Capítulo 2.
Conceptos Generales

Antes de hablar sobre el diseño e implementación del sistema, es necesario dar a conocer las tecnologías usadas, describiendo de ellas lo más importante y aquello que motivó su uso.

2.1 Agentes

Las tareas específicas que un agente puede realizar en una biblioteca digital son, entre otras:

El manejo de las peticiones particulares de los usuarios.

La selección del conjunto apropiado de fuentes de información.

La ejecución de estrategias efectivas de búsqueda sobre la red.

El entendimiento de las relaciones entre colecciones de datos.

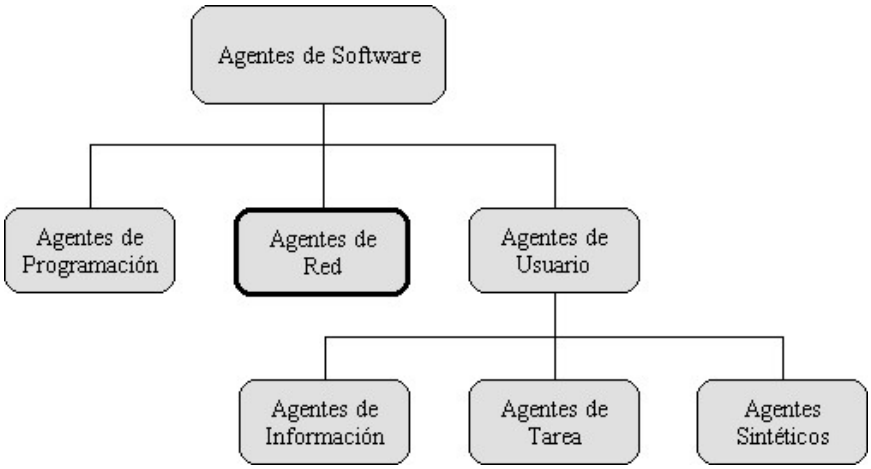

Figura 2.1 Taxonomía de Agentes. (Adaptada de Sánchez y Leggett [1997])

2.2 Agentes Móviles

Un agente móvil debe contener los siguientes modelos, según Green et al.

[1997]:

Modelo de Agente

> Este modelo define la estructura interna del agente como parte del agente móvil [Nwana 1996], en escencia define las características de autonomía, aprendizaje y cooperatividad.

Modelo de Ciclo de Vida

> Define los diferentes estados de ejecución de un agente móvil.

Modelo Computacional

> Define las habilidades computacionales de un agente, como la manipulación y control de instrucciones.

Modelo de Seguridad

> Se encarga de dos protecciones: la de los agentes sobre un nodo y la de los nodos sobre los agentes.

Modelo de Comunicación

> Es la implementación de un protocolo.

Modelo de Navegación

> Se refiere a todos los aspectos de movilidad.

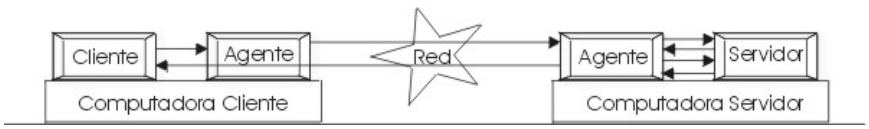

Figura 2.2 Paradigma de Agentes Móviles

2.3 Plataformas para Agentes

En esta sección revisaremos las plataformas en las que se pueden construir los sistemas de agentes móviles.

2.3.1 Aglets

Los aglets son agentes autónomos basados en Java, desarrollados por IBM. Proveen las capacidades básicas requeridas para la movilidad [Bigus 1998]. Un aglet refleja el modelo de applet en Java pero brindándole la

propiedad de movilidad.

Un aglet también puede ser un agente móvil porque soporta las ideas de ejecuciones autónomas y ruteo dinámico sobre sus itinerarios. Y tiene las siguientes características [Change y Lange 1996]:

Un esquema global único para agentes (Modelo de navegación/seguridad)

Un itinerario de viaje, para la especificación de patrones complejos de viajes con múltiples destinos y manejos de fallas automáticos (Modelo de navegación)

Un mecanismo *white-board* permitiendo que múltiples agentes colaboren y compartan información asíncronamente (Modelo de comunicación)

Un esquema de transmisión de mensajes que soporta una unión asíncrona desahogada tan bien como una comunicación síncrona entre agentes (Modelo de comunicación)

Una carga de clases dinámicamente que permite que el código Java de los agentes y la información de su estado viajen a través de la red (Modelo de navegación)

Un contexto de ejecución que proporciona un ambiente independiente del sistema actual sobre el cual se están ejecutando (Modelo computacional)

Desafortunadamente el modelo de ciclo de vida de un aglet es muy simple y algunas de sus desventajas se mencionan a continuación:

Soporte inadecuado de control de recursos

No tiene protección de referencias

No provee ayuda para la preservación y reanudación del estado de ejecución.

El ciclo de vida de un aglet [Lange y Oshima 1998] esta formado por las siguientes operaciones:

Creación

Clonación

Planificación

Retractación

Activación y Desactivación

Disposición

Mensajería

Los aglets utilizan el Protocolo de Transferencia de Agentes independiente de la plataforma para transferir agentes entre redes de computadoras.

El Protocolo de Transferencia de Agentes (ATP) [Lange y Yariv 1997] es un protocolo a nivel de aplicación para sistemas distribuidos basados en agentes, puede ser usado para transferir agentes móviles entre redes de computadoras. Mientras los agentes móviles pueden ser programados en diferentes lenguajes y para una variedad de plataformas de agentes, ATP ofrece la oportunidad para manejar la movilidad de los agentes de una forma general y uniforme.

El protocolo ATP esta basado en el paradigma petición/respuesta entre servicios entre agentes. Una petición ATP consiste de una línea de petición, campos de encabezado y contenido. La línea de petición especifica el método de la petición, y los campos de encabezado contienen los parámetros de la petición.

ATP define cuatro métodos estándar de petición:

1. Despachar, reconstruir el agente en otro nodo y comenzar su ejecución en él.
2. Retractar, traer de regreso a un agente, antes enviado a un nodo en específico.
3. Traer, recuperar información de identificación.
4. Comunicar, enviar mensajes entre agentes en ambas direcciones.

ATP soporta la técnica llamada *HTTP tunneling* que hace que una petición sea enviada fuera del *fire-wall* como una petición HTTP POST, la respuesta es regresada como una respuesta HTTP.

2.3.2 D'Agents

Agent Tcl [Kotz et al. 1996], ahora llamado D'Agents es una plataforma simple independiente del sistema de agente móvil. El modelo de navegación esta basado en un simple comando *agente_salta,* este comando puede aparecer en un agente y provocar que su estado y contexto de ejecución sea congelado y transportado a un nodo específico; esta habilidad es más sofisticada que la de los Aglets. El Modelo de Comunicación tiene tres comandos: *agente_enviar, agente_recibir* y *agente_reunir*. Cuando un agente quiere migrar a una nueva maquina, éste manda llamar a una simple función *agente_salta* , el cual automáticamente captura la completa información del estado del agente y la envía al servidor sobre una máquina destino; el servidor destino empieza apropiándose del ambiente de ejecución, cargando la información del estado y retornándolo a su lugar de origen.

D'Agents tiene importantes características, como:

Arquitectura Simple

Seguridad

Transparencia en la movilidad (TCP/IP)

Comunicación entre agentes (RPC)

El Lenguaje D'Agent es una extensión de Tcl/Tk que soporta la programación distribuida en la forma de agentes transportables. Tcl (*Tool Command Language*) es realmente dos cosas: un lenguaje script y un intérprete para este lenguaje, que es diseñado para ser fácilmente incorporado dentro de las aplicaciones [Kotz 1999].

El objetivo del proyecto D'Agents Dartmouth, según Gray [1995], es combatir algunas debilidades de los sistemas de agentes móviles, de la siguiente forma:

Reduciendo la migración a una simple instrucción y permitiendo que ésta ocurra en puntos arbitrarios.

Dando una comunicación transparente entre agentes.

Soportando múltiples lenguajes y mecanismos de transporte.

Ejecutándose en plataformas Unix y dando facilidades para otras plataformas.

Proporcionando seguridad efectiva y tolerancia a fallos en el mundo incierto del Internet.

Estado disponible al dominio público.

2.3.3. Voyager

Es una aplicación realizada 100% en Java, creada por la compañía ObjectSpace, su meta es proveer al programador un espacio para crear programas distribuidos rápidamente, fácilmente, con mucha flexibilidad y extensibilidad. Una de las grandes ventajas de este sistema es que soporta ambas arquitecturas: cliente-servidor y basada en agentes.

Características de Voyager [CS 696 1999]:

Habilitación remota de las clases

Inicialización por parte del cliente

Cargado dinámico de clases

Colección distribuida de basura

Agregación dinámica

CORBA, RMI, DCOM

Movilidad

Activación

Applets y Servlets

Servicio de nombres

Multi-reparto para objetos de Java distribuidos

Suscripción para publicar eventos remotos

Mensajería avanzada

2.3.4. Odyssey

General Magic Inc. fue el creador del primer sistema de agentes móviles denominado Telescript, cuyo periodo de vida fue muy corto pese a que estaba pensado para trabajar en una arquitectura de red. En respuesta a la popularidad de Internet y al gran auge del lenguaje de programación Java, General Magic decidió reimplementar todos los conceptos considerados en el desarrollo de Telescript pero ahora en este lenguaje. El resultado obtenido fue Odyssey, el cual es una librería de clases Java que permiten al usuario crear sus propias aplicaciones de agentes móviles [Odyssey 1997]. Maneja el Protocolo de Transferencia de Agentes Simples (SATP).

2.3.5. Concordia

Concordia [Concordia 1997] es un espacio de trabajo para desarrollar y manejar aplicaciones de agentes móviles eficientemente para accesar información en cualquier tiempo, en cualquier lugar y sobre cualquier dispositivo que soporte Java.

Las aplicaciones:

Procesan datos sobre los datos fuente

Procesan datos aún cuando el usuario esta desconectado de la red

Accesan y entregan la información a través de múltiples redes (LANs, Intranets e Internet)

Utilizan comunicación inalámbrica

Soportan múltiples dispositivos clientes, tales como computadoras de escritorio, portátiles, PDAs y teléfonos inteligentes

Concordia oculta las complejidades de programar una aplicación móvil a los programadores, desarrollando una aplicación agente-habilitado similar a un programa estacionario o no móvil. Los agentes mantienen su estado interno mientras viajan en la red, así que ellos pueden reanudar su ejecución al llegar a una nueva posición. Todos los agentes transportan trabajo que se maneja transparentemente sin la intervención del programador.

Un agente Concordia viaja en la red definido por su *Itinerario* . El Itinerario especifica a dónde viajará el agente y qué tareas deberá desarrollar cuando llegue. Los itinerarios de Concordia son especificados en tiempo de ejecución, los agentes pueden cambiar su propio itinerario basados en la información y eventos descubiertos a medida que los agentes viajan.

Algunas de las características que provee Concordia son:

Servicios de comunicaciones TCP/IP

Manejo avanzado de funciones

Colaboración

Servicio de puentes

Persistencia y manejo de cola

Itinerario

Servicio de nombres

Estructura de seguridad Concordia

Transportador de agentes ligero API

Cifrado

Componentes de Concordia:

Administrador de agentes

Administrador de seguridad

Administrador de persistencia

Administrador de comunicación inter-agente

Administrador de cola

Administrador de directorio

Director de administración

Librería de herramientas para agentes

Concordia no impone un protocolo o un servicio de computación distribuida como propio. Concordia emplea los servicios de comunicación existentes de TCP/IP, ampliamente disponibles y compatibles con redes de área local, y redes privadas y públicas inalámbricas.

2.3.6 Comparación de Plataformas

En la tabla 2.1 se presentan de una manera comparativa las plataformas antes mencionadas. Esta tabla es una extensión de los trabajo realizados por Pérez [1998] y Chevalier [2000], para su mayor comprensión a continuación se describen los siguientes términos:

Tipo de migración. Se dice que es *fuerte* cuando el agente migra con su código y estado de ejecución; y *débil* cuando solamente migra con su código y llama al método de ejecución al llegar.

Mecanismo de comunicación. Es la forma bajo la cual los agentes consiguen viajar a través de los nodos.

Ligado dinámico. Para soportar la migración, los lenguajes deben proveer mecanismos para envío de código y datos hacia otro nodo y dinámicamente ligar el código y los datos de un agente. Se dice que éste de *código remoto* cuando permite a los programadores implementar técnicas de "código en demanda", esto es, aplicaciones que cargan su código dinámicamente desde la red de acuerdo a diferentes estrategias; y de *recurso local* cuando permite que un agente accese a los recursos locales del nodo receptor.

Comunicación entre agentes. Es la forma de enviar y recibir mensajes; *asíncronos* se refiere a que no tienen que estar los dos agentes disponibles al mismo tiempo para poder comunicarse; sin embargo en los *síncronos* , sí se requiere que estén disponibles al mismo tiempo, por lo que no es tan segura la comunicación.

Tabla 2.1 Tabla Comparativa de Lenguajes de Programación para Agentes

	Versión	Organización	Estado	Plataformas	Lenguajes Requeridos	Tip Mi
Aglets	1.0.3 2.0.1	I BM Tokyo Research Lab.	En desarrollo Última actualización 22/01/2002	Win 32 (Win 95/98/NT) OS/2, AIX 4.X	Java 1.1 Java 1.2	Dé

				Solaris for SPARC/x86		
D'Agents	2.1	Darmouth College	Concluído	Unix	Tcl, Java, Scheme	Fu (Tc Jav Dé (Sc
Voyager	4.0	Object Space Inc.	En desarrollo Última actualización 31/10/2000	Java	Java 1.1 y 1.2	Dé
Odyssey	Ninguna	General Magic Inc.	Concluído	Java	Java	Dé
Concordia	1.1.7	Mitsubishi Electric ITA	En desarrollo Última actualización 1/01/2001	Win 32, Solaris, Linux HP/UX, AIX	Java 1.1	Dé

2.4 Protocolos de Comunicación

Los agentes que forman un sistema pueden colaborar entre ellos para alcanzar sus tareas respectivas, dicha colaboración se realiza mediante un protocolo de comunicación comprensible por los agentes del sistema. A continuación se presentan algunos protocolos importantes.

2.4.1 MICK

Es un Marco de Inter-Comunicación basado en KQML definido por Barceinas [1998], en el que cada agente cuenta con un ruteador capaz de enviar y recibir mensajes en KQML, reconocer un conjunto de palabras y

seguir un protocolo.

Sus elementos son:

Facilitador, es el que define la arquitectura de coordinación asistida, ya que se encarga de aceptar conexiones, de recibir solicitudes de registro de los ruteadores y de mantener tablas con las direcciones de quienes se han registrado.

Ruteador, se encarga de registrar a su agente con el facilitador, recibir mensajes, interpretarlos para solicitar la acción correspondiente a quien esta sirviendo, construir mensajes y enviarlos.

KQML, es un lenguaje de tipo declarativo que se enfoca a la parte pragmática de la comunicación.

Vocabulario, es el conjunto de palabras que los ruteadores reconocen en el contenido de los mensajes, y

Protocolo, maneja la comunicación entre el UAD (Director de Agentes de Usuario) y los agentes, entre el UAD y ALiS (Servicios de Biblioteca Activa), y entre el UAM (Administrador de Agentes de Usuario) y ALiS.

2.4.2. CORBA

CORBA (*Common Object Request Broker Architecture*) [CORBA 2000] es una especificación abierta de OMG para las arquitecturas de las aplicaciones que trabajan sobre redes. Su interoperabilidad proviene de dos partes principales: OMG Lenguaje de Definición de Interface (OMG IDL) y los protocolos estandarizados GIOP e IIOP [Orfali y Harkey 1998].

El Lenguaje de Definición de Interface (IDL) , puede ser usado para definir la interacción de objetos distribuidos sobre la red, la interfaz permite enviar objetos a métodos remotos como parámetros, ya sea pasar el objeto actual o una copia de él, y define la interacción del código remoto con el código local.

El Protocolo Inter-ORB General (GIOP) , especifica una sintaxis de transferencia estándar y un conjunto de formatos de mensajes para comunicaciones entre ORBs, un ORB (*Object Request Broker*) es un transporte de objetos, que toma a los objetos transparentemente hace peticiones a - y recibe respuestas de - otros objetos localizados local o remotamente. GIOP define siete formatos de mensajes que cubren todas las semánticas de petición/respuesta ORB.

El Protocolo Inter-ORB Internet (IIOP), especifica como los mensajes GIOP se intercambian utilizando conexiones TCP/IP. El IIOP especifica un protocolo de interoperabilidad estandarizado para el Internet, proporcionando interoperación "fuera de la caja" con otros ORBs compatibles basados en TCP/IP. Ambos protocolos: IIOP y DCE/ESIOP (

Environment-Specific Inter-ORB Protocol) tienen que integrarse a mecanismos para transmitir implícitamente el contexto de los datos que esta asociado con la transacción o los servicios de seguridad.

2.4.3 RMI

RMI (*Remote Method Invocation*) [Java RMI 2000] habilita al programador para crear aplicaciones basadas en tecnología Java distribuida, en la cual los métodos de objetos en Java remotos pueden ser invocados desde otras máquinas virtuales de Java, posiblemente en diferentes servidores. El programa puede hacer una llamada sobre un objeto remoto, una vez que éste obtiene una referencia al objeto remoto, buscándolo en el *bootstrap* llamando un servicio proveído por RMI o recibiendo la referencia como un argumento o un valor regresado. RMI utiliza la serialización de objetos con parámetros oficiales y no oficiales, y no trunca los tipos, soportando un verdadero polimorfismo orientado a objetos.

El sistema RMI consiste de cuatro capas:

1. Capa de aplicación
2. Capa de fragmento/esqueleto, transmite los datos de la capa de aplicación a la capa de referencia remota
3. Capa de referencia remota, es responsable de proveer la habilidad para soportar variaciones de referencias remotas o invocaciones a protocolos independientes del fragmento del cliente y del esqueleto del servidor
4. Capa de transporte, es responsable de establecer conexiones a direcciones remotas, manejar las conexiones, escuchar sus llamadas entrantes, manejar una tabla de objetos remotos que residen en el mismo espacio de direcciones, establecer una conexión de una llamada entrante localizando su despachador y pasándole su conexión.

Actualmente Java RMI utiliza una combinación de serialización java y el Protocolo de Método Remoto de Java (JRMP) para una búsqueda de invocaciones de métodos remotos.

2.5 Tecnologías de Interoperabilidad

Existen varias iniciativas dedicadas a crear colecciones interoperables, así como proveer servicios para accesar a dichas colecciones; y de esta forma ayudar a crear una federación.

2.5.1 Iniciativa de Archivos Abiertos (OAI)

Desarrolla y promueve estándares de interoperabilidad que intentan facilitar la difusión eficiente de contenidos. La Iniciativa de Archivos Abiertos [OAI 2001] tiene como objetivo facilitar el acceso a colecciones heterogéneas como un medio de incrementar la disponibilidad en la comunicación de investigadores.

Características:

Habilita la compartición de publicación de metadatos y texto completo por bibliotecas digitales.

Estandariza mecanismos de bajo nivel para compartir contenidos de bibliotecas.

Construye servicios administrativos en meta-bibliotecas.

Instala mecanismos organizacionales para soportar los procesos técnicos.

El **Protocolo de la Iniciativa de Archivos Abiertos para la Recolección de Metadatos** , tiene como objetivo promover un espacio de trabajo interoperable independiente que puede ser usado por una variedad de comunidades que tienen como misión la publicación de contenidos en Internet. Tiene dos clases de participantes:

1. *Proveedores de Datos* , administran sistemas que soportan el protocolo OAI como un medio de exposición de metadatos acerca del contenido en sus sistemas.

2 .*Proveedores de Servicios* , emiten peticiones del protocolo OAI a los sistemas de proveedores de datos y usa los metadatos regresados como una base para construir servicios de valores agregados.

Cada formato de metadatos que es incluido en registros por el protocolo OAI es identificado dentro de las peticiones del protocolo a un acervo por un *prefijo de metadatos* y a través de múltiples repositorios por el URL del *esquema de metadatos*.

Este protocolo define un mecanismo para la recolección de registros conteniendo metadatos de los acervos; asigna un formato de metadatos requerido para el propósito de interoperabilidad, éste formato es Dublín Core; sin embargo este protocolo no limita el formato de metadatos, éstos pueden ser registros estructurados como datos de XML, los cuales tienen una correspondencia con el esquema de XML para validación. Una especificación más completa se menciona más adelante en el Apéndice A.

Protocolos como Z39.50 tienen una funcionalidad mas completa, trabajando con el manejador de sesión y el activador de resultados, y permitir la especificación de predicados que filtran los registros

regresados. Esta funcionalidad viene a incrementar la dificultad de implementación y los costos.

2.5.2 Protocolo Z39.50

El protocolo Z39.50 deriva su nombre de haber sido desarrollado del comité número 39 del Instituto Nacional Americano de Estándares (ANSI), y por ser el estándar número 50 de la Organización Nacional de Estándares de Información (NISO). Su nombre oficial es *"Information Retrieval (Z39.50) Application Service Definition and Protocol Epecification for Open Systems Interconection"*.

Es un protocolo que permite que una computadora (cliente) busque y recupere información de otra (servidor). Z39.50 está basado en una vista abstracta de búsquedas en bases de datos, el protocolo asume que el servidor empieza con un conjunto de bases de datos con índices de búsqueda; las interacciones están basadas en el concepto de *sesión* .

El protocolo Z39.50 [Z39.50 1998]:

Es un estándar internacional para la comunicación entre sistemas. Provee poderosos medios de localización de registros dentro de una o más bases de datos sobre un servidor único.

Considera que el manejo de la información posee dos componentes principales: la selección de la información basada en algunos criterios y la recuperación de la información. Su función es proporcionar un lenguaje común para ambas actividades.

Facilita la interconexión entre los usuarios y las bases de datos donde se encuentra la información que necesitan a partir de una interfaz común y de fácil manejo, independientemente del lugar en que las bases de datos se encuentren, de la estructura de la base de datos y la forma de acceso.

Especifica el formato y los procedimientos que gobiernan el intercambio de mensajes entre un cliente y un servidor. El cliente puede enviar una búsqueda, indicar una o más bases de datos, e incluir una consulta y también parámetros, los cuales determinan si los registros identificados por la búsqueda podrían ser devueltos como parte de la respuesta. El servidor responde con una cantidad de registros identificados. El cliente puede, entonces, recuperar los registros seleccionados, asumiendo que los registros forman un conjunto de resultados y pueden ser referenciados por su posición dentro del conjunto.

El formato básico usado para el intercambio de registros bibliográficos es el MARC. La capacidad de presentar y transferir en formato MARC, permite al cliente utilizar esa información para un procesamiento posterior.

El termino "meta" viene del griego que significa algo tan alto o más fundamental que lo natural. Metadatos son entonces, datos acerca de

otros datos. Edward Fox en su aportación en [Baeza y Neto 1999] menciona que los metadatos se refieren a la descripción de un objeto digital.

Swick [1997] describe " *metadata* " como una "máquina comprensible de información acerca de objetos web". Los esquemas de metadatos vinculan un grupo de código o etiquetas que describen el contenido y/o el contenedor de objetos digitales. Su propósito es facilitar y mejorar la recuperación de información. En las siguientes subsecciones se describen algunos estándares de metadatos.

2.6.1 Dublín Core

La Iniciativa de Metadatos Dublín Core (DCMI) [Dublín Core 2001] es un foro abierto comprometido en el desarrollo de estándares de metadatos interoperables en línea que soportan un rango ancho de modelos de negocios. Es una organización dedicada a desarrollar vocabularios de metadatos especializados para describir recursos que habilitan sistemas inteligentes de información.

Cada elemento de Dublín Core es definido como un conjunto de atributos provenientes de ISO/IEC 11179, y éstos están clasificados en tres grupos que indican la clase o el ámbito de la información que se guarda en ellos:

1. Elementos relacionados principalmente con el contenido del recurso [Título, Tema, Descripción, Fuente, Lenguaje, Relación, Cobertura]
2. Elementos relacionados principalmente con el recurso cuando es visto como una propiedad intelectual [Autor, Editor, Colaboradores, Derechos]
3. Elementos relacionados principalmente con la instanciación del recurso [Fecha, Tipo de Recurso, Formato, Identificador]

Las principales características de Dublín Core son:

Simplicidad

Interoperabilidad Semántica

Concensos Internacionales

Extensibilidad

Modularidad

El Conjunto de Elementos de Metadatos Dublín Core (DCMES) fue el

primer estándar de metadatos distribuido de DCMI. DCMES provee la semántica de un vocabulario para describir el "core" (núcleo) de las propiedades de la información, tales como "Descripción", "Autor" y "Fecha".

2.6.2 MARC

Un registro MARC (*MAchine Readable Cataloging*) fue diseñado para detallar información bibliográfica, envuelve tres elementos: la estructura del registro, la designación del contenido y los datos contenidos del registro [MARC 2001].

Los formatos son definidos por cinco tipos de datos: Bibliográfico, Compañía/Personal, Autoritativo, Clasificación e Información de Comunidad:

1. Formato para datos Bibliográficos, contiene especificaciones del formato para codificar elementos de los datos necesarios para describir, recuperar y formas varias de control de material bibliografico.

2. Formato para datos *Holdings* , contiene especificaciones del formato para codificar elementos de los datos pertinentes para holdings y localización de datos para todas las formas de material.

3. Formato para datos Autoritativos, contiene especificaciones del formato para codificar elementos de los datos que identifican o controlan el contenido y la designación del contenido de aquellas porciones del registro bibliográfico que pueden ser tema para control autoritativo.

4. Formato para datos Clasificación, contiene especificaciones del formato para codificar elementos de los datos relacionados con los números de clasificación y los titulos asociados con ellos.

5. Formato para Información de Comunidad, provee especificaciones del formato para registros que contienen información acerca de eventos, programas, servicios, etc. por lo que esta información puede ser integrada dentro del mismo acceso a catálogos públicos como datos en otros tipos de registros.

Los formatos MARC 21 son formatos de comunicación, primeramente diseñados para proveer especificaciones para el intercambio de información bibliográfica y relacionada entre sistemas. Ellos son ampliamente usados en una variedad de ambientes de intercambio y de procesos. Como formatos de comunicación, no manejan almacenamiento interno o despliegan formatos para ser usados por sistemas individuales.

Un registro MARC consiste de tres principales secciones: director, directorio y campos variables. El protocolo Z39.50, el cual habilita la

búsqueda y recuperación de información bibliográfica en Internet, está particularmente diseñado para trabajar con registros MARC.

Algunas de las principales características del registro MARC, mencionadas por Heery [1996] son:

Orientado a material bibliográfico (documentos, video, audio)

Estructura de registro conformada según la norma ISO 2709

2.6.3 XML

XML no es un simple lenguaje de marcas predefinido, es un metalenguaje (un lenguaje que describe otros lenguajes), en el cual se diseñan marcas (o etiquetas) propias para diferentes clases de documentos.

Cada documento XML [XML 1998] tiene una estructura tanto lógica como física. Físicamente, el documento está compuesto de unidades llamadas *entidades* . Una *entidad* puede referirse a otras entidades con el fin de causar su inclusión en el documento. Un documento comienza en una ”raíz” o entidad documento. Lógicamente, el documento está compuesto de declaraciones, elementos, comentarios, referencias de carácter e instrucciones de proceso.

Características de XML:

Multifuncional

Simplicidad

Adaptativo

XML provee un mecanismo, la declaración de tipo de documento (*document type declaration*), con el fin de describir restricciones en la estructura lógica y soportar el uso de unidades de almacenamiento predefinidas. La declaración de tipo de documento de XML contiene o apunta a declaraciones de marcación que proveen una gramática para una clase de documentos. Esta gramática es conocida como una Definición de Tipo de Documento (*Document Type Definition*), o DTD.

El formato de Intercambio de Metadatos de XML (XMI) especifica un modelo de intercambio de información abierta que da a los desarrolladores la habilidad de intercambiar datos de programación sobre el Internet dentro de un camino estandarizado, trayendo con sigo consistencia y compatibilidad en las aplicaciones creadas en ambientes colaborativos. XML contiene los siguientes elementos estructurales:

Encabezado

Contenido

Diferencias

Extensiones

2.7 Resumen

En este capítulo se describe la tecnología de agentes, conociendo su clasificación y profundizando en los agentes móviles, analizando algunas plataformas existentes para su programación; así como protocolos de comunicación que pueden complementar la implementación de agentes. Se analizaron algunas tecnologías que ayudan en la formación de federaciones de bibliotecas digitales, como la Iniciativa de Archivos Abiertos (OAI) y Z39.50, siendo necesario conocer que son los metadatos y cuales son los formatos más comunes.