# FEASIBILITY STUDY

Depending on the results of the initial investigation, the survey is expanded to a more detailed feasibility study. An important outcome of the preliminary investigation is the determination that the system requested is feasible. A feasibility study is a test of a system proposal according to its workability, impact on the organization, ability to meet user needs, and effective use of resources.

It focuses on three major questions:

1. What are the user’s demonstrable needs and how does a candidate system meet them?
2. What resources are available for given candidate systems? Is the problem worth solving?
3. What are the likely impacts of the candidate system on the organization? How well does it fit within the organization’s master MIS plan?

Each of these questions must be answered carefully. They revolve around investigation and evaluation of the problem, identification and description of candidate systems, specification of performance and the cost of each system and final selection of the best system.

The objective of a feasibility study is not to solve the problem but to acquire a sense of its scope. During the study, the problem definition is crystallized and aspects of the problem to be included in the system are determined. Consequently, costs and benefits are estimated with greater accuracy at this stage.

There are three aspects in the feasibility study portion of the preliminary investigation:

**TECHNICAL FEASIBILITY**

Can the work for the project be done with current equipment, existing software technology and available personnel? If new technology is required, what is the likelihood that it can be developed? The technical issues usually raised during the feasibility stage of the investigation include these:

1. Does the necessary technology exist to do what is suggested?
2. Do the proposed equipments have the technical capacity to hold the data required to use the new system?
3. Will the proposed system provide adequate responses to inquiries regardless of the number or location of users?
4. Can the system be expanded if developed?
5. Are there technical guarantees of accuracy, reliability, ease of access and data security?

**ECONOMIC FEASIBILITY**

A system that can be developed technically and that will be used if installed must still be a good investment for the organization. Financial benefits must equal or exceed the costs.

The financial and economic question raised by analysts during the preliminary investigation is for the purpose of estimating the following:

1. The cost to conduct a full system investigation
2. The cost of hardware and software for the class of application being considered
3. The benefits in the form of reduced costs or fewer costly errors
4. The cost if nothing changes (i.e., the proposed system is not developed).
5. Are there sufficient benefits in creating the system to make the costs acceptable? Or, are the costs of not creating the system so great that the project must be undertaken?

**OPERATIONAL FEASIBILITY**

Proposed projects are beneficial only if they can be turned into information systems that will meet the organization’s operating requirements. Simply stated, this test of feasibility asks, will the system be used if it is developed and implemented? Will there be resistance from users that will undermine the possible application benefits?

Here are the questions that will help to test the operational feasibility of a project:

1. Is there sufficient support for the project from management? From users? If current system is well liked and used to the extent that persons will not be able to see reasons for a change, there may be resistance.
2. Are current business methods acceptable to the users?
3. Have the users been involved in planning and development of the project?
4. Will the proposed system cause harm? Will it produce poorer results in any respect or area? Will individual performance be poorer after implementation that before?
5. All operational aspects must be considered carefully?

**2.4 MySQL**

The most popular Open Source SQL database management system, is developed, distributed, and supported by Oracle Corporation. A database is a structured collection of data. It may be anything from a simple shopping list to a picture gallery or the vast amounts of information in a corporate network. To add, access, and process data stored in a computer database, you need a database management system such as MySQL Server. Since computers are very good at handling large amounts of data, database management systems play a central role in computing, as standalone utilities, or as parts of other applications. The SQL part of “MySQL” stands for “Structured Query Language”. SQL is the most common standardized language used to access databases. Depending on your programming environment, you might enter SQL directly (for example, to generate reports), embed SQL statements into code written in another language, or use a language-specific API that hides the SQL syntax. SQL is defined by the ANSI/ISO SQL Standard. The SQL standard has been evolving since 1986 and several versions exist. In this manual, “SQL-92” refers to the standard released in 1992, “SQL:1999” refers to the standard released in 1999, and “SQL:2003” refers to the current version of the standard. We use the phrase “the SQL standard” to mean the current version of the SQL Standard at any time. **MySQL software is Open Source.**Open Source means that it is possible for anyone to use and modify the software. Anybody can download the MySQL software from the Internet and use it without paying anything. If you wish, you may study the source code and change it to suit your needs. The MySQL software uses the GPL. **The MySQL Database Server is very fast, reliable, scalable, and easy to use.** If that is what you are looking for, you should give it a try. MySQL Server can run comfortably on a desktop or laptop, alongside your other applications, web servers, and so on, requiring little or no attention. If you dedicate an entire machine to MySQL, you can adjust the settings to take advantage of all the memory, CPU power, and I/O capacity available. MySQL can also scale up to clusters of machines, networked together.MySQL Server was originally developed to handle large databases much faster than existing solutions and has been successfully used in highly demanding production environments for several years. Although under constant development, MySQL Server today offers a rich and useful set of functions. Its connectivity, speed, and security make MySQL Server highly suited for accessing databases on the Internet. **MySQL Server works in client/server or embedded systems.** The MySQL Database Software is a client/server system that consists of a multi-threaded SQL server that supports different backends, several different client programs and libraries, administrative tools, and a wide range of application programming interfaces (APIs).

MySQL Server has a practical set of features developed in close cooperation with our users. It is very likely that your favorite application or language supports the MySQL Database Server.

**JAVA**:

Java becomes a special and different due to the following reason

* Secure:

As you are likely aware, every time that you download a “normal” program, you are risking a viral infection. Prior to Java, most users did not download executable programs frequently, and those who did scan them for viruses prior to execution. Even so, most users still worried about the possibility of infecting their systems with a virus. In addition to viruses, another type of malicious program exists that must be guarded against. This type of program can gather private information, such as credit card numbers, bank account balances, and passwords, by searching the contents of your computer’s local file system. Java answers both of these concerns by providing a “firewall” between a networked application and your computer. When you use a Java-compatible Web browser, you can safely download Java applets without fear of viral infection or malicious intent. Java achieves this protection by confining a Java program to the Java execution environment and not allowing it access to other parts of the computer.

* Portable:

Many types of computers and operating systems are in use throughout the world and many are connected to the Internet. For programs to be dynamically downloaded to all the various types of platforms connected to the Internet, some means of generating portable executable code is needed. As you will soon see, the same mechanism that helps ensure security also helps create portability. Indeed, Java’s solution to these two problems is both elegant and efficient.

* Object Oriented:

Although influenced by its predecessors, Java was not designed to be source-code compatible with any other language. This allowed the Java team the freedom to design with a blank slate. One outcome of this was a clean, usable, pragmatic approach to objects. Borrowing liberally from many seminal object-software environments of the last few decades, Java manages to strike a balance between the purist’s “everything is an object” paradigm and the pragmatist’s “stay out of my way” model. The object model in Java is simple and easy to extend, while simple types, such as integers, are kept as high-performance no objects.

* Robust:

The multiplatform environment of the Web places extraordinary demands on a program, because the program must execute reliably in a variety of systems. Thus, the ability to create robust programs was given a high priority in the design of Java. To gain reliability, Java restricts you in a few key areas, to force you to find your mistakes early in program development. At the same time, Java frees you from having to worry about many of the most common causes of programming errors. Because Java is a strictly typed language, it checks your code at compile time. However, it also checks your code at run time. In fact, many hard-to-track-down bugs that often turn up in hard-to-reproduce run-time situations are simply impossible to create in Java. Knowing that what you have written will behave in a predictable way under diverse conditions is a key feature of Java. To better understand how Java is robust, consider two of the main reasons for program failure: memory management mistakes and mishandled exceptional conditions (that is, run-time errors). Memory management can be a difficult, tedious task in traditional programming environments. For example, in C/C++, the programmer must manually allocate and free all dynamic memory. This sometimes leads to problems, because programmers will either forget to free memory that has been previously allocated or, worse, try to free some memory that another part of their code is still using. Java virtually eliminates these problems by managing memory allocation and deallocation for you. (In fact, deallocation is completely automatic, because Java provides garbage collection for unused objects.) Exceptional conditions in traditional environments often arise in situations such as division by zero or “file not found,” and they must be managed with clumsy and hard-to-read constructs. Java helps in this area by providing object-oriented exception handling. In a well-written Java program, all run-time errors can—and should—be managed by your program

* Multithreaded:

Java was designed to meet the real-world requirement of creating interactive, networked programs. To accomplish this, Java supports multithreaded programming, which allows you to write programs that do many things simultaneously. The Java run-time system comes with an elegant yet sophisticated solution for multiprocess synchronization that enables you to construct smoothly running interactive systems. Java’s easy-to-use approach to multithreading allows you to think about the specific behavior of your program, not the multitasking subsystem.

* Architectural-Neutral:

A central issue for the Java designers was that of code longevity and portability. One of the main problems facing programmers is that no guarantee exists that if you write a program today, it will run tomorrow—even on the same machine. Operating system upgrades, processor upgrades, and changes in core system resources can all combine to make a program malfunction. The Java designers made several hard decisions in the Java language and the Java Virtual Machine in an attempt to alter this situation. Their goal was “write once; run anywhere, anytime, forever.” To a great extent, this goal was accomplished.

* Interpreted and High performance:

As described earlier, Java enables the creation of cross-platform programs by compiling into an intermediate representation called Java byte code. This code can be interpreted on any system that provides a Java Virtual Machine. Most previous attempts at cross platform solutions have done so at the expense of performance. Other interpreted systems, such as BASIC, Tcl, and PERL, suffer from almost insurmountable performance deficits. Java, however, was designed to perform well on very low-power CPUs. As explained earlier, while it is true that Java was engineered for interpretation, the Java byte code was carefully designed so that it would be easy to translate directly into native machine code for very high performance by using a just-in-time compiler. Java run-time systems that provide this feature lose none of the benefits of the platform-independent code. “High-performance cross-platform” is no longer an oxymoron.

* Distributed:

Java is designed for the distributed environment of the Internet, because it handles TCP/IP protocols. In fact, accessing a resource using a URL is not much different from accessing a file. The original version of Java (Oak) included features for intra address- space messaging. This allowed objects on two different computers to execute procedures remotely. Java revived these interfaces in a package called Remote Method Invocation (RMI). This feature brings an unparalleled level of abstraction to client/server programming.

* Dynamic:

Java programs carry with them substantial amounts of run-time type information that is used to verify and resolve accesses to objects at run time. This makes it possible to dynamically link code in a safe and expedient manner. This is crucial to the robustness of the applet environment, in which small fragments of bytecode may be dynamically updated on a running system.

**Swing** :

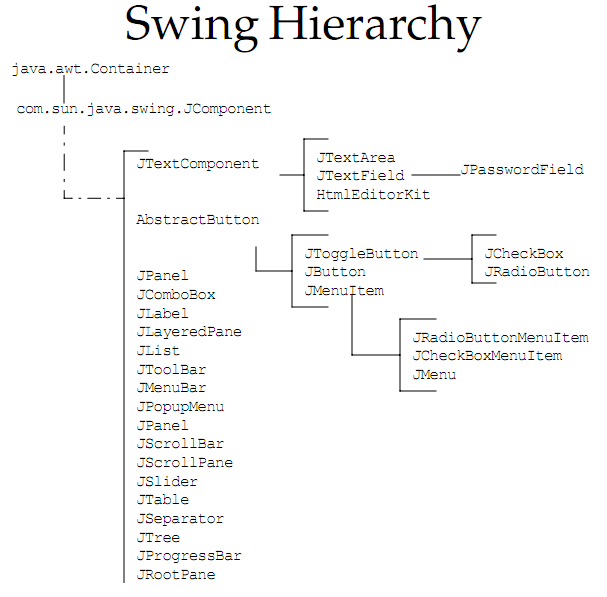
• **Pluggable look and feel:**

• Application appears to be platform specific

• There are custom Swing components

• **Swing architecture:**

• Built around APIs that implement various parts of

 AWT

• Most components do not use platform-specific

implementations like AWT

**JDBC™ API**

The JDBC API is the API for connectivity with relational database systems. The JDBC API has two parts: an application-level interface used by the application components to access a database, and a service provider interface to attach a JDBC driver to the J2EE platform. Support for the service provider interface is not required in J2EE products.

**2.5.5 JDK**

Tomcat will operate under any Java Development Kit (JDK) environment that provides a JDK 1.2 (also known as Java2 Standard Edition, or J2SE) or later platform. You will need a Java Development Kit, as opposed to a Java Runtime Environment, so that your servlets, other classes, and JSP pages can be compiled. Tomcat 4 has been extensively tested with JDK 1.3.1, which is recommended.

**Java Applications**

Every Java application contains a class that defines a method called main(). The name of this class is the name that you use as the argument to the Java interpreter when you run the application. You can call the class whatever you want, but the method which is executed first in an application is always called main(). When you run your Java application, the method main()will typically cause methods belong- ing to other classes to be executed, but the simplest possible Java application program consists of one class containing just the method main(). As you will see below, the main()method has a particular fixed form, and if it is not of the required form, it will not be recognized by the Java interpreter as the method where execution starts.

You can see how this works by taking a look at just such a Java program. You need to enter the program code using your favorite plaintext editor, or if you have a Java development system with an editor, you can enter the code for the example using that. When you have entered the code, save the file with the same name as that used for the class and with the extension .java. For this example the file name will be OurFirstProgram.java. The code for the program is shown in Figure 1-9 The program consists of a definition for a class I have called OurFirstProgram. The class definition contains only one method, the method main(). The first line of the definition for the method main()is always of the form:

public static void main(String[] args)