type of account to withdraw from (e.g. checking) from a menu of possible accounts, and to choose a dollar amount from a menu of possible amounts. The system verifies that it has sufficient money on hand to satisfy the request before sending the transaction to the bank. (If not, the customer is informed and asked to enter a different amount.) If the transaction is approved by the bank, the appropriate amount of cash is

dispensed by the machine before it issues a receipt. (The dispensing of cash is also recorded in the ATM's log.)

A withdrawal transaction can be cancelled by the customer pressing the Cancel key any time prior to choosing the dollar amount.



Withdrawal funds activity diagram