**SOFTWARE ENVIRONMENT**

**INTRODUCTION TO FROND END**

The software requirement specification is produced at the culmination of the analysis task. The function and performance allocated to software as part of system engineering are refined by establishing a complete information description as functional representation, a representation of system behavior, an indication of performance requirements and design constraints, appropriate validation criteria.

**USER INTERFACE**

\* Swing - Swing is a set of classes that provides more powerful and flexible components that are possible with AWT. In addition to the familiar components, such as button checkboxes and labels, swing supplies several exciting additions, including tabbed panes, scroll panes, trees and tables.

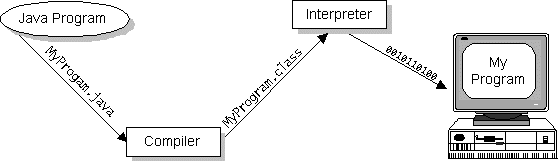
\* Applet - Applet is a dynamic and interactive program that can run inside a web page displayed by a java capable browser such as hot java or Netscape.

**JAVA:**

Java was conceived by James Gosling, Patrick Naughton , Chris Wrath, Ed Frank, and Mike Sheridan at Sun Micro system. It is an platform independent programming language that extends it’s features wide over the network.

* It’s a light weight package, as they are not implemented by platform-specific code.
* Related classes are contained in javax.swing and its sub packages, such as javax.swing.tree.
* Components explained in the Swing have more capabilities than those of AWT.

Java is also unusual in that each Java program is both compiled and interpreted. With a compiler, you translate a Java program into an intermediate language called Java byte codes--the platform-independent codes interpreted by the Java interpreter. With an interpreter, each Java byte code instruction is parsed and run on the computer. Compilation happens just once; interpretation occurs each time the program is executed. This figure illustrates how this works.



Java byte codes can be considered as the machine code instructions for the Java Virtual Machine (Java VM). Every Java interpreter, whether it's a Java development tool or a Web browser that can run Java applets, is an implementation of the Java VM. The Java VM can also be implemented in hardware.

Java byte codes help make "write once, run anywhere" possible. The Java program can be compiled into byte codes on any platform that has a Java compiler

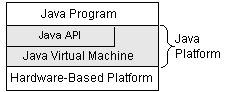
The byte codes can then be run on any implementation of the Java VM. For example, the same Java program can run on Windows NT, Solaris, and Macintosh.

### THE JAVA PLATFORM

A platform is the hardware or software environment in which a program runs. The Java platform differs from most other platforms in that it's a software-only platform that runs on top of other, hardware-based platforms. Most other platforms are described as a combination of hardware and operating system.

The Java API is a large collection of ready-made software components that provide many useful capabilities, such as graphical user interface (GUI) widgets. The Java API is grouped into libraries (packages) of related components.

The following figure depicts a Java program, such as an application or applet, that's running on the Java platform. As the figure shows, the Java API and Virtual Machine insulates the Java program from hardware dependencies.



As a platform-independent environment, Java can be a bit slower than native code. However, smart compilers, well-tuned interpreters, and just-in-time byte code compilers can bring Java's performance close to that of native code without threatening portability.

With packages of software components that provide a wide range of functionality. The core API is the API included in every full implementation of the Java platform. The core API gives you the following features:

* **The Essentials**: Objects, strings, threads, numbers, input and output, data structures, system properties, date and time, and so on.
* **Applets**: The set of conventions used by Java applets.
* **Networking**: URLs, TCP and UDP sockets, and IP addresses.
* **Internationalization**: Help for writing programs that can be localized for users worldwide. Programs can automatically adapt to specific locales and be displayed in the appropriate language.
* **Security**: Both low-level and high-level, including electronic signatures, public/private key management, access control, and certificates.
* **Software components**: Known as JavaBeans, can plug into existing component architectures such as Microsoft's OLE/COM/Active-X architecture, OpenDoc, and Netscape's Live Connect.
* **Object serialization**: Allows lightweight persistence and communication via Remote Method Invocation (RMI).
* **Java Database Connectivity (JDBC)**: Provides uniform access to a wide range of relational databases.
* Java not only has a core API, but also standard extensions. The standard extensions define APIs for 3D, servers, collaboration, telephony, speech, animation, and more.

**NETWORKING BASICS**

Ken Thompson and Dennis Ritchie developed UNIX in concert with the C language at Bell Telephone Laboratories, Murray Hill, New Jersey, in 1969. In 1978, Bill Joy was leading a project at Cal Berkeley to add many new features to UNIX, such as virtual memory and full-screen display capabilities.

By early 1984, just as Bill was leaving to found Sun Microsystems, he shipped 4.2BSD, commonly known as Berkeley UNIX.4.2BSD came with a fast file system, reliable signals, inter process communication, and, most important, networking.

**SOCKET OVERVIEW**

A network socket is a lot like an electrical socket. Various plugs around the network have a standard way of delivering their payload. Anything that understands the standard protocol can “plug in” to the socket and communicate.

Internet protocol (IP) is a low-level routing protocol that breaks data into small packets and sends them to an address across a network, which does not guarantee to deliver said packets to the destination.

Transmission Control Protocol (TCP) is a higher-level protocol that manages to reliably transmit data. A third protocol, User Datagram Protocol (UDP), sits next to TCP and can be used directly to support fast, connectionless, unreliable transport of packets.

**CLIENT/SERVER**

A server is anything that has some resource that can be shared. There are compute servers, which provide computing power; print servers, which manage a collection of printers; disk servers, which provide networked disk space; and web servers, which store web pages. A client is simply any other entity that wants to gain access to a particular server.

In Berkeley sockets, the notion of a socket allows as single computer to serve many different clients at once, as well as serving many different types of information. This feat is managed by the introduction of a port, which is a numbered socket on a particular machine.

A server process is said to “listen” to a port until a client connects to it. A server is allowed to accept multiple clients connected to the same port number, although each session is unique. To mange multiple client connections, a server process must be multithreaded or have some other means of multiplexing the simultaneous I/O.

**RESERVED SOCKETS**

Once connected, a higher-level protocol ensues, which is dependent on which port you are using. TCP/IP reserves the lower, 1,024 ports for specific protocols. Port number 21 is for FTP, 23 is for Telnet, 25 is for e-mail, 79 is for finger, 80 is for HTTP, 119 is for Netnews-and the list goes on. It is up to each protocol to determine how a client should interact with the port.

**JAVA AND THE NET**

Java supports TCP/IP both by extending the already established stream I/O interface. Java supports both the TCP and UDP protocol families. TCP is used for reliable stream-based I/O across the network. UDP supports a simpler, hence faster, point-to-point datagram-oriented model.

**INET ADDRESS**

The InetAddress class is used to encapsulate both the numerical IP address and the domain name for that address. We interact with this class by using the name of an IP host, which is more convenient and understandable than its IP address. The InetAddress class hides the number inside. As of Java 2, version 1.4, InetAddress can handle both IPv4 and IPv6 addresses.

Factory Methods

The InetAddress class has no visible constructors. To create an InetAddress object, we use one of the available factory methods. Factory methods are merely a convention whereby static methods in a class return an instance of that class. This is done in lieu of overloading a constructor with various parameter lists when having unique method names makes the results much clearer.

Three commonly used InetAddress factory methods are shown here.

static InetAddress getLocalHost( ) throws UnknownHostException

static InetAddress getByName(String hostName) throws UnknowsHostException

static InetAddress[ ] getAllByName(String hostName)

throws UnknownHostException

The getLocalHost( ) method simply returns the InetAddress object that represents the local host. The getByName( ) method returns an InetAddress for a host name passed to it. If these methods are unable to resolve the host name, they throw an UnknownHostException.

On the internet, it is common for a single name to be used to represent several machines. In the world of web servers, this is one way to provide some degree of scaling. The getAllByName ( ) factory method returns an array of InetAddresses that represent all of the addresses that a particular name resolves to. It will also throw an UnknownHostException if it can’t resolve the name to at least one address. Java 2, version 1.4 also includes the factory method getByAddress( ), which takes an IP address and returns an InetAddress object. Either an IPv4 or an IPv6 address can be used.

**INSTANCE METHODS**

The InetAddress class also has several other methods, which can be used on the objects returned by the methods just discussed. Here are some of the most commonly used.

Boolean equals (Object other)- Returns true if this object has the same Internet address as other.

byte[ ] getAddress( )- Returns a byte array that represents the object’s Internet address in network byte order.

String getHostAddress( )- Returns a string that represents the host address associated with the InetAddress object.

String getHostName ( ) - Returns a string that represents the host name associated with the InetAddress object.

boolean isMulticastAddress( )- Returns true if this Internet address is a multicast address. Otherwise, it returns false.

String toString( )- Returns a string that lists the host name and the IP address for conveneince.

Internet addresses are looked up in a series of hierarchically cached servers. That means that your local computer might know a particular name-to-IP-address mapping autocratically, such as for itself and nearby servers. For other names, it may ask a local DNS server for IP address information. If that server doesn’t have a particular address, it can go to a remote site and ask for it. This can continue all the way up to the root server, called InterNIC (internic.net).

**TCP/IP CLIENT SOCKETS**

TCP/IP sockets are used to implement reliable, bidirectional, persistent, point-to-point, stream-based connections between hosts on the Internet. A socket can be used to connect Java’s I/O system to other programs that may reside either on the local machine or on any other machine on the Internet.

There are two kinds of TCP sockets in Java. One is for servers, and the other is for clients. The ServerSocket class is designed to be a “listener,” which waits for clients to connect before doing anything. The Socket class is designed to connect to server sockets and initiate protocol exchanges.

The creation of a Socket object implicitly establishes a connection between the client and server. There are no methods or constructors that explicitly expose the details of establishing that connection. Here are two constructors used to create client sockets:

Socket(String hostName, int port) Creates a socket connecting the local host to the named host and port; can throw an UnknownHostException or anIOException.

Socket(InetAddress ipAddress, int port) Creates a socket using a preexisting InetAddress object and a port; can throw an IOException.

A socket can be examined at any time for the address and port information associated with it, by use of the following methods:

InetAddress getInetAddress ( ) - Returns the InetAddress associated with the Socket object.

Int getPort ( ) Returns the remote port to which this Socket object is connected.

int getLocalPort( ) Returns the local port to which this Socket object is connected.

Once the Socket object has been created, it can also be examined to gain access to the input and output streams associated with it. Each of these methods can throw an IOException if the sockets have been invalidated by a loss of connection on the Net.

InputStream getInputStream ( )Returns the InputStream associated with the invoking socket.

OutputStream getOutputStream ( ) Returns the OutputStream associated with the invoking socket.

**TCP/IP SERVER SOCKETS**

Java has a different socket class that must be used for creating server applications. The ServerSocket class is used to create servers that listen for either local or remote client programs to connect to them on published ports. ServerSockets are quite different form normal Sockets.

When we create a ServerSocket, it will register itself with the system as having an interest in client connections. The constructors for ServerSocket reflect the port number that we wish to accept connection on and, optionally, how long we want the queue for said port to be. The queue length tells the system how many client connection it can leave pending before it should simply refuse connections. The default is 50. The constructors might throw an IOException under adverse conditions. Here are the constructors:

ServerSocket(int port) Creates server socket on the specified port with a queue length of 50.

Serversocket(int port, int maxQueue)-Creates a server socket on the specified port with a maximum queue length of maxQueue.

ServerSocket(int port, int maxQueue, InetAddress localAddress)-Creates a server socket on the specified port with a maximum queue length of maxQueue. On a multihomed host, localAddress specifies the IP address to which this socket binds.

ServerSocket has a method called accept( ), which is a blocking call that will wait for a client to initiate communications, and then return with a normal Socket that is then used for communication with the client.

**2.2 INTRODUCTION TO BACK END**

**MICROSOFT SQL SERVER 2005**

Micro soft SQL Server 2005 provides a scalable database that combines ease of use with complex analysis and data warehousing tools. SQL Server includes a rich graphical user interface (UI) along with a complete development environment for creating data- driven applications.

Commerce server takes advantages of SQL Server data warehousing and analysis capabilities in several key areas. The commerce server data warehouse, for example, uses SQL Server Data Transformation Services (DTS) to transform data stored in SQL Server database to the format used by Commerce server resources.

**WINDOWS 2005 AND SQL SERVER SECURITY:**

Existing Microsoft ® 2005 accounts (Active Directory users of groups) must be granted permissions to connect to Microsoft ® Microsoft ® SQL Server ™ before they can access a database. If all members of a windows group require connections to SQL Server, you can grant login permissions for the group as a whole.

Managing group permissions is easier than managing permissions for individual users. If you do not want a group to be granted permissions collectively, you can grant permissions to connect to SQL Server for each individual user.

**ACTIVE DIRECTORY USERS AND GROUPS:**

In windows 2005, users are individuals who have an account that provides specific privileges to access information and resources. Granting permission to users to develop , manage , and use workflow applications is dependent upon the integration of windows 2005 domain accounts and SQL Server roles. If a number of users all have the same permissions, they can be treated as a single unit, called a group, which can be assigned permissions that apply to all members of the group. Individuals can be added to or removed from groups as desired.

There are two types of windows groups: **Global** and **Local**

Global groups contain user accounts from the windows 2005 server domain in which they are created on a computer running windows 2005 professional.

Local groups can contain user accounts and global from the domain in which they are created and any trusted domain. Local groups cannot contain other local groups. In addition, Windows 2005 has predefined, built – in local groups, such as administrators, Users and Guests. By default, these built – in groups always are available on any windows 2005 computers, unless they are removed explicitly.

To grant access to SQL Server to a windows local or global group, specify the domain or computer name on which the group is defined, flowered by a backslash, and then the group name. For example, to grant access to the windows 2005 group SQL\_users in the

Windows 2005 domain LONDON, specify LONDON/SQL\_users as the group name.

However, to grant access to a window built – in local group, specify BUILT IN, instead of the domain or computer name. To grant access to the built – in windows local group Administrators, specify BUILTIN/Administrator as the group name to add to SQL server.

You must have appropriate permissions on the server to create windows groups or users or to create SQL servers users or roles.For additional information about windows accounts, see your windows documentation.

**SQL SERVER LOGINS**

SQL Server logins are the account identifiers that control access to any SQL server will not complete connections unless it has first verified that the login you specified is valid. This verification of the login is called authentication.

A member of the SQL server sysadmin fixed- server role first must specify to SQL Server all the windows accounts or groups that can connect to SQL Server. Your access to SQL Server is controlled by your windows account or group, which is authenticated when you log on to the windows operating system on the client.

When Connecting, the SQL Server client software requests a windows trusted connection to SQL Server. Windows will not open a trusted connection unless the client has logged on successfully using a valid windows account. The properties of a trusted connection include the windows group and user accounts of the client that opened the connection. SQL Server gets the user account information from the trusted connection properties and matches them against the windows accounts defined as valid SQL Server logins. If SQL Server finds a match, it accepts the connection. You are identified in SQL Server finds a match, it accepts the connection. You are identified in SQL Server by your windows group or user account.

**DATABASE ROLES**

Using database roles, you can collect users into a single unit to which you can apply permissions. Permissions granted to, denied to or revoked from a role also apply to any members of the role.

SQL Server roles exist within a database and cannot span more than one database. Because roles are unique to each database, you can reuse a role name, such as “reviewer” in each database that you create. To assign users and groups to data base roles, the users and groups must have valid windows domain accounts and SQL Server logins.

If you make any changes to the membership of database roles in your workflow application, you must synchronize the user directory for role permissions to work properly.

* Users can belong to more than one database role at a time.
* Roles can contain windows accounts and other SQL Server users and roles.
* A scalable model is provided for setting up the right level of security within a database

It is easy to manage permissions in a database if you define a set of roles based on job functions and assign each role the permissions that apply to that job. Then, you can move users between roles rather than having the permissions for each individual user.

The owner of a role determines who can be added or removed from the role. The owner is either the user explicitly specified as the owner when the role is created or the user who created the role when no owner is specified. If you make any changes to the membership of database roles in your workflow application, you must synchronize the user directory for role permissions to work properly. Database roles are created for a particular database. In SQL Server 7.0 and SQL Server 2005, users can belong to multiple roles. Because users can belong to more than one database role at a time, it is no longer required for users to assume temporarily the identify (and permissions) of other users through aliases.

Note if you plan to make a template based on a workflow application, you should use role based permissions for everything, because the set of database users will be different for each instance of a project based on the template.

**DATABASE USER ACCOUNT**

While a SQL Server login makes it possible for a user to access SQL Server, a database user account is required for the user to access a specific database. Then, these user accounts can be associated with the roles defined in you workflow application.

A user account can be a member of any number of roles within the same workflow application. For example, a user can be a member of the admin role and the author’s role for the same database, with each role granting different permissions.

The effective permissions on an object granted to a member of more than one role are the cumulative permissions of the roles, although denied permissions in one role has precedence over the same permissions granted in another role. For example, the admin role might grant access to a table while the author’s role denies access to the same table. A member of both roles is denied access to the table, because denied access is the most restrictive.