SISTEMAS DISTRIBUIDOS CUESTIONARIO 1

Sistema distribuido: Colección de computadoras independientes que le dan al usuario la idea de constituir un único sistema coherente.

Como deben ser: ser abierto y debe ser escalable.

Como también se los conoce: como Middleware.

Como deben permanecer el hardware y software: diferencias entre los distintos tipos de hardware y sistemas operativos deben permanecer ocultas.

Migración, replicación, reubicación y concurrencia:

- **Migración:** que un recurso pudiera moverse a otra ubicación
- **Replicación:** del número de copias de un recurso
- **Reubicación:** que un recurso pudiera moverse a otra ubicación mientras está en uso
- **Concurrencia:** que un recurso pueda ser compartido por varios usuarios que compiten por él.

Grado de apertura:

- **Interoperabilidad:** las implementaciones de sistemas y componentes de fabricantes pueden coexistir y trabajar juntos.
- **Extensible:** debe ser fácil de configurar para componentes diferentes, aparte de agregar nuevos o reemplazar los existentes sin afectar a los que permanecen en su lugar
- **Flexibilidad:** los componentes son relativamente pequeños y fáciles de reemplazar o adaptar

Escalabilidad: Se puede medir desde varios puntos de vista:

- **Respecto a su tamaño:** podemos agregar fácilmente recursos y usuarios. Hay que evitar los algoritmos centralizados.
- **Desde el punto de vista geográfico:** usuario y recursos pueden radicar muy lejos uno de los otros
- **Escalabilidad administrativa:** puede ser fácil de manejar incluso involucrando muchas organizaciones administrativas diferentes.

Técnicas de escalamiento:

- Sirven para **disminuir el impacto de los problemas de escalar un sistema**, las cuales son:
    - Ocultar las latencias de comunicación
    - **Distribución:** dividir un componente en partes más pequeñas y distribuir dichas partes a lo largo del sistema
    - **Replicación:** replicar los componentes a lo largo del sistema distribuido

Sistema distribuido ideal: debe tener una red confiable, segura y homogénea por lo que la topología no debe cambiar, su latencia debería ser cero, el ancho de banda infinito, el costo de transporte igual a cero, que exista un administrador.

Que se tiene en cuenta en todos los sistemas distribuidos: (debe tener en cuenta)

- Arquitectura
- Procesos
- Comunicación
- Asignación de nombres
- Sincronización
- Consistencia
- Replicación
- Tolerancia a fallas
- Seguridad

Sistemas distribuidos basados en la coordinación:

- Los mensajes son direccionados por tema, por lo que no tienen la dirección de su receptor.
- Los procesos que desean recibir mensajes deben suscribirse a un tema específico.
- El middleware se encarga de que los mensajes se encaucen desde los editores hasta los suscriptores.

Sistemas distribuidos de cómputo:

- **En cluster:** El HW consta de una colección de estaciones de trabajo similares o PC, conectadas cercanamente por medio de una LAN de alta velocidad. Cada nodo ejecuta el mismo Sistema Operativo.
- **En malla (grid):** consta de sistemas distribuidos construidos generalmente como un conjunto de sistemas de cómputo en donde cada sistema podría caer dentro de un dominio administrativo distinto, y obviamente distintas sus características de HW, SW y de tecnología de red.

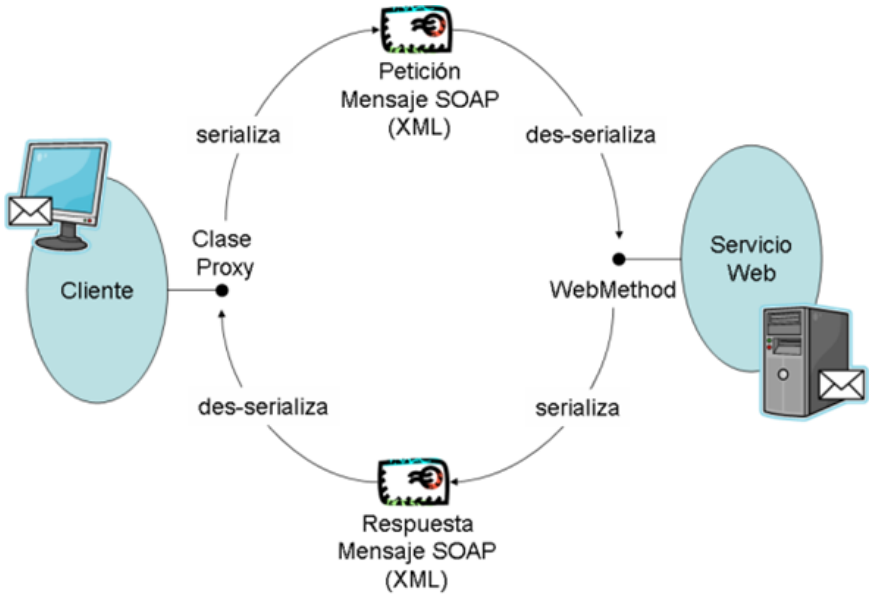
A que se conoce como RPC transaccional: es la llamada a un procedimiento remoto, el cual es un programa que utiliza una computadora para ejecutar código en otra máquina remota sin tener que preocuparse por las comunicaciones entre ambas, de forma que parezca que se ejecuta local. Los clientes usan procesos que se ejecutan en CPUs remotas, pueden ser síncronas o asíncronas.

Desafíos sistemas distribuidos:

- heterogeneidad de redes
- HW de computadores
- sistemas operativos
- lenguajes de programación e implementación de distintos desarrolladores.

Gráfico servicio web:

(graficaría)



Describa WSDL – UDDI Y SOAP:

- **WSDL (Web Service Description Language**): lenguaje formal similar a los lenguajes de interfaz usados para soportar la comunicación basada en RPC. Puede ser transformado en forma automática en Resguardo del lado del cliente y del lado del servidor.
- **UDDI (Universal Description, Discovery, and Integration):** define un modo de publicar y encontrar información sobre servicios Web.
    - **UDDI tiene dos funciones:**
        - Es un protocolo basado en SOAP que define cómo se comunican los clientes con los registros UDDI.
        - Es un conjunto de registros duplicados globales en particular.
    - **Registro UDDI:**
        - **Páginas blancas:** dirección, contacto y otros identificadores conocidos.
        - **Páginas amarillas:** categorización industrial basada en taxonomías.
        - **Páginas verdes:** información técnica sobre los servicios que aportan las propias empresas.
- **SOAP (Simple Object Access Protocol):** estructura donde una gran parte de la comunicación entre dos procesos puede ser estandarizada.

Que aspectos se evalúan en los webs services:

- **Sincronización**: es necesario que los servidores se sincronicen entre sí y que sus acciones sean coordinadas.
- **Consistencia y Replicación**: se idean estrategias para guardar en caché y replicar contenido tanto estático como dinámico, es decir, documentos generados a consecuencia de una solicitud. El almacenamiento en caché del lado del cliente ocurre generalmente en 2 lugares:
    - Los browsers cuentan con un almacenamiento en la caché que se pueda configurar indicando en qué momento se deberá verificar la consistencia
    - El sitio de un cliente ejecuta con frecuencia un proxy web. Un proxy web acepta solicitudes de clientes locales y las transfiere a servidores web. Cuando llega una respuesta, el resultado se transfiere al cliente.
      La replicación sirve para que el sitio sea continuo y de fácil acceso.
- **Algunas métricas**:
    - **Métrica de latencia:** mide el tiempo requerido para realizar una acción

    - **Métrica de uso de red:** se relaciona con el ancho de banda consumido en función del número de bytes que se van a transferir
    - **Métrica espacial:** mide la distancia entre nodos en función del número de conexiones intermedias de enrutamiento a nivel de red o de conexiones intermedias entre sistemas autónomos
    - **Métricas de consistencia:** informa en qué medida o grado se está desviando una réplica de su copia maestra.
    - **Métrica financiera:** forma otra clase para medir que tan bien está funcionando una CDN. Está estrechamente relacionada con la infraestructura real de Internet.
- **Tolerancia a fallas:** se logra principalmente por medio de almacenamiento en caché del lado del cliente y replicación de servidores. Los problemas se agravan para servicios diseñados para manejar fallas bizantinas, un servicio que tolere este tipo de fallas (BTS) puede necesitar actuar como cliente de otro servicio no replicado.
- **Seguridad**: el método para establecer un canal seguro en la web es usar una capa de socket seguro (SSL), que fue implementada originalmente por Netscape, hoy es conocido como protocolo de seguridad en la capa de transporte (TLS)

Que propone Castro y Liskov: Una solución al sistema BFT debe manejar:

- Los clientes de un servicio BTF deberán ver al servicio como a cualquier otro servicio web
- Deberá garantizar la consistencia interna cuando actúa como cliente
- Deberán tratarse como un servicio BFT que actúa como cliente y como una sola entidad.

Canal seguro se establece en 2 (dos) fases:

- El cliente informa al servidor acerca de algoritmos criptográficos que puede manejar, así como también, cualquier método de compresión que soporte.
- La segunda fase es la autenticación: siempre se requiere que el servidor se autentique a sí mismo, por lo que el cliente pasa un certificado que contiene una clave pública firmada por una autoridad de certificación. Si el servidor requiere que el cliente se autentifique, también el cliente deberá enviar un certificado al servidor.

Modelos arquitectónicos: Estilo arquitectónico se formula en términos de componentes, el intercambio entre componentes y cómo estos elementos se pueden configurar juntos en un sistema.

- **Cuales son:**
    - en capas
    - basadas en objetos
    - centradas en datos
    - basadas en eventos.

Invocaciones RPC – RMI – CORBA y compárelos:

- **RPC:** lógica de negocios escrita como procedimientos que se invoca en forma remota
- **RMI:** lógica de negocios escrita como objetos cuyos métodos son invocados en forma remota
- **CORBA:** lógica de negocios escrita como objetos que son invocados en forma remota.
- **Comparación:**
    - **RMI** es más lento que **RPC** por la interpretación byte-code en la JVM, pero se puede mejorar con el compilador JIT
    - **RMI** tiene una ventaja adicional sobre **RPC** porque pertenece al mundo orientado a objetos de Java**.**
    - Los sistemas **RPC** tradicionales son lenguajes neutrales, de modo que no pueden proveer la funcionalidad que no estén disponibles en sus plataformas destino. **RMI** por su naturaleza Java se conecta a los sistemas usando métodos nativos, por lo que puede proveer tecnología distribuida en forma directa, natural y potente.

Construir un web Service:

- Archivo
- Nuevo Proyecto
- Sitio web vacío de ASP.NET
- En la solución se agrega un elemento
- Servicio Web asmx
- Para invocarlo tenemos que agregar referencia de servicio
- Servicio Web de esta solución.

Picamos en invocar

Web service da la respuesta en XML

Vinculación física entre cliente y servidor:

- En el pop up sitio web picamos agregar referencia de servicio
- Luego avanzada y agregar referencia web
- Alternativas
    - Servicio web de esta solución
    - Servicio web de la maquina local
    - Examinar servidores UDDI de la red local
- Elegimos servicio web de esta solución
- Y vamos a ver el web service y elegimos agregar referencia web
- Finalmente se declara al cliente como indicar y se ejecuta

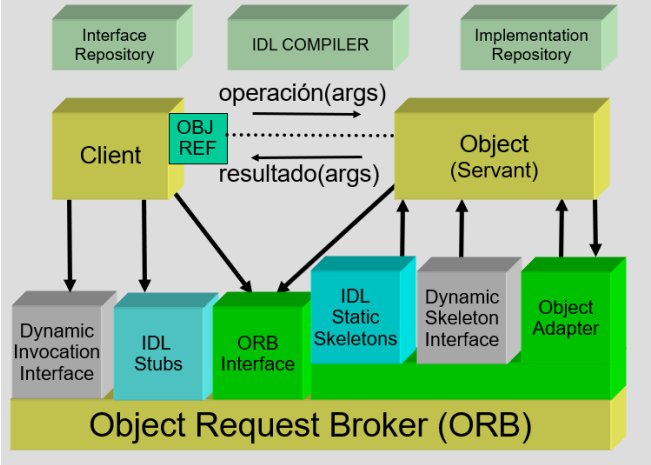
Construir aplicación CORBA:

- Definir la interfaz del objeto u objetos utilizando IDL
- Compilar el fichero IDL. Esto produce el código stub y skeleton que proporciona transparencia en la localización.
    - Esto significa que cooperan con la librería ORB para convertir una referencia a un objeto en una conexión por red a un servidor remoto y codifican los argumentos proporcionados en una operación sobre la referencia del objeto, la transportan al método correcto en el objeto denotado por dicha referencia, ejecutan el método y devuelven los resultados
- Identificar las interfaces y clases generadas por el compilador IDL que necesitaremos usar o especializar para invocar o implementar operaciones
- Escribir código para inicializar el ORB e informarle de cualquier objeto CORBA que hayamos creado
- Compilar todo el código generado junto con el código de nuestra aplicación con un compilador Java
- Ejecutar la aplicación distribuida.

IDL: lenguaje declarativo para la definición de interfaces de objetos CORBA. Es una forma independiente del lenguaje en la que los programadores y los usuarios de los objetos pueden estar seguros de que se invoca a la operación correcta del objeto correcto, aunque la única información adicional que se necesita es la referencia al objeto.

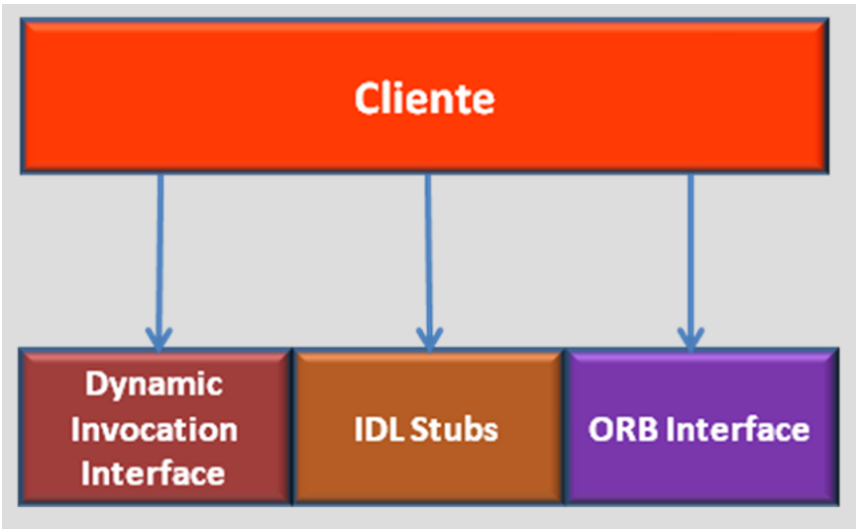
CORBA arquitectura:

- **Elementos:**

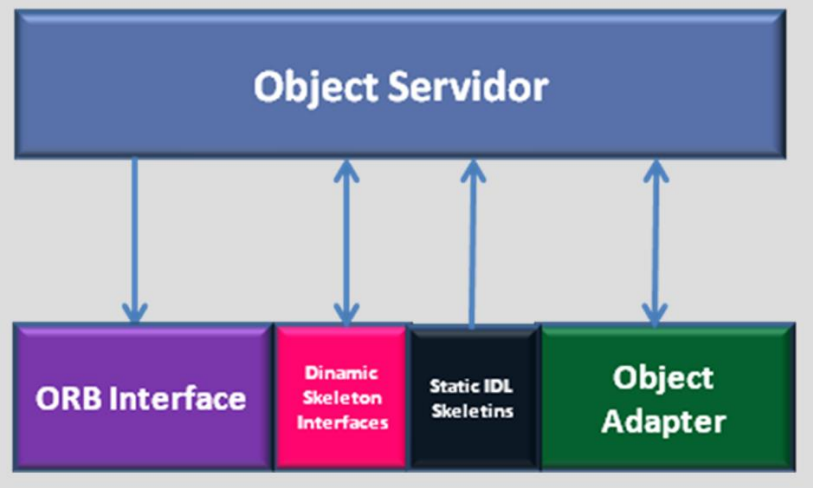


  - Adaptador de Objetos (OA)
  - Stub and Skeleton
  - Repositorio de Interfaces (IR)
  - Repositorio de Implementaciones
  - Lenguaje de definición de Interfaces (IDL)
  - Objeto Referencia

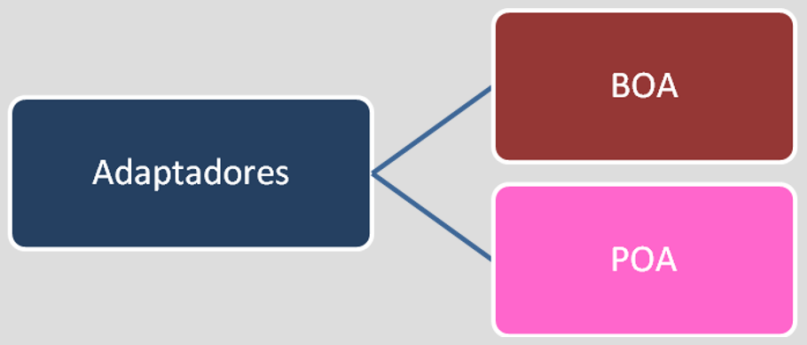
- **Cliente:**



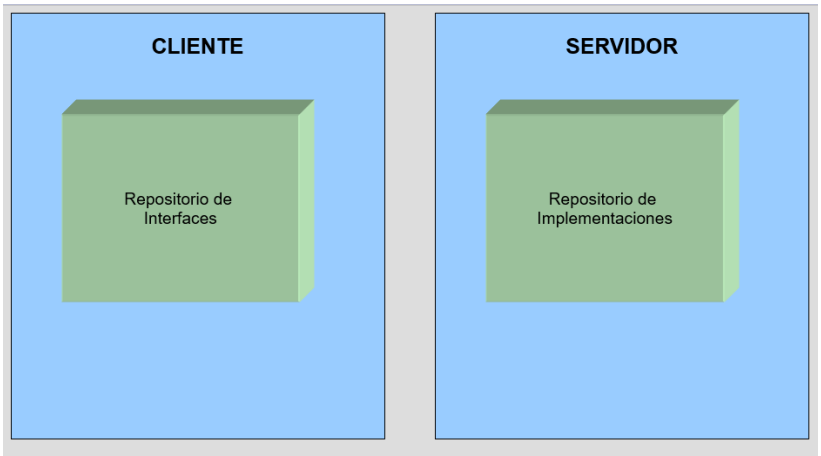
- **Implementación**



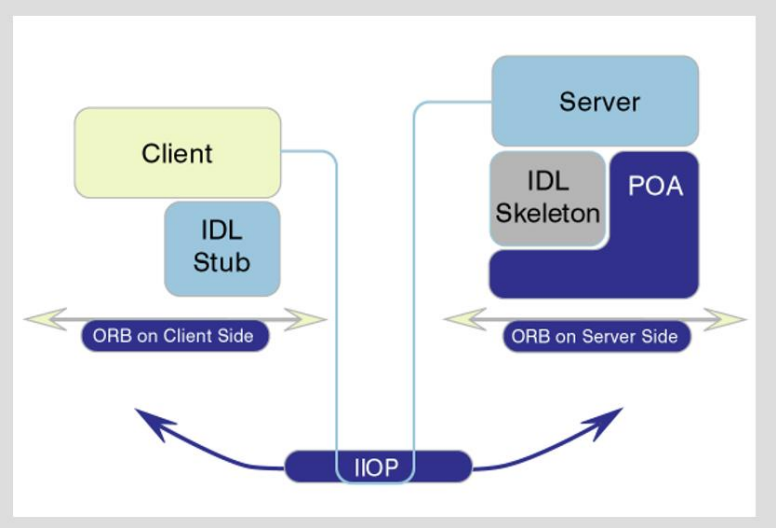
- **Adaptadores**



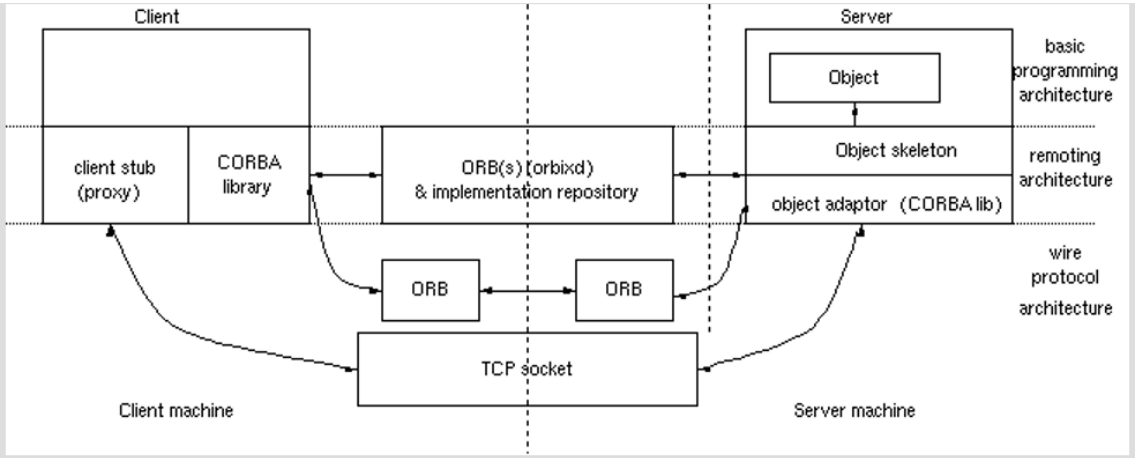
- **Repositorios**



- **Internet Inter-ORB Protocol**



- **Ejemplo de implementación**



SERVICIO WEB CUESTIONARIO 2

Servicio web: tecnología que utiliza un conjunto de protocolos y estándares que sirven para intercambiar datos entre aplicaciones (las aplicaciones pueden estar en lenguajes de programación diferentes y sobre cualquier plataforma).

SOA: SOA (Service Oriented Architecture) es un tipo de arquitectura de software que se basa en la integración de aplicaciones mediante servicios.

Que permite SOA: Permite la creación de una arquitectura que se integra (o forma parte) de sistemas de información y/o aplicaciones que puedan llegar a ser bastante escalables, que permitan modelar todo el proceso de negocios de una organización además es capaz de invocar y exponer.

Que son SOAP y REST relacionado con SOA: Son servicios que representan dos tipos muy diferentes de una implementación de arquitectura SOA.

REST: (Representational state transfer) Patrón de arquitectura construido bajo verbos simples que se adaptan fácilmente con HTTP

SOAP: (simple object access protocol) Protocolo de mensajería basado en XML que puede utilizarse por una variedad de protocolos de transporte (como HTTP)

Como se manifiestan los diferentes estados durante el ciclo de vida para SOAP y REST:

- **SOAP:** el movimiento por los diferentes estados se realiza interactuando con un extremo único del servicio que puede encapsular y proporcionar acceso a muchas operaciones y tipos de mensaje.
- **REST:** se permite un número limitado de operaciones y dichas operaciones se aplican a recursos representados y direccionados por direcciones HTTP, los mensajes capturan el estado actual o requerido del recurso (lo que hace que REST funcione mejor en aplicaciones WEB donde se puede hacer uso de HTTP para tipos de datos que no son XML).

SOAP permite a los diseñadores: encapsular la complejidad del sistema, dando lugar a interfaces generadas automáticamente que permiten facilitar el diseño del sistema.

SOAP como protocolo: Protocolo estándar que define como dos objetos en diferentes procesos pueden comunicarse por medio de intercambio de datos XML

SOAP relacionado con la robustez: Es bastante robusto, debido a que tiene un tipado más fuerte que REST, además de que permite agregar los atributos y permite definir espacios de nombre, evitando la ambigüedad.

Tipo de tecnología es rest: REST es un web service para sistemas distribuidos que utiliza el protocolo HTTP para transportar datos.

Tipos de datos transmite rest: Permite transmitir prácticamente cualquier tipo de dato debido a que está definido en el header content type.

- XML
- JSON (más utilizado por su velocidad y tamaño)
- Binarios (imágenes y documentos)
- Texto

Con que modelo está relacionado REST: con el modelo MVC.

Cual está fuertemente acoplado: SOAP.

Característica saliente de REST y SOAP:

- **REST:** Componentes débilmente acoplados.
- **SOAP:** Componentes fuertemente acoplados.

Ventaja para REST y SOAP:

- **REST:** Fácil de construir y adoptar.
- **SOAP:** Generalmente fácil de utilizar.

Característica tecnológica para REST y SOAP

- **REST:** Pocas operaciones con muchos recursos
- **SOAP:** Muchas operaciones con pocos recursos

Protocolos involucrados para REST y SOAP

- **REST:**
    - HTTP:
        - GET
        - POST
        - PUT

        - DELETE
    - XML auto descriptivo
    - Sincrono
- **SOAP:**
    - SMTP
    - HTTP POST
    - MQ
    - Tipado fuerte, XML schema
    - Síncrono y asíncrono

Respecto a seguridad REST y SOAP:

- **REST:**
    - HTTPS.
    - Comunicación punto a punto y segura.
- **SOAP**
    - WS security.
    - Comunicación origen y destino segura.

Paginas blancas, amarillas y verdes; y con que están relacionadas:

Todas esas páginas son un tipo de registro del UDDI, (Universal Description, Discovery and Integration) que es un registro en el catálogo que se hace en XML.

El registro de un negocio en UDDI tiene tres partes:

- **Páginas blancas:** dirección, contacto y otros identificadores conocidos.
- **Páginas amarillas:** categorización industrial basada en taxonomías.
- **Páginas verdes:** información técnica sobre los servicios que aportan las propias empresas.

Stack de web services:

Web Services Protocol Stack: es una colección de protocolos y estándares para redes de Computadores que son utilizados para definir, localizar, implementar y hacer que un Servicio Web interactúe con otro. La Pila de Protocolos para servicios está comprendida principalmente por cuatro áreas:

- **Servicio de Transporte:** responsable del transporte de mensajes entre las Aplicaciones de red y los protocolos en los cuales se incluyen protocolos tales como HTTP, SMTP, FTP, así como también el más reciente Blocks Extensible Exchange Protocol (BEEP).

- **Mensajería XML:** responsable por la codificación de mensajes en un formato común XML así que ellos puedan ser entendidos en cualquier extremo de una conexión de red. Actualmente, esta área incluye protocolos tales como XML-RPC, SOAP y REST.

- **Descripción del Servicio**: usado para describir la interfaz pública de un Servicio Web específico. El formato de interfaz Web Services Description Language - WSDL es típicamente usado para este propósito.

- **Descubrimiento de servicios**: centraliza servicios en un registro común tal que los servicios Web de la red puedan publicar su localización y descripción, y hace que sea fácil descubrir que servicios están disponibles en la red. Actualmente, la API UDDI se utiliza normalmente para el descubrimiento de servicios.

A través de que concepto la UDDI establece una distinción entre la abstracción y la implementación: establece una distinción similar entre la abstracción y la implementación con el concepto de tModels. La estructura tModel, abreviatura de "Technology Model" (modelo de tecnología), representa huellas digitales técnicas, interfaces y tipos abstractos de metadatos.

- El resultado de los tModels son las plantillas de enlace, que son la implementación concreta de uno o más tModels.
- Dentro de una plantilla de enlace se registra el punto de acceso de una implementación particular de un tModel.
- Del mismo modo que el esquema de WSDL permite separar la interfaz y la implementación, UDDI ofrece un mecanismo que permite publicar por separado los tModels de las plantillas de enlace que hacen referencia a ellos.

Por ejemplo, un grupo industrial o de estándares publica la interfaz canónica para un sector particular y, a continuación, varias empresas escriben implementaciones de esta interfaz.

Como es el proceso de publicación de la UDDI:

- Primero es especificar el tipo de negocio al que da soporte su servicio web. Normalmente, esto significa elegir un tipo de negocio existente en una lista, pero también se puede crear un nuevo tipo de negocio.
    - Para cada tipo de negocio hay una clave de empresa asociada. Los servicios web habilitados para bus de integración de servicios utilizan esta clave, en combinación con la clave de servicio, para encontrar el servicio web en el registro.
- Añadir un modelo técnico. Los modelos técnicos son categorías genéricas. Mediante estos modelos un usuario del registro UDDI puede buscar un tipo de servicio, en lugar de tener que conocer los detalles de acceso de un servicio específico.
    - Los servicios web habilitados para bus interactúan con registros UDDI en el nivel de servicios Web individuales y, por lo tanto, no utilizan modelos técnicos.
- Añadir el servicio web. El registro UDDI asigna una clave de servicio a su servicio, y lo publica.
    - Los servicios web habilitados para bus utilizan esta clave, en combinación con la clave de empresa, para encontrar el servicio web en el registro.

WDSL distinguir a los puertos, de los mensajes: diferencia a los puertos de los mensajes debido a que un documento WDSL se compone de un elemento raíz el cual a su vez se compone de diferentes elementos, entre los cuales existen dos elementos diferentes para ellos, uno para puertos y otro para los mensajes.

WSDL como define a los servicios: definen los servicios como colecciones de puntos finales de red o puertos. En WSDL, la definición abstracta de puntos finales y de mensajes se separa de la instalación concreta de red o de los enlaces del formato de datos. Esto permite la reutilización de definiciones abstractas: mensajes, que son descripciones abstractas de los datos que se están intercambiando y tipos de puertos, que son colecciones abstractas de operaciones. Las especificaciones concretas del protocolo y del formato de datos para un tipo de puerto determinado constituyen un enlace reutilizable. Un puerto se define por la asociación de una dirección de red y un enlace reutilizable; una colección de puertos define un servicio. Por esta razón, un documento WSDL utiliza los siguientes elementos en la definición de servicios de red:

- **Types**: contenedor de definiciones del tipo de datos que utiliza algún sistema de tipos (por ejemplo, XSD).
- **Message**: definición abstracta y escrita de los datos que se están comunicando.
- **Operation**: descripción abstracta de una acción admitida por el servicio.
- **Port Type:** conjunto abstracto de operaciones admitidas por uno o más puntos finales.
- **Binding**: especificación del protocolo y del formato de datos para un tipo de puerto determinado.
- **Port**: punto final único que se define como la combinación de un enlace y una dirección de red.
- **Service**: colección de puntos finales relacionados.

CUESTIONARIO 1 AGENTES DE SOFTWARE

1. Seleccione un modelo de inteligencia distribuida y defínala

Modelo de Blackboard: Este sistema permite la resolución de problemas complejos mediante la división de tareas a un conjunto de subsistemas

El sistema Blackboard se compone de tres elementos:

- **Las fuentes de conocimiento:** que son elementos independientes que contribuyen a la solución de un problema
- **La pizarra o blackboard:** es una fuente de información pública accesible para las fuentes de conocimiento, las cuales procesan y modifican la información a fin de resolver el problema
- **el sistema de control:** quien se encarga de activar de forma oportuna cada fuente de conocimiento, así como gestionar el flujo de información entre la pizarra y las fuentes de información.

2. Seleccione un modelo de inteligencia artificial paralela y defínala

Algoritmos de razonamiento paralelo: algoritmo que puede ser ejecutado por partes en el mismo instante de tiempo por varias unidades de procesamiento, para finalmente unir todas las partes y obtener el resultado correcto.

3. ¿Qué es una intención en el mundo de los agentes?

Objetivo Concreto: son las metas que el agente decide alcanzar, cuando el agente tiene varias intenciones, estas tienen que ser mutuamente consistentes.

4. ¿Cuáles son los aspectos salientes que diferencian a los agentes cognitivos de los reactivos?

Cognitivo:

- **Individualmente** inteligente
- Compuesto por un **pequeño** número de agentes cognitivos
- **Tienen conocimiento del pasado**
- Agente complejo que puede llegar a acuerdos con otros agentes.
- Capaz de efectuar operaciones complejas.

Reactivo:

- Son **globalmente** inteligentes
- Compuesto por **gran** número de agentes reactivos (lo cual implica tener en cuenta nuevas teorías de cooperación y comunicación que permitan el desempeño de estas acciones)
- **No tienen memoria**
- Bajo Nivel: no dispone de protocolo ni de lenguaje de comunicación
- Solo puede responder a estímulos

5. ¿Cuál es el impacto cultural de los agentes de Software?

Puede generar problemas de privacidad impredecibles si no cuenta con una comprensión completa del perfil del usuario.

6. Nombre 3 propiedades genéricas de los agentes

- Reactivo: responde a cambios en el ambiente
- Autónomo: ejerce control sobre sus propias acciones
- Orientado por objetivos: no actúa simplemente a respuestas
- Continuo: se ejecuta continuamente
- Comunicativo: se comunica
- Aprendizaje: cambia su comportamiento según su experiencia previa
- Movilidad: capaz de transportarse de máquina
- Flexible: sus acciones no responden a un libreto
- Carácter

7. Un agente tiene dos características destacables en el comportamiento. Nombre 2

- Funcionamiento continuo y autónomo
- Comunicación con el entorno y otros agentes por medio de un lenguaje o formalismo de comunicación
- Robustez
- Razonamiento y aprendizaje
- Movilidad

8. ¿Cómo se comunican los agentes entre sí?

Se comunican mediante una infraestructura que cuente con:

- Pilas de protocolos en los nodos
- Mecanismos de envíos y recepción
- Localización de los agentes
- Identificación y ciclo de vida

9. ¿Cuáles son los modelos de los agentes? Nombre dos.

- Agentes que dialogan en lenguaje natural
- Agentes que aprenden
- Agentes que razonan con un determinado formalismo lógico o difuso
- Agentes que buscan información
- Agentes que resuelven problemas

10. Nombre dos tipos de agentes

- Reactivos
- Cognitivos
- Híbridos

11. La lógica deóntica ¿Para qué me sirve?

Sirve para analizar formalmente las normas o proposiciones que tratan acerca de las normas

12. Identifique a los agentes de software según su movilidad

Fijos y móviles.

13. Nombre un mecanismo para resolver los conflictos

Agentes especialistas: detectan conflictos, comprueban su existencia y los resuelven enviando directrices a los agentes adecuados. La resolución de conflictos forma parte del rol del agente.

Negociación: los conflictos se resuelven intercambiando información y es necesario relajar las exigencias en la resolución de los objetivos.

14. ¿Qué cosas debo identificar para comprender a un sistema basado en agentes?

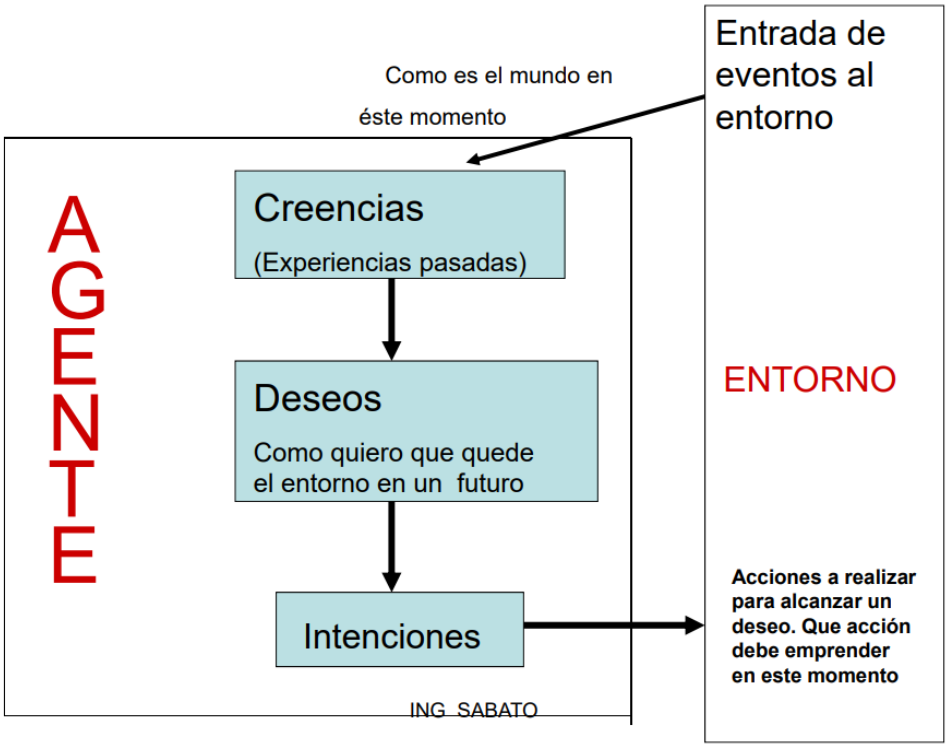
- Agentes que lo forman
- Comportamiento externo
- Relaciones con otros agentes
- Entorno computacional
- Finalidad

15. ¿Cuáles son las arquitecturas involucradas en la construcción de los agentes?

- Deliberativas:
  - Arquitectura BDI (Believe Desires Intentions)
  - Arquitectura Abstracta
- Reactivas:
  - Agentes reactivos en robótica
- Híbridas
  - Arquitecturas de capas
  - Sistemas Inteligentes Adaptativos

16. Grafique la arquitectura BDI (Descripción)

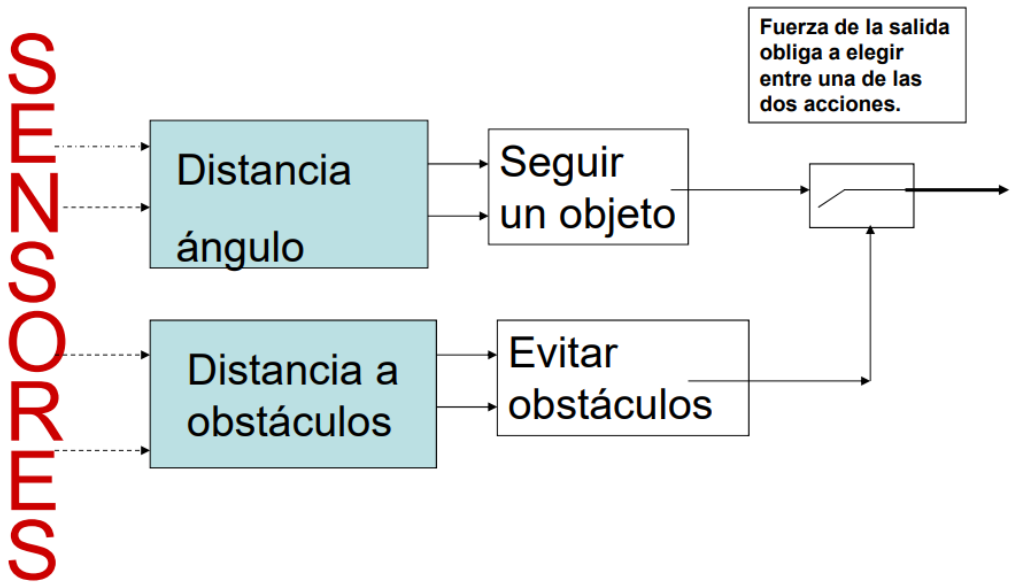
Está caracterizada por el hecho de que los agentes que la implementan están dotados de los estados mentales de Creencias, Deseos e Intenciones



17. Grafique la arquitectura reactiva o de subsunción (Descripción)

Se caracteriza por no tener como elemento central de razonamiento un modelo simbólico y por no usar razonamiento simbólico complejo.

Estas arquitecturas, también conocidas como de subsunción manejan jerarquías de tareas que definen un comportamiento. Se las organiza en jerarquías de capas, de menor a mayor nivel de abstracción.



18. ¿En qué son similares las arquitecturas?

En ambos casos es necesario identificar un conjunto de acciones o habilidades del agente que permitan al agente real (robot) interaccionar con el entorno en el que se encuentra inmerso.

El conjunto de habilidades necesita de la existencia de distintos niveles jerárquicos en las estructuras de control. Se distinguen dos niveles de control: El nivel más alto realiza el razonamiento a largo plazo relacionado con tareas complejas y el nivel más bajo son las que se relacionan con tareas sencillas con razonamiento a corto plazo

19. ¿Qué características definen el modelo o estándar FIPA (Foundation for Intelligent Physical-Agents)?

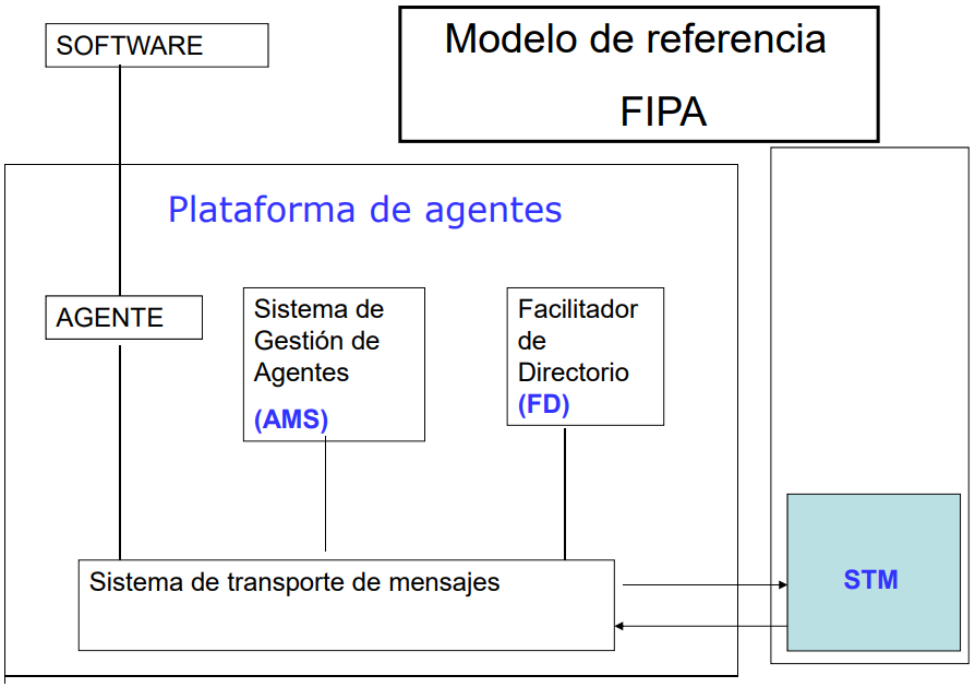
Sigue el principio de definir solo el comportamiento externo dejando las decisiones de diseño a cada equipo de desarrollo.

Define los servicios que debe proporcionar toda la plataforma de agentes:

- sistema de transporte de mensajes
- sistema de gestión de agentes
- servicio de directorio
- canal de comunicaciones para los agentes

Sirve para comercio electrónico

20. Grafique el modelo FIPA (Descripción)*



La comunicación entre los servicios se hace mediante mensajes ACL (Agent Communication Language)

21. ¿Qué cosas componen el modelo FIPA?

- Sistemas de gestión de agentes: conoce en todo momento el estado de la plataforma y los agentes que pertenecen a ella
- Facilitador de directorio: contiene un servicio de **páginas amarillas** que permite buscar un agente por sus capacidades y no sólo por su nombre
- Sistema de transporte de mensajes: elemento encargado de gestionar el envío de mensajes entre agentes de una plataforma y entre agentes de distintas plataformas.

22. Diferencie el concepto de Ontología

Filosófico: el estudio del ser (existencia, realidad, categorías fundamentales y relaciones del ser en cuanto ser)
Agentes de software: conjunto de símbolos o términos junto con su correspondiente interpretación o significado.

23. Los diagramas AUML tienen elementos. ¿Cuáles son?

AUML: Lenguaje de representación de protocolos basado en los diagramas de secuencia UML.

- Roles de los agentes
- Líneas de vida: define el período de vida en que existe el agente. Dicha línea puede descomponerse en dos o más líneas de vida para mostrar paralelismo. Existe paralelismo de tipo AND, OR y OR - exclusivo.
- Hilos de interacción: muestran el período de tiempo en el que un agente está realizando una tarea como reacción a un mensaje entrante. Se representa como un rectángulo alargado que se dibuja sobre la línea de vida del agente.
- Mensajes: se representa como una flecha continua horizontal que parte de un hilo de interacción a otro de otro agente.

24. Nombre y describa un agente intermediario

Agente intermediario: ayudan a otros en la localización y su conexión con proveedores de servicios.

- Agente mediador: proporciona acceso a algún tipo de fuente de información
- Broker: realiza la función de interfaz entre los agentes que proporcionan servicios y los que usan esos servicios, haciendo de intermediario en sus transacciones. Todas las comunicaciones entre pares de agentes pasan a través del bróker. Su acción se basa en la colaboración y negociación con otros agentes.
- Matchmaker: devuelve una lista ordenada y valorada de agentes que proporcionan el servicio solicitado. Esa valoración indica el grado de adaptación al servicio solicitado.

25. Cuando se modela con agentes existen dos puntos de vista: Uno externo y uno interno. Nombre alguna característica saliente de ambos.

- Externo: Se identifican los agentes y sus interacciones. Su propósito es identificar una jerarquía de clases de agentes.
- Interno: Describe el comportamiento de cada uno de los agentes. El agente persigue la satisfacción de sus objetivos.

26. ¿Cómo hace un agente intermediario para coordinar la mediación?

Son capaces de entender y procesar de forma automática peticiones y descripciones de servicios. La coordinación de la mediación requiere:

- Capacidad de comunicación con los agentes
- Sistemas y usuarios implicados en el proceso de mediación conforme a protocolos, convenciones y políticas existentes.
- Cualquier interacción de coordinación entre cualquier tipo de agente se realiza mediante el uso estandarizado de algún lenguaje de comunicación entre agentes (como FIPA - ACL) protocolos de comunicación y políticas de comunicación.

27. ¿Cuántos tipos de agentes intermediarios conoce?

3. Broker, Matchmaker y mediador.

28. ¿Qué es lo que intenta WSDL (Web Service Description Language)?

WSDL intenta separar los servicios (definidos en términos abstractos) de los formatos de datos y protocolos concretos utilizados por la implementación y define asociaciones entre las descripciones abstractas y sus realizaciones específicas.

29. ¿Cuáles son las primitivas de transmisión?

- One-way (el punto final recibe un mensaje)
- Request-Response (el punto final recibe un mensaje y responde con otro)
- Notification (el punto final envía un mensaje).
- Enlaces (Binding) definen los detalles sobre protocolos y formato de datos concretos para los mensajes de un tipo de puerto particular.

30. ¿Qué es OWL-S?

Es un lenguaje de ontología definido sobre XML para describir servicios Web.

31. ¿Cuáles son las herramientas utilizadas para desarrollar sistemas de agentes inteligentes?

- Ingeniería de Software para estructurar el proceso de desarrollo
- Inteligencia Artificial para dotar a los programas de capacidad para tratar situaciones imprevistas y tomar decisiones
- Programación Concurrente y Distribuida para tratar la coordinación de tareas ejecutadas en diferentes máquinas y bajo diferentes tipos de planificación.

32. ¿Cuáles serían esas plataformas de desarrollo?

JADE y ZEUS.

33. ¿Cuántas clases de agentes de información existen?

Existen 4.

- De información cooperativos o no cooperativos
- De búsqueda
- De filtrado
- De monitorización

34. ¿Cuáles son las aplicaciones en la recuperación de información?

Situaciones que implican la participación en múltiples escenarios de manera simultánea en los que existe un alto grado de variabilidad y las decisiones se deben realizar con limitaciones de tiempo.

- Infraestructura comercial
- Bienes o servicios para comercializar
- Cualidades de procesos de transacción
- Grado de apertura del mercado

35. ¿En qué consiste el lenguaje OiL?

OIL: Este lenguaje para la descripción e intercambio de ontologías combina 3 disciplinas.

- **Representación basada en marcos:** OIL usa una orientación basada en marcos para definir los conceptos y los atributos de la ontología.
- **Lógicas descriptivas (DL):** con lógicas descriptivas se definen las propiedades que tiene que satisfacer un objeto para pertenecer a un concepto concreto. OIL hereda de las lógicas descriptivas su semántica formal y su eficiente modelo de razonamiento.
- **Lenguajes "web":** Con esto nos referimos a que podemos expresar la sintaxis de OIL mediante XML o RDF-Schema, de modo que una ontología realizada en OIL pueda ser transmitida por Internet

36. ¿Cuáles son los elementos de los que se compone una ontología? *

La ontología se compone de tres partes (o subontologías) cuyo elemento principal es la clase Service y ellas son:

- **ServiceProfile:** describe que es lo que hace el servicio (páginas amarillas). Describe las propiedades de un servicio necesarias para descubrimiento automático, como la funcionalidad que proporciona, sus entradas – salidas, precondiciones y efectos.
- **ServiceModel:** describe como se ejecuta el servicio. (modelo de procesos, flujo de control de datos involucrados en su uso, etc.). Está diseñado para facilitar la ejecución y monitorización automática de servicios.
- **ServiceGrounding:** especifica los detalles de cómo se accede al servicio. Típicamente se especifica un protocolo de comunicación, formato de los mensajes y detalles como el número de puerto para contactar un servicio.

37. Finalmente ¿Qué es una ontología? *

Especificación formal y explicita de una conceptualización compartida. (conceptualización como construcción de un modelo abstracto de algún dominio o fenómeno)

38. ¿Qué es el STM de la arquitectura FIPA?

Es el elemento encargado de gestionar el envío de mensajes entre agentes de una plataforma y entre agentes de distintas plataformas.

39. Establezca 3 consideraciones acerca de las ventajas al construir un sistema usando agentes de Software

- Servicios con mayor funcionalidad que los convencionales.
- Simplifican el uso ocultando la complejidad
- Menor costo de desarrollo

40. Tipos de agentes:

Reactivos Cognitivos, Compradores, Personales, De monitoreo y vigilancia, de minería de datos, híbridos, individualistas, cooperantes, agente-persona, agente-agente, que requieren y no requieren entorno especial.

CUESTIONARIO 2 DE AGENTES DE SOFTWARE.

41. ¿DE DONDE PROVIENE O CUAL ES EL ORIGEN DE LOS SISTEMAS MULTIAGENTES?

Estos sistemas evolucionar a partir de la inteligencia artificial distribuida (IAD), de la solución distribuida de problemas (SDP) y de la inteligencia artificial paralela (IAP).

42. EXPLIQUE IAD (Inteligencia Artificial Distribuida)

Campo de la IA dedicado al estudio de técnicas y conocimiento necesario para la coordinación y distribución del conocimiento y acciones en un entorno con múltiples agentes

43. ¿QUE COMPONENTES TIENE UN AGENTE COGNITIVO?

Funcionalidad, creencias, conocimiento, control y comunicaciones.

44. ¿QUE SE DERIVA DE UNA INTENCION?

Las acciones de un agente y la capacidad con la cual estos pueden usar conceptos para predecir y explicar el comportamiento de otros agentes.

45. ¿COMO RESUMIRIA EL COMPORTAMIENTO DE UN AGENTE?

Plan de actividades que se ejecutan de acuerdo con su experiencia y conocimiento.

46. ¿COMO SON LOS SISTEMAS EXPERTOS RESPECTO DE LOS DE AGENTES DE SOFTWARE?

- No están acoplados a su entorno
- No están diseñados para comportarse de forma reactiva o proactiva
- No tienen en cuenta la habilidad social.

47. ¿COMO SON LOS OBJETOS RESPECTO DE LOS AGENTES DE SOFTWARE?

- Los objetos son menos autónomos que los agentes
- Los objetos carecen de un comportamiento flexible, reactivo, proactivo y social
- Los agentes tienen al menos un hilo en ejecución, los objetos no.

48. ¿MUY CONCRETAMENTE CUALES SERAN LOS IMPACTOS ORGANIZACIONALES Y CULTURALES DE LOS AGENTES?

- Organizacional: Introduce la responsabilidad operacional y las búsquedas rápidas en internet, identificando las mejores ofertas
- Cultural: Puede generar problemas de privacidad impredecibles si no cuenta con una comprensión completa del perfil del usuario.

49. PARA HAAG EXISTEN 4 TIPOS DE AGENTES. ¿CUALES SON?

- Compradores
- Personales
- De monitoreo y vigilancia
- De minería de datos.

50. EXISTE UNA TIPOLOGIA DE AGENTES BASADA EN 3 CARACTERISTICAS. NOMBRELAS Y DESCRIBALAS.

- Colaborativos: cooperativos y autónomos
- De Interfaz: autónomos y de aprendizaje
- De aprendizaje y colaborativos: cooperativos y de aprendizaje
- Smart: los 3 anteriores en uno.

51. NOMBRE 3 APLICACIONES DE AGENTES DE SOFTWARE

- Cronogramas y planificación de fabricación
- Control de procesos
- Control de tráfico aéreo
- Ubicación de contenedores
- Gestión de transporte de electricidad
- Apps de medicina.

52. ¿DENTRO DE UN MODELO D ORGANIZACIÓN, COMO ES LA COMUNICACIÓN ENTRE AGENTES?

- QUE es lo que se pretende decir (semántica del emisor)
- QUIEN lo dice (el emisor)
- A QUIEN o quienes se lo dice (el receptor)
- DE QUE FORMA lo dice (protocolo)
- CUANDO lo dice
- CON QUE MEDIO efectúa la comunicación (paradigma)
- Que es lo que el RECEPTOR ENTIENDE (semántica del receptor)

53. DEMAZEU PROPUSO UNA CLASIFICACION CONOCIDA COMO AEIOU. DE QUE SE TRATA?.LA RESPUESTA DEBE SER AMPLIA PERO MUY CONCRETA

- A(agente)-caracteriza sus rasgos individuales: arquitectura, funcionamiento interno, complejidad, etc.
- E(entorno)-caracteriza los requisitos computacionales para que el agente funcione adecuadamente.
- I(interacción)-considera las capacidades de comunicación del agente con quien se comunica y la forma de comunicación (tipo de mensajes, contenido, protocolos, etc.)
- O(organización)-considera el papel del agente en el conjunto del sistema y el modelo de cooperación con otros agentes.
- U(utilidad)-permite clasificar a los agentes de acuerdo con su finalidad.

- Se usan 2 **criterios**
  - el dominio de la aplicación
  - el tipo de tarea que se realiza dentro de él

54. ¿CUALES SON LOS FORMALISMOS PARA LA REPRESENTACION DEL CONOCIMIENTO?

- Lógicos (de predicados, modal, temporal, deóntica)
- Computacionales (objetos, trames, procedimientos y ontologías combinados con reglas)

55. ¿COMO SE RESUELVEN LOS PARADIGMAS DE RESOLUCION DE PROBLEMAS?

Se resuelven mediante

- Resolución basada en objetivos
- Resolución dirigida por eventos o creencias
- Planificación.

56. COMO SE RESUELVEN LOS CONFLICTOS INTERNOS Y EXTERNOS

- Agentes especialistas: detectan conflictos, comprueban su existencia y los resuelven enviando directrices a los agentes adecuados.
- Negociación: los conflictos se resuelven intercambiando información y si es necesario relajando las exigencias en la resolución de los objetivos.

57. ¿QUE DEBO IDENTIFICAR PARA COMPRENDER MEJOR A LOS AGENTES?

- Los agentes que lo forman
- Su comportamiento externo
- Las relaciones con los otros agentes
- El entorno computacional
- Su finalidad

58. ¿QUE SON LA OMG Y LA ITEF? (MUY RESUMIDAMENTE)

- OMG (Object management group): Consorcio dedicado al cuidado y establecimiento de diversos estándares de tecnologías orientadas a objetos.

- ITEF (Internet Engineering Task Force): organización internacional abierta de normalización, que tiene como objetivos el contribuir a la ingeniería de Internet, actuando en diversas áreas, como transporte, enrutamiento y seguridad.

59. A ESTA ALTURA, REDEFINA EL CONCEPTO DE AGENTE

Un agente debería ser:

- autónomo
- sociable
- proactivo
- inteligente
- adaptable con una forma y un comportamiento bien definido.

Su funcionamiento depende del sistema operativo y de un conjunto de recursos electrónicos, computacionales cuyo nivel de abstracción abarca los elementos usuales de clases, librerías, objetos o componentes.

Al concepto de agente hay que asociarle un patrón arquitectónico especificando con claridad:

- la forma externa, con las interfaces que ofrece y las que utiliza.
- la estructura interna.
- el comportamiento: ciclo básico y comportamiento específico.
- el entorno computacional necesario para funcionar.

60. NOMBRE SIN DESCRIBIR COMO EL ESTANDAR FIPA DIVIDE LA COMUNICACIÓN ENTRE AGENTES

- Actos de comunicación
- Protocolos de interacción
- Lenguajes de contenido

61. DEFINA ARQUITECTURA DE AGENTES Y MUY CONCRETAMENTE, DE QUE SE TRATA.

La arquitectura determina los mecanismos que usa un agente para reaccionar a los estímulos. Uno de los aspectos básicos que diferencia una arquitectura de otra es el método de descomposición del trabajo en tareas particulares, es decir una planificación ya que un agente no es más que un sistema que ejecuta acciones en un entorno determinado.

62. ¿EN QUE SE BASA UN MODELO SIMBOLICO? ¿QUE ES Y COMO SE CONSTRUYE?

Un modelo simbólico está representado explícitamente y es en donde las decisiones se toman usando mecanismos de razonamiento lógico

Los modelos simbólicos se construyen mediante reglas abstractas para representar un objeto real mediante una codificación matemática, geométrica y estadística

63. ¿CUAL ES EL MODELO TEORICO DE LA ARQUITECTURA BDI?

Actitudes -basado en lógica modal (permite razonar sobre lo que podría ser o se cree que es, en lugar de lo que realmente es).

Modelo - “mundos posibles”: se maneja una semántica de mundos posibles con una relación de accesibilidad entre ellos.

La relación de accesibilidad enlaza la situación actual (el mundo actual) con todas las que son posibles a partir de ella. De este modo los conceptos de verdad necesaria y verdad posible se definen a partir de esta relación de accesibilidad

Los agentes BDI se pueden modelar usando una estructura, basada en la lógica de los mundos posibles.

64. ¿A GRANDES RASGOS COMO ES UNA ARQUITECTURA REACTIVA?

-No tiene como elemento central de razonamiento un modelo simbólico.

-No usa razonamiento simbólico complejo

-Manejan jerarquías de tareas que definen un comportamiento.

65. EXPLIQUE SIMILITUDES ENTRE ARQUITECTURAS.

En la deliberativa reactiva, en ambos casos es necesario identificar un conjunto de acciones o habilidades del agente que permitan al agente real (robot) interaccionar con el entorno en el que se encuentra inmerso.

Se distinguen dos niveles de control: El nivel más alto realiza el razonamiento a largo plazo relacionado con tareas complejas y el nivel más bajo son las que se relacionan con tareas sencillas con razonamiento a corto plazo

66. TODA NUEVA TECNOLOGIA PUEDE TENER ALGUNOS PROBLEMAS. ¿CUALES SON CON RELACION A LA ARQUITECTURA FIPA?

- Interoperabilidad
- Escalabilidad

67. ¿QUE DEFINE EL ESTANDAR FIPA?

Define los servicios que debe proporcionar toda la plataforma de agentes:

- sistema de transporte de mensajes
- sistema de gestión de agentes
- servicio de directorio
- canal de comunicaciones para los agentes

68. ¿CUALES SON LOS COMPONENTES DE UNA PLATAFORMA FIPA?

- Sistemas de gestión de agentes
- Facilitador de directorio
- Sistema de transporte de mensajes

69. ¿COMO ES EL SERVICIO DE ONTOLOGIA EN FIPA?

Toda la plataforma FIPA se basa en la comunicación entre agentes mediante ACL (Agent Communication Language). Cuando dos agentes están conversando es necesario que ambos estén de acuerdo en el lenguaje y la ontología empleados.

Labores básicas del servicio:

- mantener un conjunto de ontologías de uso público accesible a los agentes
- traducir expresiones entre diferentes ontologías
- responder a consultas sobre términos de las ontologías que gestiona.
- facilitar la identificación y uso de ontologías compartidas entre los agentes
- descubrir nuevas ontologías y ponerla a disposición de todos los agentes.

70. ¿LOS PROTOCOLOS DE COMUNICACIONES FIPA, COMO SON Y EN QUE SE BASAN?

En la especificación FIPA encontramos ya definidos una serie de protocolos muy comunes basados en agentes (FIPA IPL, Interaction Protocol Library).

Para la definición de estos protocolos estándar FIPA ha definido un lenguaje de representación de protocolos llamado AUML, basado en los diagramas de secuencia de UML a los que denomina diagramas de protocolo

71. ¿QUE ES LA COMPARTICION DE CONOCIMIENTO Y CUALES SON LOS PROBLEMAS QUE SOLUCIONA?

Es un concepto que surge para que el contenido de los mensajes intercambiados sea comprendido por los interlocutores que así lo requieran. Surge del problema que se da cuando dos agentes están comunicándose con el mismo lenguaje, pero no comparten el mismo significado para los conceptos que se manejan en el texto del mensaje.

72. ¿CUALES SON LOS ELEMENTOS QUE COMPONEN UNA ONTOLOGIA?

- Conceptos: cualquier entidad sobre la cual se pueda decir algo: árbol, libro, etc. –
- Relaciones: representan una interacción entre conceptos del dominio. Parte-de, subclase-de.
- Funciones: son un caso especial de relaciones, en el que el elemento n-esimo de la relación es único para los n-1 restantes. Por ej., madre-de, precio-de
- Axiomas: son sentencias siempre verdaderas para dicho dominio. Por ej. “el lunes va después del domingo”.
- Instancias: son conceptos concretos.

73. NOMBRE SIN DESCRIBIR LENGUAJES DE REPRESENTACION MAS SIGNIFICATIVOS.

KIF, OCML, FLogic, KM, lenguajes basados en lógica formal y lenguajes basados en frames.

74. ¿CUAL FUE LA DISCIPLINA QUE DIO ORIGEN A LOS MODELOS DE COORDINACION?

Fue la teoría de la organización.

75. ¿QUE ES UN AGENTE INTERMEDIARIO Y COMO SE COORDINAN LOS SERVICIOS DE MEDIACION?

Son aquellos que ayudan a otros en la localización y su conexión con proveedores de servicios.

La coordinación de la mediación requiere:

- Capacidad de comunicación con los agentes
- Sistemas y usuarios implicados en el proceso de mediación conforme a protocolos, convenciones y políticas existentes.

También debe garantizar

- la privacidad de los datos
- el anonimato y la verificación de las supuestas capacidades de los agentes registrados (uso de certificados de seguridad, intercambio de matrices de confianza, reputación)

76. ¿COMO ES EL MODELO DE AGENTES? SEA CONCRETO

Es una metodología que considera:

- Un punto de vista externo, en el cual se identifican agentes y sus interacciones
- Un punto de vista interno, que describe el comportamiento de cada uno de los agentes.

77. ¿CON QUE LENGUAJES SE IMPLEMENTAN LOS AGENTES?

La mayoría se los implementa en Java

78. ¿COMO ES LA APLICACIÓN DE AGENTES EN LA GESTION DE TRANSPORTE ELECTRICO?

Existe una topología determinada que ante cambios intempestivos en los valores de potencia activa, reactiva, tensión y corriente pueden generar situaciones de alarma.

Un sistema razonable debería
-Detectar la existencia de fallas
-Determinar la causa, localización y tipo de fallos, incluyendo la identificación de algún equipo dañado
-Analizar la situación de la red una vez que llegue a un estado estable
-Preparar un plan de restauración para devolver la red a su estado original de operación

79. NOMBRE VENTAJAS CONCRETAS PARA DESARROLLAR MEDIANTE AGENTES

- Servicios con mayor funcionalidad que los convencionales.
- Simplifican el uso ocultando la complejidad
- Menor costo de desarrollo
- Facilitan reusabilidad
- Integración e interoperabilidad con otras tecnologías y sistemas
- Versatilidad y capacidad de integración con otras tecnologías.
- Evolución y mantenimiento

CUESTIONARIO DE MOBILE

80. ¿CUALES SON LOS PROTOCOLOS DE MOBILE?

WAP (Wireless Application Protocol) es un estándar abierto internacional para aplicaciones que utilizan las comunicaciones inalámbricas. A través de él se realiza la comunicación. Bajo este término se resumen varios protocolos de Internet y propiedades técnicas que adaptar el contenido web para pequeñas pantallas de teléfonos móviles, teniendo en cuenta las conexiones de Internet móvil más lentas.

81. CLASIFIQUE CONCRETAMENTE DISPOSITIVOS MOVILES

- PDAs (Personal Digital Assistants): caben en una mano y tienen algunas características en común:
  - Pantalla grande en proporción al tamaño del equipo
  - Soporte de pantalla touch-screen
  - Carecen de teclado y el ingreso de texto es a través del stylus (lápiz que hace lo que el ratón)
  - Permiten comunicarse con un teléfono celular y acceder a internet
  - Poseen un S.O con soft utilitario y es posible instalarles aplicaciones adicionales.
- PocketPc: ofrece un S.O Windows reducido.
- Handled PC: son equipos tipo agenda electrónica que se abren y poseen pantalla y teclado
- SPOTs (Small Personal Object Technology)
- Tarjetas Inteligentes: SIM Card de teléfonos celulares GSM con tecnología Java Card
- Teléfonos Celulares

82. ¿QUE ES UNA SOLUCION STAND-ALONE?

Se desarrollan para ser instaladas y ejecutadas sobre el equipo móvil y que funcionan en forma desconectada de Internet o servidor central. Para ello se debe generar un paquete ejecutable en el formato correcto para el tipo de S.O sobre el cual será instalado, así como también, por la versión de este.

83. ¿QUE ES UNA SOLUCION ONLINE(CONECTADA)?

Es una solución a través de internet usando páginas WEB o WAP para la interfaz de esta en el equipo y toda la ejecución se realiza en el servidor.

84. ESTABLEZCA LA DIFERENCIA ENTRE STAND-ALONE Y ONLINE; VENTAJAS Y DESVENTAJAS.

En Stand-Alone se crean soluciones para que funcionen de forma desconectada. En Online no es necesaria la instalación de solución y se pueden hacer operaciones complejas.

Stand-alone:

- **Ventajas:**
  - Ejecución veloz
  - Manejo de memoria: se aprovecha las características de bajo nivel del equipo
  - Se puede trabajar sin necesidad de estar conectado
  - Soporte de sincronización con un equipo de escritorio
- **Desventajas:**
  - Se debe desarrollar diferentes versiones para cada S.O
  - No pueden consultar con centros de datos remotos
  - No pueden soportar grandes cantidades de información para búsqueda o almacén

Online:

- **Ventajas:**
  - No es necesario distribuir ni instalar ninguna aplicación
  - Se pueden hacer operaciones complejas ya que la ejecución se hace en el servidor
  - Se puede trabajar con gran cantidad de información
- **Desventajas:**
  - No se puede acceder a capacidades de bajo nivel del equipo
  - Se necesita estar conectado para poder usarlo
  - No se pueden usar todos los controles de ingreso disponibles, solo los propuestos por el lenguaje en cuestión
  - Ejecución más lenta (se debe recargar la información contra el servidor)

85. ¿QUÉ ES UNA SOLUCION SMART CLIENT?

Una solución SmartClient junta las soluciones stand-alone con las online. Consta de aplicaciones ejecutables que se distribuyen e instalan en los equipos, pero también usan la conexión para comunicarse e interactuar con un servidor. La diferencia, es que la aplicación debe ser capaz de seguir ejecutándose aun cuando el equipo pierda conexión al servidor.

Ventajas:

- Reúne lo mejor del mundo conectado y desconectado
- Permite consultar grandes capacidades de información y hacer uso de funciones de bajo nivel de los equipos
- Permite seguir trabajando cuando el equipo se desconecta

Desventajas:

- Es de desarrollo más complejo
- Se debe crear el cliente basándose en cada tipo y versión del S.O
- Se debe distribuir e instalar el cliente en todos los equipos

86. CUANDO SE TRABAJA CON APLICACIONES QUE SE EJECUTAN DIRECTAMENTE SOBRE EL EQUIPO DISTINGUIMOS 2 TIPOS DE CODIGO PARA GENERAR DOS TIPOS DE CODIGO; CODIGO NATIVO Y CODIGO MANEJADO. DESCRIBA Y COMPARE VENTAJAS Y DDESVENTAJAS DE LOS DOS ANTERIORES.

Código Nativo: Implica que el archivo ejecutable que instalemos en el equipo está expresado con código ensamblador entendible por el S.O y por el procesador del equipo.

Ventajas:

- Mayor rapidez de ejecución
- Código más compacto
- Acceso al 100% de las capacidades del equipo
- No requiere la instalación de ningún agregado para la ejecución
- Se posee acceso directo a memoria y de bajo nivel.

Desventajas:

- En cada S.O o hardware se necesita recompilar el proyecto y generar ejecutables distintos
- Si trabajamos con 2 equipos distintos, debemos tener en cuenta diferencias de hardware en algunas funciones
- Se posee acceso directo a memoria (puede traer problemas)

Código Manejado: código que es entendible para un aplicativo intermedio entre nuestro programa y el hardware (máquina virtual). La máquina virtual interpreta el código manejado y lo convierte en tiempo real (just in time) a código nativo subordinado al sistema y hardware en que se encuentra.

Ventajas:

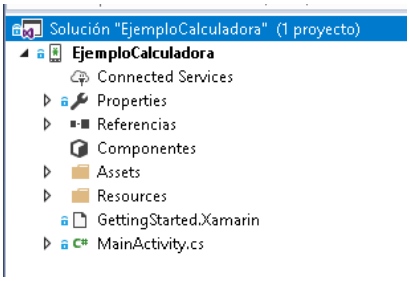
- Con un solo proyecto y compilación podremos ejecutar nuestra aplicación en diversos sistemas hardware
- No se posee acceso directo a memoria (ya que la VM lo administra automáticamente)
- Se genera un solo paquete de instalación para todos los equipos.

Desventajas:

- No se accede al 100% de los recursos del equipo, solo a los que se definición como parte del estándar de la VM
- No se tiene acceso de bajo nivel a recursos o a memoria

87. ESCRIBA LA SECUENCIA PARA CREAR UNA CALCULADORA EN ALGUN EMULADOR DE MOBILE EN VISUAL STUDIO 2017.
Se crea un proyecto en Visual Studio, seleccionando una Aplicación en Blanco (Android)

Al crearse la aplicación nos aparecerá una estructura de archivos (lo que determina si es el correcto es la aparición de la carpeta Layout dentro de Resources)



Creamos un formulario que representará en este caso la calculadora, insertando un archivo XML, esto permite arrastrar controles que queramos usar en la interfaz.

Luego le añadiremos funcionalidad a la interfaz gráfica instanciando los controles que deseamos utilizar, creando sus eventos correspondientes para poder utilizarlos (diferencia con C# o ASP.Net).

```
        SetContentView(Resource.Layout.Main);

        //btn1 click, suma
        Button b1 = FindViewById<Button>(Resource.Id.button1);
        b1.Click += B1_Click;
    }

    //Handles click
    private void B1_Click(object sender, System.EventArgs e)
    {
        EditText input1 = FindViewById<EditText>(Resource.Id.editText1);
        EditText input2 = FindViewById<EditText>(Resource.Id.editText2);
        TextView r = FindViewById<TextView>(Resource.Id.textView3);

        r.Text = (int.Parse(input1.Text) + int.Parse(input2.Text)).ToString();
    }
}
```

Para comprobar que la aplicación realice correctamente lo que necesitamos, se programan la funcionalidad y luego se ejecuta el proyecto a través de una emulación que proporciona Xamarin o también se permite ejecutar en un Android a través de conexión USB.

88. ¿CUALES SON LOS ENTORNOS DE DESARROLLO MAS POPULARES PARA CREAR DISPOSITIVOS MOVILES?
Entornos de desarrollos populares: Native Script, React Native, Ionic, Xamarin, Kotlin, Flutter.

89. ¿ENCUENTRA ALGUNA VINCULACION ENTRE AZURE Y DISPOSITIVOS MOVILES?
En Azure permite crear y hospedar un servidor back-end para cualquier aplicación móvil utilizando back-end móvil como servicio (MBaaS), el cual permite:

- Compilación única para usuarios de cualquier plataforma: escribiendo el código en un único lenguaje y seleccionando cualquier dispositivo de destino, ya sea Android, iOS o Windows
- Crear con rapidez aplicaciones móviles capaces de interactuar
- Utilizar la sincronización de datos sin conexión para crear aplicaciones con alta capacidad de respuesta
- Conectar aplicaciones con datos locales

CUESTIONARIO PROGRAMACION PARALELA

90. ¿EN QUE CIRCUNSTANCIAS SE APLICA PROGRAMACION PARALELA, CUAL ES LA RAZON DE SU APLICACIÓN O NECESIDAD? CUANDO APLICAR PROGRAMACION PARALELA

La primera respuesta es la limitación física de la computación. Necesitamos concurrencia.

En cálculos de gran volumen y de seria importancia. Existe el paradigma de memoria compartida y el paradigma de pasos de mensaje. La PP es necesaria por una limitación física de la computación y por el proceso de integración y miniaturización de componentes electrónicos.

91. ¿CUALES SON LOS PASOS DE LA METODOLOGIA PARA APLICAR PROGRAMACION PARALELA? ¿PORQUE ES MUCHO MAS COMPLEJA QUE LA PROGRAMACION SECUENCIAL?

Asignación y descomposición.

Es una tarea mucho más compleja que en el ámbito secuencial ya que presenta una serie de aspectos añadidos entre los que se destacan:

- Concurrencia
- Asignación de datos y códigos a los procesadores
- Acceso simultáneo a datos compartidos
- Escalabilidad

92. ¿COMO SE HACE PARA DESCOMPONER UN CALCULO?

En programación paralela la descomposición de un cálculo consiste en trocear las partes de menor tamaño con el objetivo de que muchas de esas partes se pueden realizar de forma independiente entre sí

93. NOMBRE ALGUNA TECNICA DE DESCOMPOSICION

- De dominio
- Funcional
- Recursiva
- Exploratoria
- Especulativa
- enfoques mixtos.

94. NOMBRE DOS PROBLEMAS COMPLEJOS Y DOS MODELOS MATEMATICOS COMPLEJOS

Problemas complejos: con masiva cantidad de datos, resolución de algoritmos en tiempo real

Modelos matemáticos: Estudios de modelos de la biosfera y meteorología, estudio del genoma humano.

95. ¿QUE IMPLICANCIAS TIENE LA PROGRAMACION PARALELA?

Primero, ante un problema es necesario establecer cuáles son las partes independientes en las que se puede dividir este problema, es decir establecer un conjunto de tares independientes que pueden ejecutarse de forma simultánea o sea en paralelo.

Segundo, es necesario especificar en que recursos computacionales van a ejecutarse las tareas independientes generadas, es decir, se deben asignar tareas a procesadores.

Por último, las estructuras de datos que se van a usar deben garantizar la disponibilidad de los datos a aquellos procesadores que van a ejecutar los procesos que usan dichos datos.

Además, debe existir una gestión completa de los procesos que garantice la actualización de los datos, que controle el orden de ejecución de los procesos que no son independientes, que reasigne procesos o redistribuya datos si esto fuera necesario

96. ¿CUALES SON LOS NIVELES DE PARALELISMO?

- Hardware
- Software.

97. ¿CUALES SON LOS TIPOS DE PARALELISMO?

- Explicito
- Implícito.

98. MEMORIA DISTRIBUIDA. EXPLIQUE FUNCIONAMIENTO, VENTAJAS Y DESVENTAJAS.

Funcionamiento: Existe algún tipo de herramienta lógica que se encarga de difundir el código entre los distintos procesadores previamente por lo que al inicio de un determinado programa el código se encuentra en memoria local de cada uno de los procesadores. Los datos se encuentran en la memoria local de alguno de ellos. Todos los procesadores ejecutan el mismo código, pero este puede contener partes que dependan del índice del procesador por lo que el código puede particularizarse de forma tal que cada procesador ejecute un conjunto diferente de instrucciones o ejecute las mismas instrucciones sobre diferentes datos.

Ventajas:

- no existen conflictos de acceso a memoria o son muy reducidos
- al admitir un número muy elevado de procesadores permite un paralelismo masivo
- el diseño y la fabricación suele ser más asequible, son escalables
- son adecuados para su integración mediante técnicas VLSI, incorporándose procesador, memoria y enlaces en un solo chip.
- permiten implementaciones sencillas construidas a partir de computadoras personales a un costo no excesivo.

Desventajas:

- su programación es más compleja
- es difícil concebir algoritmos paralelos eficientes.
- los mensajes (que son difícil de evitar) proporcionan un tiempo adicional de ejecución de los programas
- es difícil equilibrar la carga computacional entre los distintos procesadores.
- no se aprovecha totalmente la memoria. A pesar del incremento de memoria asociado con un mayor número de procesadores, debe replicarse el núcleo del sistema operativo y el código de los programas en cada procesador.

99. CUALES SON LAS CARACTERISTICAS QUE DISTINGUEN A LOS MULTICOMPUTADORES (M. DISTRIB)

- La memoria es privada (cada procesador tiene un mapa de direcciones propio que no es accesible directamente a los demás)
- La comunicación entre procesadores es por paso de mensajes a través de una red de interconexión
- Cada nodo es una computadora clásica -los nodos colaboran para resolver juntos un mismo problema (ejecutan la misma aplicación)
- La compartición de datos es explícita, ya que el acceso a datos comunes es por paso de mensajes

100. EXPLIQUE LA ARQUITECTURA DE SISTEMAS DISTRIBUIDOS

No comparten una memoria común y los procesadores son autónomos.

101. ¿LA PROGRAMACION EN MEMORIA COMPARTIDA (OpenMP), COMO ES?

- se basa en directivas

- el paralelismo se especifica a través de directivas que se insertan en el código C/C++ o Fortran.

102. NOMBRE Y EXPLIQUE 3 METRICAS DE PROGRAMACION PARALELA

- Ley de AMDAHL: Mide el límite en la ganancia de velocidad obtenible por la existencia de una parte del problema que es intrínsecamente secuencial.
- SPEEDUP: Es la ganancia de velocidad que se consigue con un algoritmo paralelo al resolver un problema con respecto a un algoritmo secuencial para el mismo problema.
- Eficiencia: porción de tiempo que los procesadores se dedican a trabajo útil, es speedup dividido cantidad de procesadores.
- Coste: tiempo o trabajo realizado por todo el sistema para la resolución del problema.

103. Arquitectura:

a. OpenMP: memoria compartida

b. MPI: memoria distribuida

104. CARRITO EN XML

```
public partial class CarritoXML : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        if(!IsPostBack)
        {
            XmlDocument xmlDoc = new XmlDocument();
            XmlTextReader xmlReader = new XmlTextReader(Server.MapPath("Catalogo.xml"));
            xmlReader.WhitespaceHandling = WhitespaceHandling.None;
            xmlDoc.Load(xmlReader);
            Session.Add("CatalogoSesion", xmlDoc);

            xmlReader.Close();
        }
    }

    protected void btnAgregar_Click(object sender, EventArgs e)
    {
        int n;
        int cantidad = 1;

        XmlDocument xmldocumento = (XmlDocument)Session["CatalogoSesion"];
        n = 1;
        cantidad = int.Parse(txtCantidad.Text);
        double precio = double.Parse(xmldocumento.DocumentElement.ChildNodes[n].ChildNodes[2].InnerText);


    }
}
```

El XML seria de la siguiente forma:

INVENTARIO2.XML

```
<?xml version="1.0" encoding="utf-8" ?>
<Inventario>
  <Producto Color="Rojo" SuministradorId="2">
    <ProductoId>1</ProductoId>
    <ProductoNombre>Manzanas</ProductoNombre>
    <ProductoPrecio>4,23</ProductoPrecio>
  </Producto>
  <Producto Color="Verde" SuministradorId="6">
    <ProductoId>2</ProductoId>
    <ProductoNombre>Peras</ProductoNombre>
    <ProductoPrecio>3,12</ProductoPrecio>
  </Producto>
</Inventario>
```

105. BANCO EN XML

```
using System.Xml.Xsl;
public partial class _Default : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        if (IsPostBack == false)
        {
        XPathDocument xpathdocum = new XPathDocument(Server.MapPath("banco.xml"));
            XPathNavigator xnavegador;
            XPathNodeIterator iterador;
            xnavegador = xpathdocum.CreateNavigator();
            iterador = xnavegador.Select("banco/cuenta/@cuentanum");
            while (iterador.MoveNext())
            {
                DropDownList1.Items.Add(iterador.Current.Value);
            }
            iterador = xnavegador.Select("banco/cuenta/cuentanombre");
            while (iterador.MoveNext())
            {
                DropDownList2.Items.Add(iterador.Current.Value);
            }
        }
    }
    protected void Button1_Click(object sender, EventArgs e)
    {
        string cuentan = DropDownList1.SelectedItem.Text.Trim();
        string cuentanob = DropDownList2.SelectedItem.Text.Trim();
        XPathDocument xpathdocum = new XPathDocument(Server.MapPath("banco.xml"));
        XPathNavigator xnavegador;
        XPathNodeIterator iterador;
        xnavegador = xpathdocum.CreateNavigator();
        iterador = xnavegador.Select("banco/cuenta[@cuentanum= '" + cuentan + "']/cuentasaldo");
        iterador.MoveNext();
        TextBox1.Text = String.Format("{0:C}", iterador.Current.Value);
        iterador = xnavegador.Select("banco/cuenta[cuentanombre='" + cuentanob + "']/cuentasaldo");
        iterador.MoveNext();
        TextBox2.Text = string.Format("{0:C}", iterador.Current.Value);
    }
}
```

El XML seria de la siguiente forma:

```
<?xml version="1.0" encoding="utf-8" ?>
<Banco>
  <Cuenta>
    <CuentaNum>CU123</CuentaNum>
    <CuentaNombre>Miguel Hernández</CuentaNombre>
    <CuentaSaldo>5623,12</CuentaSaldo>
    <CuentaSucursal>Badajoz</CuentaSucursal>
  </Cuenta>
  <Cuenta>
    <CuentaNum>CA056</CuentaNum>
    <CuentaNombre>Neus Espinal</CuentaNombre>
    <CuentaSaldo>12623,12</CuentaSaldo>
    <CuentaSucursal>Barcelona</CuentaSucursal>
  </Cuenta>
  <Cuenta>
    <CuentaNum>CH009</CuentaNum>
    <CuentaNombre>Esteban Ruíz</CuentaNombre>
    <CuentaSaldo>623,12</CuentaSaldo>
    <CuentaSucursal>Burgos</CuentaSucursal>
  </Cuenta>
  <Cuenta>
    <CuentaNum>CX555</CuentaNum>
    <CuentaNombre>Antonia Pérez</CuentaNombre>
    <CuentaSaldo>6023,12</CuentaSaldo>
    <CuentaSucursal>Barcelona</CuentaSucursal>
  </Cuenta>
</Banco>
```

CLASE COMPLEJIDAD

106. Estado del arte actual de la complejidad

Década del 90: antes un lenguaje se lo estructuraba y se calculaba el exponente, y si daba mayor que 9 ese tramo de código era complejo, sino era más simple y fácil de mantener.

Cálculo de mackein

Hoy en día: todo cambio porque los sistemas de computación cambiaron, especialmente por la nube. Hoy se manejan grandes cantidades de datos (imágenes video texto) que van a un reservorio, que además se nutre del IoT de suerte tal que los datos de los sensores RFC RFID y NFC van "arriba" para ser administrados. Bases de datos SQL y No-SQL, manejo de datos por HADOOP, que no solamente maneja grandes datos, sino que además algoritmos que levantan el sistema de mantenimiento ante una caída, además de tener un concepto de replicación.

107. PaaS, IaaS, SaaS. Que privilegios tiene el usuario frente a la nube:

- **Sistema más complejo**
  - Sistemas distribuidos cliente servidor donde el servidor en una red puede ser cliente en otra.
- **Este volumen se maneja por medio de la concurrencia, programación paralela y distribuida y concurrencia. Sutil diferencia entre todas**
  - Si hay que confiar en la concurrencia que sincrónica o asincrónicamente esta todo manejado por un reloj global (norma NTS norteamericanas), guiado por relojes atómicos (Gracias a él tenemos los GPS), ni hablar que los tiempos que transcurren en los satélites es distintos a los de la tierra, los atómicos tienen en cuenta estas diferencias.

108. LAS 3 V: VELOCIDAD, VERACIDAD, VOLUMEN.

109. Cuando hablamos de complejidad, hay varios frentes para abordarlo:

- Teoría de complejidad
- Teoría de la información
- Teoría de números (aritmética superior)

110. Con que lenguaje particular se tiene en cuenta el estudio de complejidad computacional:

- **Por LISP**
  - LISP processing
  - Es funcional, no imperativo
  - De matemática pura
  - Los datos son átomos y listas
    - **Átomos:** son elementos individuales
    - **Listas:** Conjunto de átomos
  - Lambda es lo que voy a computar y no como computarlo
    - Es como resolver algo con la máquina de Turing.
  - Es un lenguaje que es una parte compilada y otra interpretada (lenguaje intermedio)
  - Trabaja con notación polaca y expresiones lambda.
    - Es como resolver algo con la máquina de Turing.
  - No hay que tipificar.
  - Evaluación perezosa, otros lenguajes son "impacientes" (c++, c#).
- **Otro lenguaje: Python**
  - Es lindo, pero no es completo, no tiene las notaciones de LISP
  - Totalmente interpretado, pero al no trabajar expresiones lambda se lo utiliza muy bien para inteligencia artificial, pero no para cosas para la complejidad.
  - Hay que tipificar
  - Lo que importa es QUE voy a computar, no COMO.

111. Problema de la parada y la incompatibilidad de los números reales para máquinas de Turing

- La cantidad de números reales en la recta real es infinita, y la cantidad de programas en el mundo también, pero el infinito de los números reales es mayor y lo hace incompatible. No se puede calcular
- Es similar al problema de la parada de las expresiones binomiales.
- En base a esto infiere la incompletitud a partir de la incompatibilidad (contraparte de Leibniz).
  - Chaitin dice que la incompletitud es a partir de lo aleatorio
- las ecuaciones diofánticas se comportan igual que una computadora (de forma abstracta, nos olvidamos de lo que conocemos como computadora).
- Newton y Leibniz se disputaban en el 1600 sobre quien fue el primero en lograr el teorema para unir las derivadas con las integrales
- Leibniz estaba muy orgulloso sobre descubrir la cuadratura del círculo, y lo represento como una serie infinita divergente, que según el representaba la existencia de dios.

    - Se le ocurrió sacudir un pincel sobre un lienzo blanco. Obviamente se llenó de tinta, pero la dispersión era aleatoria. Sin embargo, el encontró un patrón en esa aleatoriedad.
    - Fue el primero que pudo inferir la complejidad a partir de la aleatoriedad
- Hilbert se pone a desafiar a sus amigos sobre los 20 problemas con la pretensión de axiomatizar toda la matemática del universo. Fracasa porque aparece Gödel con la teoría de incompletitud (3 tomos).

112. ¿Si la matemática está basada todo en axiomas, como los podés demostrar?

Todo lo que está por encima de esos axiomas no se pueden demostrar y por lo tanto la matemática está incompleta

- De todas maneras, pudo crear el sistema axiomático formal (SAF)
- Turing intenta ver cómo utilizarlo
- Otros matemáticos llegan a descubrir la utilidad de un SAF.
    - Estos matemáticos lo que quieres es trabajar con ecuaciones diofánticas (x^n +y^n = z^n)
    - Fermat dijo que este tipo de ecuación con un N > 2 y con x y z > 0 no tiene solución.
    - Muchos matemáticos famosos intentaron resolverlo y no pudieron.
    - Se pudo resolver finalmente gracias a Andrew Wiles
        - Tomas las ecuaciones diofánticas para crear expresiones binomiales de suerte tal que la parte izquierda de esta se reemplaza con números naturales para ver cómo reacciona la parte derecha.

113. Problema de decisión

- En tiempo polinomial: tratables
- En tiempo exponencial: intratables (PARA LOS EQUIPAMIENTOS DE HOY)

114. La **entropía** (2da ley de la termodinámica) está relacionada con la complejidad como concepto abstracto. La biología también. La teoría de sistemas deriva de la biología.

115. Constante o número de Chaitin: es incomputable, esta entre 0 y 1, y representa la complejidad en un cálculo y ataca a los números naturales, dependiendo de la cantidad de decimales, se pueden tener en el cálculo el valor de la complejidad que está representado en términos probabilísticos por Chaitin.

- revoleando una moneda al azar, al cabo de un tiempo puede tener una sucesión de 1s y 0s que están en ese reservorio de programas elegantes.
    - En castellano: Como va a calcular si tiene como objetivo la sucesión que representa a un número natural en particular
- mezcla lo incomputable con lo aleatorio, porque decidir que esa sucesión de números es aleatoria es indecidible / indemostrable. Destruye el concepto de los 90 de la complejidad computacional, porque es también binario.

116. Programa elegante: aquel que es irreducible

117. La **complejidad** se mide en términos **de probabilidad / probabilísticos.**