Universidad Complutense

Facultad de Informática

eGorilla

Glosario de Términos

Asignatura: *Ingeniería del Software*

Curso Académico: *2008/2009*

Grupo: *4º B*

Índice

1. Introducción

En este documento se le presenta al lector el vocabulario empleado en la documentación de la aplicación, **definiendo** conceptos utilizados así como **acrónimos** y **sinónimos** empleados.

Dado su gran crecimiento a medida que se va profundizando en el desarrollo de la aplicación, se ha optado por separar este apartado, que originalmente formaba parte de la especificación de requisitos, en un documento aparte para facilitar su ampliación y ante la conclusión de las revisiones de dicho documento de especificación de requisitos.

2. Vocabulario

1.1. Definiciones

1.1.1. Nodo

Componente activo de la red P2P desarrollada ya sea cliente, servidor o las dos cosas.

1.1.2. Paquete

Unidad mínima de transferencia de información en la red P2P.

1.1.3. Md5

Algoritmo de reducción criptográfico de 128 bits que dado un nombre asigna un identificador único al mismo garantizando (en un 99%) que no existan dos identificadores de archivo iguales en la red P2P.

1.1.4. IP, TCP, UDP

Protocolos asociados a **teoría de redes**. Son protocolos estándares que se emplean en las tecnologías de redes actuales.

1.1.5. Puerto

Un **puerto de red** es una interfaz para comunicarse con un programa a través de una red. Un puerto de red puede ser un puerto serie o un puerto paralelo; suelen ser numerados. La implementación del protocolo en el destino utilizará ese número para decidir a qué programa entregará los datos recibidos.

1.1.6. Dirección IP

Una **dirección IP** es un número que identifica de manera lógica y jerárquica a una interfaz de un dispositivo (habitualmente una computadora) dentro de una red que utilice el protocolo IP (*Internet Protocol*), que corresponde al nivel de red del protocolo TCP/IP.

1.1.7. Socket

Socket designa un concepto abstracto por el cual dos programas (posiblemente situados en computadoras distintas) pueden intercambiarse cualquier flujo de datos, generalmente de manera fiable y ordenada.

Un socket queda definido por una **dirección IP**, un **protocolo** y un número de **puerto**.

1.1.8. Chunk

Parte ó fragmento de un archivo ó fichero.

1.1.9. Máquina Virtual de Java (JVM)

Una **Máquina virtual Java** (**Java Virtual Machine**) es un programa nativo, es decir, ejecutable en una plataforma específica, capaz de interpretar y ejecutar instrucciones expresadas en un código binario especial (el **Java bytecode**), el cual es generado por el compilador del lenguaje Java.

El código binario de Java no es un lenguaje de alto nivel, sino un verdadero código máquina de **bajo nivel**, viable incluso como lenguaje de entrada para un microprocesador físico. Como todas las piezas del rompecabezas Java, fue desarrollado originalmente por **Sun Microsystems**.

La gran ventaja de la máquina virtual java es aportar portabilidad al lenguaje de manera que desde Sun Microsystems se han creado diferentes máquinas virtuales java para diferentes arquitecturas y así un programa .class escrito en un **Windows** puede ser interpretado en un entorno **Linux**. Tan solo es necesario disponer de dicha máquina virtual para dichos entornos.

1.1.10. JDK

Java Development Kit o (**JDK**), es un software que provee herramientas de desarrollo para la creación de programas en java. Puede instalarse en una computadora local o en una unidad de red.

En la unidad de red se puede tener la aplicación distribuida en varias computadoras y trabajar como una sola aplicación.

1.1.11. JRE

JRE es el acrónimo de **Java Runtime Environment** (entorno en tiempo de ejecución Java) y se corresponde con un conjunto de utilidades que permite la ejecución de programas java sobre todas las plataformas soportadas.

JVM (máquina virtual Java) es una instancia de JRE en tiempo de ejecución, este es el programa que interpreta el código Java y además por las librerías de clases estándar que implementan el API de Java. Ambas JVM y API deben ser consistentes entre sí, de ahí que sean distribuidas de modo conjunto.

Un usuario sólo necesita el JRE para ejecutar las aplicaciones desarrolladas en lenguaje Java, mientras que para desarrollar nuevas aplicaciones en dicho lenguaje es necesario un entorno de desarrollo, denominado JDK, que además del JRE (mínimo imprescindible) incluye, entre otros, un compilador para Java.

1.1.12. NetBeans

IDE desarrollado por Sun Microsystems en Junio de 2000 que permite que las aplicaciones sean desarrolladas a partir de un conjunto de componentes de software llamados **módulos**. Un módulo es un archivo Java que contiene clases de java escritas para interactuar con las APIs de NetBeans y un archivo especial (manifest file) que lo identifica como módulo. Las aplicaciones construidas a partir de módulos pueden ser extendidas agregándole nuevos módulos. Debido a que los módulos pueden ser desarrollados independientemente, las aplicaciones basadas en la plataforma NetBeans pueden ser extendidas fácilmente por otros desarrolladores de software.

1.1.13. Eclipse

IDE desarrollado originalmente por **IBM** aunque actualmente es desarrollado por la **Fundación Eclipse** que consiste en un entorno de desarrollo integrado de **código abierto** multiplataforma para desarrollar lo que el proyecto llama "Aplicaciones de Cliente Enriquecido", opuesto a las aplicaciones "Cliente-liviano" basadas en navegadores. Esta plataforma, típicamente ha sido usada para desarrollar entornos de desarrollo integrados (del inglés IDE), como el IDE de Java llamado **Java Development Toolkit** (JDT) y el compilador (ECJ) que se entrega como parte de Eclipse (y que son usados también para desarrollar el mismo Eclipse).

1.1.14. Jargs

Librería desarrollada por Steve Purcell empleada para el parseo o análisis de los argumentos introducidos en la línea de comandos por los desarrolladores sobre el lenguaje java.

1.1.15. Log4j

Es una biblioteca open source desarrollada en Java por la **Apache Software Foundation** que permite a los desarrolladores de software elegir la salida y el nivel de granularidad de los mensajes o "logs" (logging) a tiempo de ejecución y no a tiempo de compilación como es comúnmente realizado. La configuración de salida y granularidad de los mensajes es realizada a tiempo de ejecución mediante el uso de archivos de configuración externos.

Por defecto Log4J tiene **6 niveles de prioridad** para los mensajes (trace, debug, info, warn, error, fatal). Además existen otros dos niveles extras (all y off):

- **FATAL:** se utiliza para mensajes críticos del sistema, generalmente después de guardar el mensaje el programa abortará.
- **ERROR:** se utiliza en mensajes de error de la aplicación que se desea guardar, estos eventos afectan al programa pero lo dejan seguir funcionando, como por ejemplo que algún parámetro de configuración no es correcto y se carga el parámetro por defecto.
- **WARN:** se utiliza para mensajes de alerta sobre eventos que se desea mantener constancia, pero que no afectan al correcto funcionamiento del programa.
- **INFO:** se utiliza para mensajes similares al modo "verbose" en otras aplicaciones.
- **DEBUG:** se utiliza para escribir mensajes de depuración. Este nivel no debe estar activado cuando la aplicación se encuentre en producción.
- **TRACE:** se utiliza para mostrar mensajes con un mayor nivel de detalle que debug.
- **ALL:** este es el nivel de máximo detalle, habilita todos los logs (en general equivale a TRACE).

- **OFF:** este es el nivel de mínimo detalle, deshabilita todos los logs.

En Log4J los mensajes son enviados a una (o varias) salida de destino, lo que se denomina un **appender**.

Existen varios appenders disponibles y configurados, aunque también podemos crear y configurar nuestros propios appenders.

Típicamente la salida de los mensajes es redirigida a un **fichero de texto *.log*** (***FileAppender*, *RollingFileAppender***), a un **servidor remoto donde** almacenar registros (***SocketAppender***), a una **dirección de correo electrónico** (***SMTPAppender***), e incluso en una **base de datos** (***JDBCAppender***).

Casi nunca es utilizado en un entorno de producción la **salida a la consola** (***ConsoleAppender***) ya que perdería gran parte de la utilidad de Log4J.

1.1.16. JFreeChart

Se trata de una librería de clases java libre desarrollada y mantenida por **David Gilbert** perteneciente al **proyecto JFree** para la **generación de gráficos** en diferentes modelos (pie, lineal, barras, etc.) sobre 2D ó 3D.

1.1.17. FindBugs

Es una herramienta **Opensource** desarrollada en **Java** por la **Universidad de Maryland** que escanea el código fuente de una aplicación y muestra un árbol con los errores clasificados por categorías, mostrando para cada uno el código donde aparece y cómo debería mejorarse.

Gracias a esta herramienta se puede incrementar el rendimiento de una aplicación además de eliminar bugs potenciales.

1.1.18. Archivo

Es la unidad básica en la red P2P susceptible de ser compartida o de de ser descargada. Cada archivo será gestionado por el módulo GestorDeFicheros encargado del fragmentado, reensamblado y de la gestión de los diferentes tipos de archivo que se pueden compartir en la red P2P.

1.1.19. XML

Lenguaje de programación basado en etiquetas empleado para desarrollar diferentes aplicaciones web así como otros usos.

1.1.20. Properties

Fichero de configuración donde se especifican los valores o características de un determinado sistema. Su contenido está formado por cláusulas del estilo nombreCampo = valor.

1.2. Acrónimos

1.2.1. P2P

Acrónimo de Peer to Peer. Se refiere a una red que no tiene clientes ni servidores fijos, sino una serie de nodos que se comportan simultáneamente como clientes y como servidores respecto de los demás nodos de la red.

1.2.2. SVN

Acrónimo de Sistema de Control de Versiones. Es software libre bajo una licencia de tipo Apache/BSD y se le conoce también como svn por ser ese el nombre de la herramienta de línea de comandos. Los archivos versionados no tienen cada uno un número de revisión independiente. En cambio, todo el repositorio tiene un único número de versión que identifica un estado común de todos los archivos del repositorio en cierto punto del tiempo. Para la gestión del svn se emplearán las herramientas TortoiseSVN y RapidSVN.

1.2.3. DCU

Diagrama de casos de uso.

1.2.4. DSS

Diagrama de secuencia del sistema.

1.2.5. ECS

Elemento de Configuración del Software.

1.2.6. ERS

Especificación de Requisitos del Software.

1.2.7. CU

Caso (o Casos) de Uso. Hay dos variantes en el documento de **Casos_De_Uso.doc**:

- **CUS:** Casos de uso del servidor.

- **CUC:** Casos de uso del cliente.

1.2.8. GR

Gestión de Riesgos.

1.2.9. GCS

Gestión de Configuración del Software.

1.2.10. TECS

Tabla de Elementos de Configuración del Software para controlar el estado de la configuración.

1.2.11. DS

Documento de Diseño del Sistema.

1.2.12. IDE

Entorno de desarrollo integrado (**Integrated Development Enviroment**)

1.2.13. GUI

Graphic User Interface ó Interfaz Gráfica de Usuario.

1.2.14. IEEE

Instituto de Ingenieros Eléctricos y Electrónicos (**Institute of Electrical and Electronics Engineers**)

1.3. Sinónimos

1.3.1. Sistema = Aplicación = eGorilla.

1.3.2. Fichero = Archivo.

1.3.3. Cliente eGorilla = Cliente, Servidor eGorilla = Servidor.

1.3.4. Socket = Conexión.

1.3.5. Chunk = Fragmento = Trozo = Parte de archivo ó fichero