11 System Conception

System conception deals with the genesis of an application. Initially some person, who understands both business needs and technology, thinks of an idea for an application. Developers must then explore the idea to understand the needs and devise possible solutions. The purpose of system conception is to defer details and understand the big picture—what need does the proposed system meet, can it be developed at a reasonable cost, and will the demand for the result justify the cost of building it?

This chapter introduces the automated teller machine (ATM) case study that threads throughout the remainder of the book.

11.1 Devising a System Concept

Most ideas for new systems are extensions of existing ideas. For example, a human relations department may have a database of employee benefit choices and require that a clerk enter changes. An obvious extension is to allow employees to view and enter their own changes. There are many issues to resolve (security, reliability, privacy, and so on), but the new idea is a straightforward extension of an existing concept.

Occasionally a new system is a radical departure from the past. For example, an online auction automates the ancient idea of buyers bidding against each other for products, but the first online auction systems were brand new software. The concept became feasible when several enabling technologies came into place: the Internet, widespread personal computer access, and reliable servers. The large customer base and low unit cost due to automation changed the nature of auctions—an online auction can sell inexpensive items and still make a profit. In addition, online systems have made the auction process concurrent and distributed.

Here are some ways to find new system concepts.

- **New functionality**. Add functionality to an existing system.

- **Streamlining**. Remove restrictions or generalize the way a system works.
- **Simplification**. Let ordinary persons perform tasks previously assigned to specialists.
- **Automation**. Automate manual processes.
- **Integration**. Combine functionality from different systems.
- **Analogies**. Look for analogies in other problem domains and see if they have useful ideas.
- **Globalization**. Travel to other countries and observe their cultural and business practices.

11.2 Elaborating a Concept

Most systems start as vague ideas that need more substance. A good system concept must answer the following questions.

- **Who is the application for?** You should clearly understand which persons and organizations are stakeholders of the new system. Two of the most important kinds of stakeholders are the financial sponsors and the end users.

  The financial sponsor are important because they are paying for the new system. They expect the project to be on schedule and within budget. You should get the financial sponsors to agree to some measure of success. You need to know when the system is complete and meets their expectations.

  The users are also stakeholders, but in another sense. The users will ultimately determine the success of the new system by an increase (or decrease) in their productivity or effectiveness. Users can help you if they are receptive and provide critical comments. They can improve your system by telling you what is missing and what could be improved. In general, users will not consider new software unless they have a compelling interest—either personal or business. You should try to help them find a vested interest in your project so that you can obtain their buy-in. If you cannot get their buy-in, you should question the need for the project and reconsider doing it.
- **What problems will it solve?** You must clearly bound the size of the effort and establish its scope. You should determine which features will be in the new system and which will not. You must reach various kinds of users in different organizations with their own viewpoints and political motivations. You must not only decide which features are appropriate, but you must also obtain the agreement of influential persons.
- **Where will it be used?** At this early stage, it is helpful to get a general idea of where the new system might be used. You should determine if the new system is mission-critical software for the organization, experimental software, or a new capability that you can deploy without disrupting the workflow. You should have a rough idea about how the new system will complement the existing systems. It is important to know if the software will be used locally or will be distributed via a network. For a commercial product, you should characterize the customer base.

- **When is it needed?** Two aspects of time are important. The first is the *feasible* time, the time in which the system can be developed within the constraints of cost and available resources. The other is the *required* time, when the system is needed to meet business goals. You must make sure that the timing expectations driven by technical feasibility are consistent with the timing the business requires. If there is a disconnect, you must initiate a dialogue between technologists and business experts to reach a solution.
- **Why is it needed?** You may need to prepare a business case for the new system if someone has not already done so. The business case contains the financial justification for the new system, including the cost, tangible benefits, intangible benefits, risk, and alternatives. You must be sure that you clearly understand the motivation for the new system. The business case will give you insight into what stakeholders expect, roughly indicate the scope, and may even provide information for seeding your models. For a commercial product, you should estimate the number of units that can be sold and determine a reasonable selling price; the revenue must cover costs and a profit.
- **How will it work?** You should brainstorm about the feasibility of the problem. For large systems you should consider the merits of different architectures. The purpose of this speculation is not to choose a solution, but to increase confidence that the problem can be solved reasonably. You might need some prototyping and experimentation.

11.2.1 The ATM Case Study

Figure 11.1 lists our original system concept for an Automated Teller Machine (ATM). We ask high-level questions to elaborate the initial concept.

Develop software so that customers can access a bank's computers and carry out their own financial transactions without the mediation of a bank employee.

Figure 11.1 System concept for an automated teller machine

- **Who is the application for?** A number of companies provide ATM products. Consequently, only a vendor or a large financial company could possibly justify the cost and effort of building ATM software.

  A vendor would be competing for customers in an established market. A large vendor could certainly enter such a market, but might find it advantageous to partner with or acquire an existing supplier. A small vendor would need some special feature to differentiate itself from the crowd and attract attention.

  It is unlikely that a financial company could justify developing ATM software just for its own use, because it would probably be more expensive than purchasing a product. If a financial company wanted special features, it could partner with a vendor. Or it might decide to create a separate organization that would build the software, sell it to the sponsoring company, and then market it to others.

For the ATM case study, we will assume that we are a vendor building the software. We will assume that we are developing an ordinary product, since deep complexities of the ATM problem domain are beyond the scope of this book.

- **What problems will it solve?** The ATM software is intended to serve both the bank and the customer. For the bank, ATM software increases automation and reduces manual handling of routine paperwork. For the customer, the ATM is ubiquitous and always available, handling routine transactions whenever and wherever the customer desires. ATM software must be easy to use and convenient so that customers will use it in preference to bank tellers. It must be reliable and secure since it will be handling money.
- **Where will it be used?** ATM software has become essential to financial institutions. Customers take it for granted that a bank will have an ATM machine. ATM machines are available at many stores, sporting events, and other locations throughout the world.
- **When is it needed?** Any software development effort is a financial proposition. The investment in development ultimately leads to a revenue stream. From an economic perspective, it is desirable to minimize the investment, maximize the revenue, and realize revenue as soon as possible. Thoughtful modeling and OO techniques are conducive to this goal.
- **Why is it needed?** There are many reasons why a vendor might decide to build a software product. If other companies are making money with similar products, there is an economic incentive to participate. A novel product could outflank competitors and lead to premium pricing. Businesses commission internal efforts for technology that is difficult to buy and critical to them. We have no real motivation to develop ATM software, other than to demonstrate the techniques in this book.
- **How will it work?** We will adopt a three-tier architecture to separate the user interface from programming logic, and programming logic from the database. In reality, the architecture is *n*-tier, because there can be any number of intermediate programming levels communicating with each other. We will discuss architecture further in the *System Design* chapter.

11.3 Preparing a Problem Statement

Once you have fleshed out the raw idea by answering the high-level questions, you are ready to write a requirements statement that outlines the goals and general approach of the desired system.

Throughout development, you should distinguish among requirements, design, and implementation. Requirements describe how a system behaves from the user's point of view. The system is considered as a black box—all we care about is its external behavior. For example, some requirements for a car are that when you press on the accelerator pedal, the car goes faster, and when you step on the brake, the car slows down. Design decisions are engineering choices that provide the behavior specified by the requirements. For example, some design decisions are how the internal linkages are routed, how the engine is controlled, and what kinds of brake pads are on the wheels. Implementation deals with the ultimate realization in programming code.

Frequently customers mix true requirements with design decisions. Usually this is a bad idea. If you separate requirements from design decisions, you preserve the freedom to change a design. Typically there are many possible ways to design a system, and you should defer a solution until you fully understand a problem.

A system concept document may include an example implementation. The purpose of the example is to show how the system could be implemented using current technology at a reasonable cost. It is a "proof of existence" statement. However, make it clear that the sample implementation could be done differently in the final system. The sample implementation is merely proposed as a possibility.

For example, when the Apollo program to put a man on the moon in the 1960s was first proposed, the plan was to place a rocket in earth orbit, then launch a landing vehicle directly to the moon's surface. In the final successful program, the rocket was launched directly into a lunar orbit, from which the lander was launched to the moon's surface. It was not a bad thing to make the first proposal, however, as this gave confidence that there was a feasible approach.

As Figure 11.2 shows, the problem statement should state what is to be done and not how it is to be implemented. It should be a statement of needs, not a proposal for a system architecture. The requestor should avoid describing system internals, as this restricts development flexibility. Performance specifications and protocols for interaction with external systems are legitimate requirements. Software engineering standards, such as modular construction, design for testability, and provision for future extensions, are also proper.

Requirements Statement	Design	Implementation
■ Problem scope	■ General approach	■ Platforms
■ What is needed	■ Algorithms	■ Hardware specs
■ Application context	■ Data structures	■ Software libraries
■ Assumptions	■ Architecture	■ Interface standards
■ Performance needs	■ Optimizations	
	■ Capacity planning	

Figure 11.2 Kinds of requirements. Do not make early design and implementation decisions or you will compromise development.

A problem statement may have more or less detail. A requirement for a conventional product, such as a payroll program or a billing system, may have considerable detail. A requirement for a research effort in a new area may lack details, but presumably the research has some objective that should be clearly stated.

Most problem statements are ambiguous, incomplete, or even inconsistent. Some requirements are just plain wrong. Some requirements, although precisely stated, have unpleasant consequences on the system behavior or impose unreasonable implementation costs. Some requirements do not work out as well as the requestor thought. The problem statement is just a starting point for understanding the problem, not an immutable document. The purpose of the subsequent analysis (next chapter) is to fully understand the problem and its implications. There is no reason to expect that a problem statement prepared without a full analysis will be correct.

11.3.1 The ATM Case Study

Figure 11.3 shows a problem statement for an automated teller machine (ATM) network.

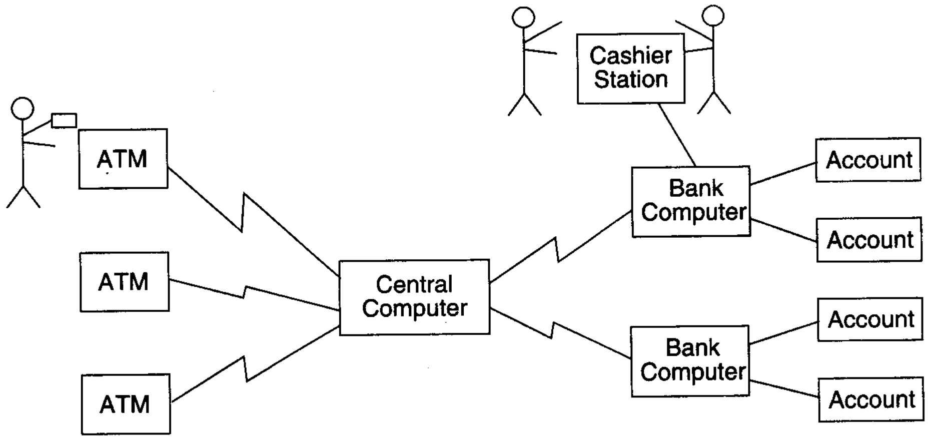

Figure 11.3 ATM network. The ATM case study threads throughout the remainder of this book.

Design the software to support a computerized banking network including both human cashiers and automatic teller machines (ATMs) to be shared by a consortium of banks. Each bank provides its own computer to maintain its own accounts and process transactions against them. Cashier stations are owned by individual banks and communicate directly with their own bank's computers. Human cashiers enter account and transaction data.

Automatic teller machines communicate with a central computer that clears transactions with the appropriate banks. An automatic teller machine accepts a cash card, interacts with the user, communicates with the central system to carry out the transaction, dispenses cash, and prints receipts. The system requires appropriate recordkeeping and security provisions. The system must handle concurrent accesses to the same account correctly.

The banks will provide their own software for their own computers; you are to design the software for the ATMs and the network. The cost of the shared system will be apportioned to the banks according to the number of customers with cash cards.

11.4 Chapter Summary

The first stage of a project is to devise a new idea. The idea can involve a new system or an improvement to an existing system. Before investing time and money into development, it is necessary to evaluate the feasibility of the system, the difficulty and risk of developing it, the demand for the system, and the cost-benefit ratio. This process should consider the viewpoints of all the stakeholders of the system and should make the trade-offs necessary to provide a good chance of success, not just technical success, but also business success. This process usually results in some adjustments to the original idea. When the system conception stage is complete, write a problem statement that serves as the starting point for analysis. The problem statement need not be complete, and it will change during development, but the writing of the statement helps to focus the attention of the project.

business case	problem statement
cost-benefit trade-off	requirement
design decision	stakeholder
implementation constraint	system conception

Figure 11.4 Key concepts for Chapter 11

Exercises

11.1 (3) Consider a new antilock braking system for crash avoidance in an automobile. Elaborate the following high-level questions and explain your answers.

a. Who is the application for? Who are the stakeholders? Estimate how many persons in your country are potential customers.
b. Identify three features that should be included and three features that should be omitted.
c. Identify three systems with which it must work.
d. What are two of the largest risks?

11.2 (3) Repeat Exercise 11.1 for software that supports Internet selling of books.

11.3 (3) Repeat Exercise 11.1 for software that supports the remodeling of kitchens.

11.4 (3) Repeat Exercise 11.1 for an online auction system.

11.5 (4) Prepare a problem statement, similar to that for the ATM system, for each of the following systems. You may limit the scope of the system, but be precise and avoid making implementation decisions. Use 75–150 words per specification.

a. bridge player
b. change-making machine
c. car cruise control
d. electronic typewriter
e. spelling checker
f. telephone answering machine

11.6 (3) Rephrase the following requirements to make them more precise. Remove any design decisions posing as requirements:

a. A system to transfer data from one computer to another over a telecommunication line. The system should transmit data reliably over noisy channels. Data must not be lost if the receiving end cannot keep up or if the line drops out. Data should be transmitted in packets, using a master–slave protocol in which the receiving end acknowledges or negatively acknowledges all exchanges.

b. A system for automating the production of complex machined parts. The parts will be designed using a three–dimensional drafting editor that is part of the system. The system will produce tapes that can be used by numerical control (N/C) machines to actually produce the parts.

c. A desktop publishing system, based on a what-you-see-is-what-you-get philosophy. The system will support text and graphics. Graphics include lines, squares, rectangles, polygons, circles, and ellipses. Internally, a circle is represented as a special case of an ellipse and a square as a special case of a rectangle. The system should support interactive, graphical editing of documents.

d. A system for generating nonsense. The input is a sample document. The output is random text that mimics the input text by imitating the frequencies of combinations of letters of the input. The user specifies the order of the imitation and the length of the desired output. For order *N*, every output sequence of *N* characters is found in the input and at approximately the same frequency. As the order increases, the style of the output more closely matches the input.

The system should generate its output with the following method: Select a position at random in the document being imitated. Scan forward in the input text until a sequence of characters is found that exactly matches the last $N - 1$ characters of the output. If you reach the end of the input, continue scanning from the beginning. When a match is found, copy the letter that follows the matched sequence from the input to the output. Repeat until the desired amount of text is generated.

e. A system for distributing electronic mail over a network. Each user of the system should be able to send mail from any computer account and receive mail on one designated account. There should be provisions for answering or forwarding mail, as well as saving messages in files or printing them. Also, users should be able to send messages to several other users at once through distribution lists. Each computer on the net should hold any messages destined for computers that are down.