ARQUITECTURA Y DISEÑO DE SISTEMAS

ESTILOS Y PATRONES DE ARQUITECTURA – PARTE 2

ELSA ESTEVEZ
UNIVERSIDAD NACIONAL DEL SUR
DEPARTAMENTO DE CIENCIAS E INGENIERIA DE LA COMPUTACION

CONTENIDO

1	REPASO EJEMPLO COMÚN
2	CLASIFICACIÓN - ESTILOS SIMPLES

- ESTILOS DE MEMORIA COMPARTIDA
- ESTILOS DE INTÉRPRETE
- ESTILO DE INVOCACIÓN IMPLÍCITA
- ESTILO PEER TO PEER

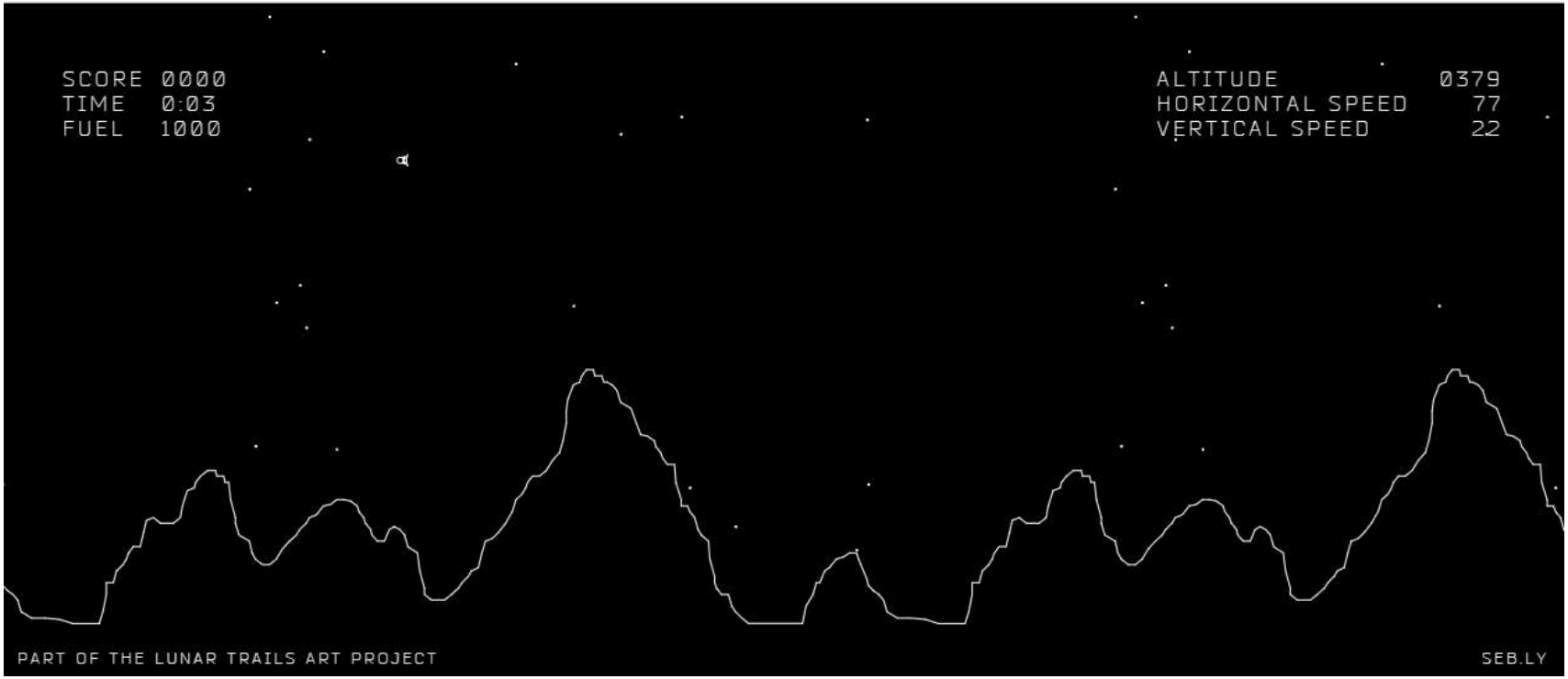
REPASO EJEMPLO – LUNAR LANDER

Lunar Lander es un videojuego desarrollado por Atari, Inc. en 1979. El juego trata sobre pilotear un módulo lunar y alunizarlo de manera segura en la luna.

Fecha de lanzamiento al mercado: agosto de 1979

Géneros: Videojuego de simulación de vehículos

Plataformas: Arcade, Microsoft Windows, Xbox 360, Windows Phone”



REPASO LUNAR LANDER – DESCRIPCIÓN

- Juego popular en los años 70
- Referencia: http://moonlander.seb.ly/
- Reglas
  - ✓ El piloto controla el descenso de un lunar lander estilo Apollo
  - ✓ El piloto controla la aceleración del descenso
  - ✓ El combustible es limitado
  - ✓ La altitud inicial y la velocidad están predefinidas
  - ✓ Si aterriza con una velocidad de descenso < 5, gana aunque no tenga combustible

UNA PROPUESTA

ESTILOS SIMPLES

Estilos influenciados por lenguajes tradicionales
- Programa principal y subrutinas
- Orientación a objetos ✓

Por capas
- Maquinas virtuales
- Cliente-servidor ✓

Estilos de Flujo de Datos
- Batch sequencial
- Pipe and Filters ✓

Memoria compartida
- Blackboard
- Basado en reglas

Intérprete
- Intepreter
- Mobile Code

Invocación Implícita
- Publish-Subscriber
- Event Based

Peer to Peer

ESTILOS COMPLEJOS

- C2
- Objetos distribuidos

Ref: *Software Architecture, Foundations Theory and Practice*

ESTILO DE MEMORIA COMPARTIDA

CARACTERÍSTICAS

- Múltiples componentes acceden al mismo almacenamiento de datos.
- Los componentes se comunican entre ellos a través de dicha memoria
- El diseño se centra especialmente en los repositorios compartidos
- Se originaron con el uso de datos globales en aplicaciones en C y Pascal.

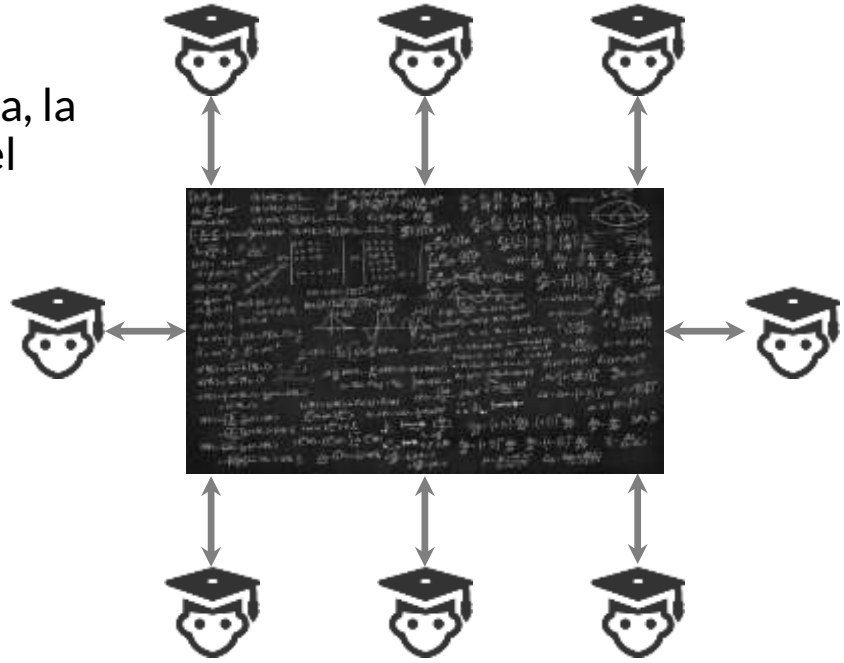
ESTILOS

Blackboard & Repository

Basado en reglas

1- BLACKBOARD

- Expertos cooperando para resolver un problema planteado en un pizarrón
- Cada uno identifica una parte del problema, la resuelve y pone el resultado de nuevo en el pizarrón
- Con esa solución, otro experto puede identificar otro sub-problema y resolverlo
- Así se continua hasta resolver el problema general
- El estado de la información en el pizarrón determina el orden de ejecución de los distintos programas expertos.



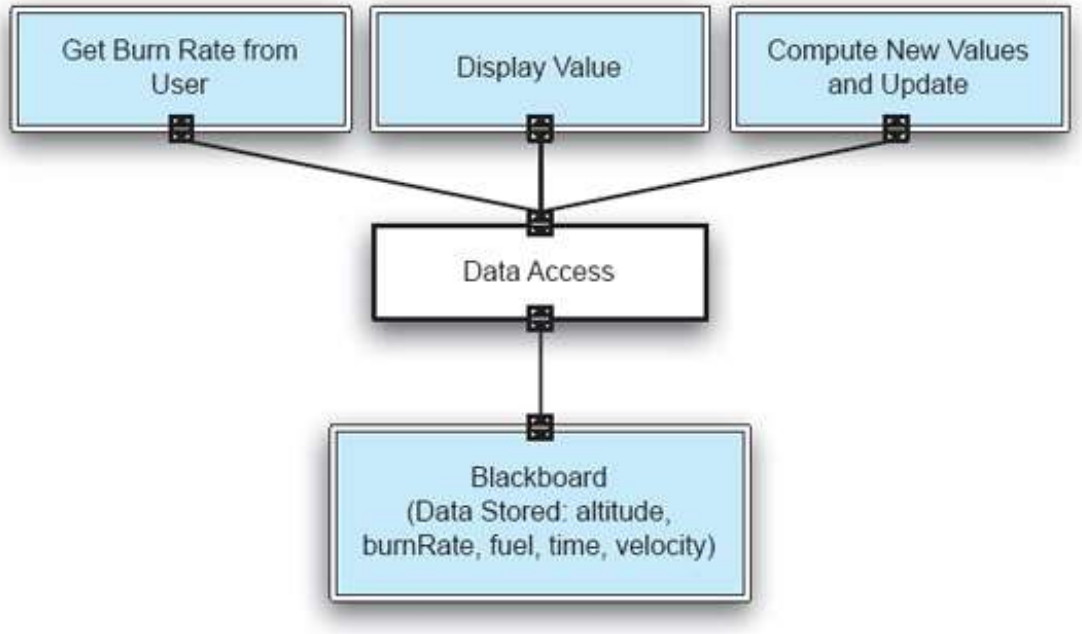
BLACKBOARD – ANÁLISIS 1

DESCRIPCIÓN	Programas independientes que acceden y se comunican a través de un repositorio de datos global (blackboard).
COMPONENTES	Blackboard (estructura central de datos) Programas independientes operando sobre la pizarra.
CONECTORES	Acceso al blackboard ▪ Referencia directa a memoria, llamada a procedimiento, consultas a la base de datos, ...
ELEMENTOS DE DATOS	Datos almacenados en el blackboard
TOPOLOGÍA	Estrella, con el blackboad al medio

BLACKBOARD – ANÁLISIS 2

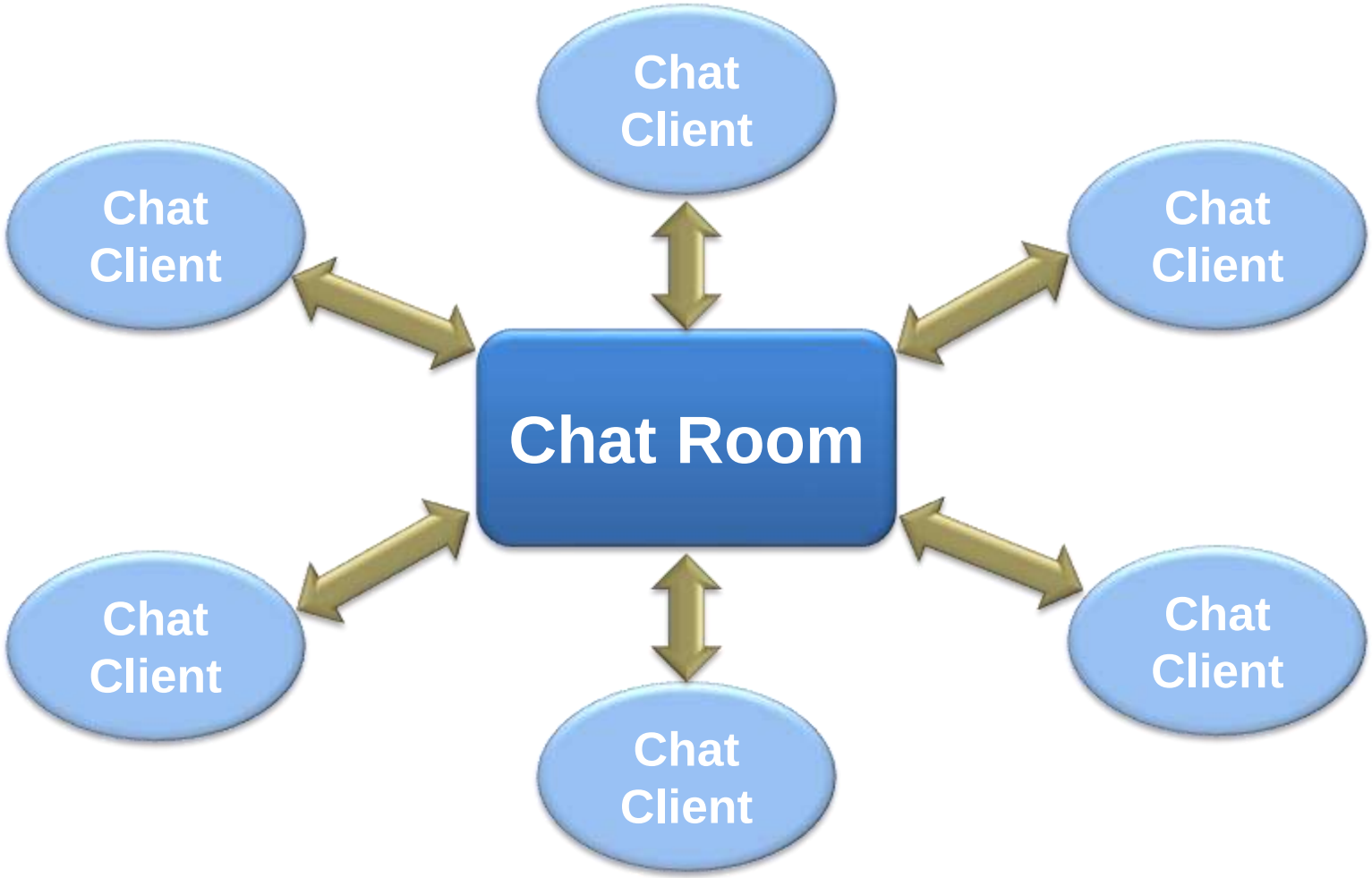
RESTRICCIONES ADICIONALES	Detección de cambios en el blackboard ▪ Polling sobre el blackboard ▪ Blackboard Manager se encarga de notificar cambios
CUALIDADES	▪ La solución completa a un problema no tiene que ser pre-planificada. La evolución del estado determina las estrategias a ser adoptadas.
USOS TÍPICOS	▪ Resolución de problemas heurísticos en inteligencia artificial ▪ Compiladores
PRECAUCIONES	▪ Si la interacción entre programas “independientes” necesita de reglas de regulación compleja ▪ Cuando los datos en el blackboard están sujetos a cambios frecuentes y se requiere propagarlos entre todos componentes participantes. ▪ Existe otra estrategia más simple

BLACKBOARD - DISEÑO



- *Blackboard* - mantiene el estado del juego
- *Get Burn Rate from User* - actualiza la velocidad de descenso en base el input del usuario
- *Display Value* - muestra al usuario el estado de la nave y otros aspectos del juego
- *Compute New Values and Update* - actualiza el estado del juego en base al tiempo y el modelo físico
- *Data Access* - es el conector

EJEMPLO - CHAT ROOM



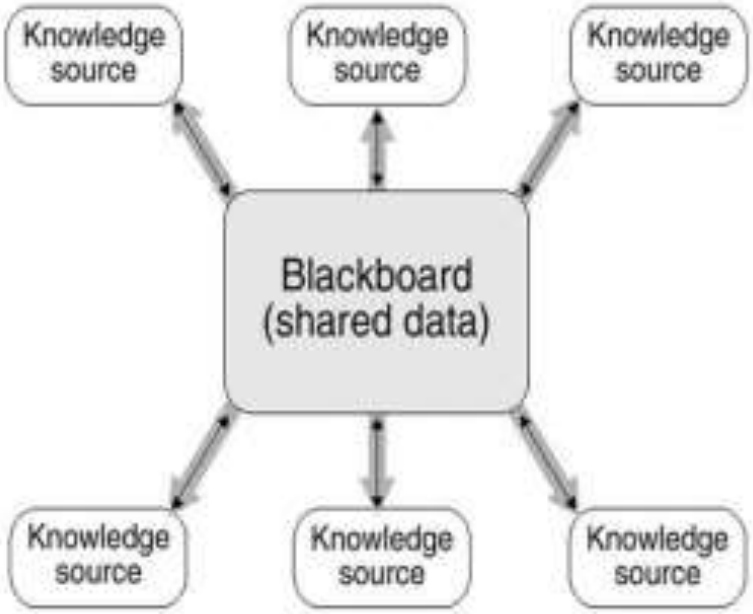
REPOSITORIO

- Es una especialización de Blackboard
- El repositorio compartido solamente recibe y procesa los requerimientos de sus clientes, no hay notificaciones

Ejemplo: Sistema Universitario

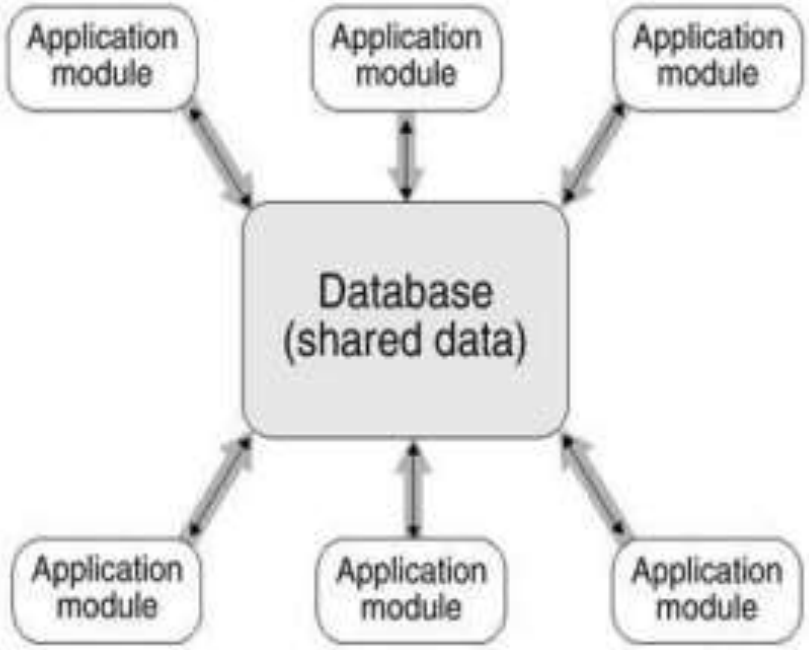
DIFERENCIAS ENTRE BLACKBOARD Y REPOSITORY

BLACKBOARD



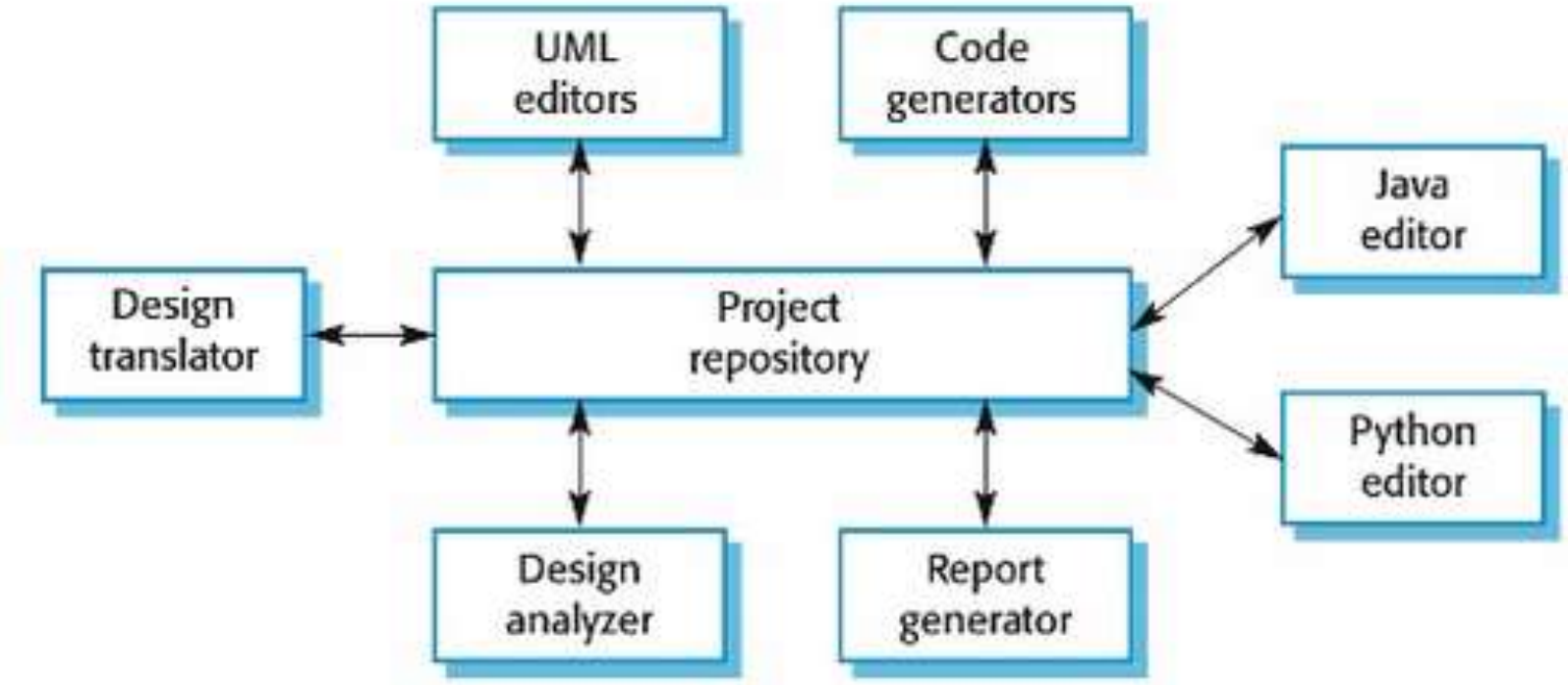
Agente Activo
Notifica a clientes los cambios en el estado compartido

REPOSITORY



Agente Pasivo
Los clientes deben enviar requerimientos y reciben respuestas

EJEMPLO – ENTORNO DE DESARROLLO INTEGRADO (IDE)



2 – BASADO EN REGLAS

- Es un tipo de arquitectura de memoria compartida altamente especializada.
- La memoria compartida se denomina base de conocimiento.
  - ✓ Contiene "hechos" (sentencias de valores de variable) y "reglas de producción" que consisten en condicionales sobre los hechos
- Almacena y manipula conocimiento para interpretar información de manera útil
- El estilo es especialmente usado en inteligencia artificial
  - ✓ Diagnóstico médico basado en síntomas
  - ✓ Movimiento de juegos deducido

2 – BASADO EN REGLAS - COMPONENTES

INTERFAZ DE USUARIO	Provee dos modos: ✓ Ingresar hechos y reglas ✓ Ingresar consultas (goals)
MOTOR DE INFERENCIAS	Opera sobre la base de conocimiento en respuesta a una entrada de usuario: ✓ Hechos y reglas son agregados a la base de conocimiento ✓ Las consultas son comparadas contra los hechos existentes ✓ Match exacto - retorna true ✓ Si no hay match exacto - evalúa las reglas correspondientes para determinar la validez de la consulta
BASE DE CONOCIMIENTO	Memoria compartida que contiene: ✓ Hechos: sentencias de valores de variables ✓ Reglas: Cláusulas “if... then” sobre el conjunto de variables

BASADO EN REGLAS – ANÁLISIS 1

DESCRIPCIÓN	El motor de inferencias parsea la entrada del usuario ▪ Si es un hecho/regla, la agrega a su base de conocimiento ▪ Si es una consulta (goal), obtiene las reglas aplicables desde la base de conocimiento e intenta resolverla.
COMPONENTES	Interfaz de usuario, motor de inferencias, base de conocimiento
CONECTORES	Los componentes están estrechamente conectados a través de: ▪ llamadas a procedimientos ▪ acceso a datos compartidos
ELEMENTOS DE DATOS	Reglas/hechos y consultas
TOPOLOGÍA	3 capas altamente acopladas: ✓ interfaz de usuario ✓ motor de inferencia ✓ base de conocimiento

BASADO EN REGLAS – ANÁLISIS 2

CUALIDADES	▪ Modificabilidad - el comportamiento de la aplicación puede ser modificado agregando o eliminando reglas dinámicamente ▪ Facilita el prototipado de sistemas pequeños
USOS TÍPICOS	Cuando el problema puede ser entendido como una cuestión de resolver repetidamente un conjunto de predicados
PRECAUCIONES	Cuando se tiene una gran cantidad de reglas, entender las interacciones entre múltiples reglas afectadas por los mismos hechos llega a ser difícil.

BASADO EN REGLAS – DISEÑO

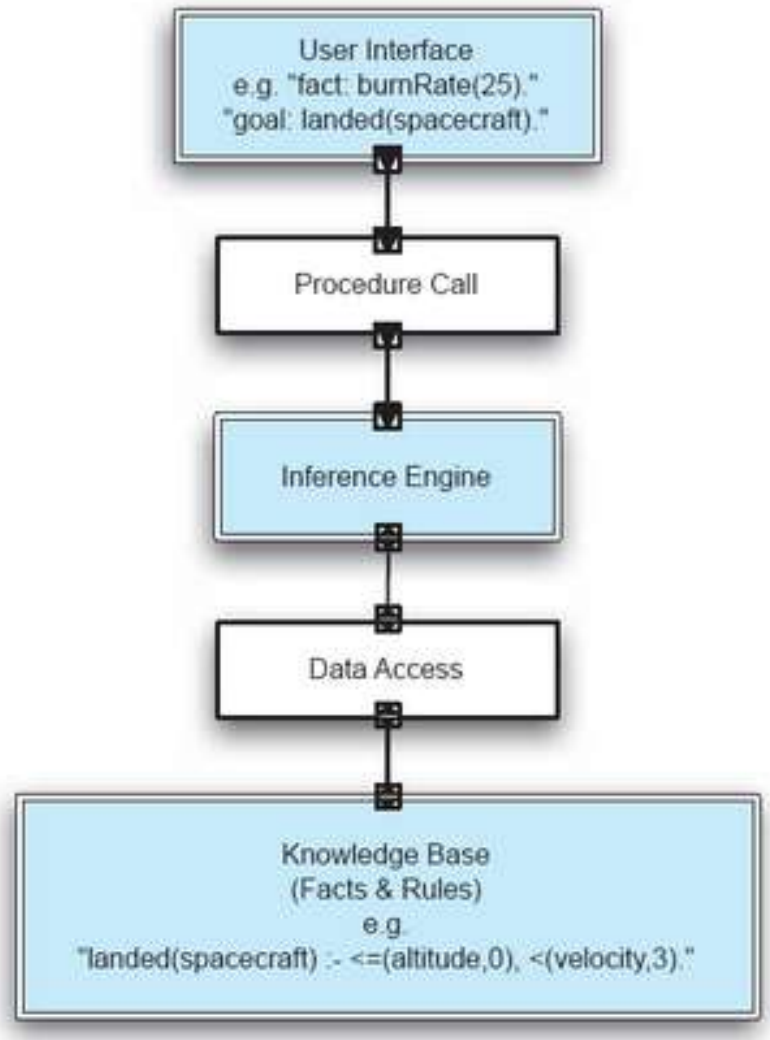
- Pensar el problema como un conjunto de hechos y reglas
- Se carga como hecho la velocidad de combustión

  Hecho: burnRate(25)

- Para ver el estado de la nave, el usuario cambia al modo consulta y pregunta si la nave aterrizó con éxito

  Goal: landed(spacecraft)

- Para resolver la consulta, el motor de inferencia consulta la base de conocimiento, si los hechos existentes y reglas de producción satisfacen la condición “altitud <= 0” y “velocidad < 5” retorna verdadero, caso contrario falso.



CONTENIDO

1 REPASO EJEMPLO COMÚN

2 CLASIFICACIÓN - ESTILOS SIMPLES

- ESTILOS DE MEMORIA COMPARTIDA
- ESTILOS DE INTÉRPRETE
- ESTILO DE INVOCACIÓN IMPLÍCITA
- ESTILO PEER TO PEER

UNA PROPUESTA

ESTILOS SIMPLES

Estilos influenciados por lenguajes tradicionales
- Programa principal y subrutinas ✓
- Orientación a objetos

Por capas
- Maquinas virtuales
- Cliente-servidor ✓

Estilos de Flujo de Datos
- Batch sequencial
- Pipe and Filters ✓

Memoria compartida
- Blackboard
- Basado en reglas ✓

Intérprete
- Intepreter
- Mobile Code

Invocación Implícita
- Publish-Subscriber
- Event Based

Peer to Peer

ESTILOS COMPLEJOS

- C2
- Objetos distribuidos

Ref: *Software Architecture, Foundations Theory and Practice*

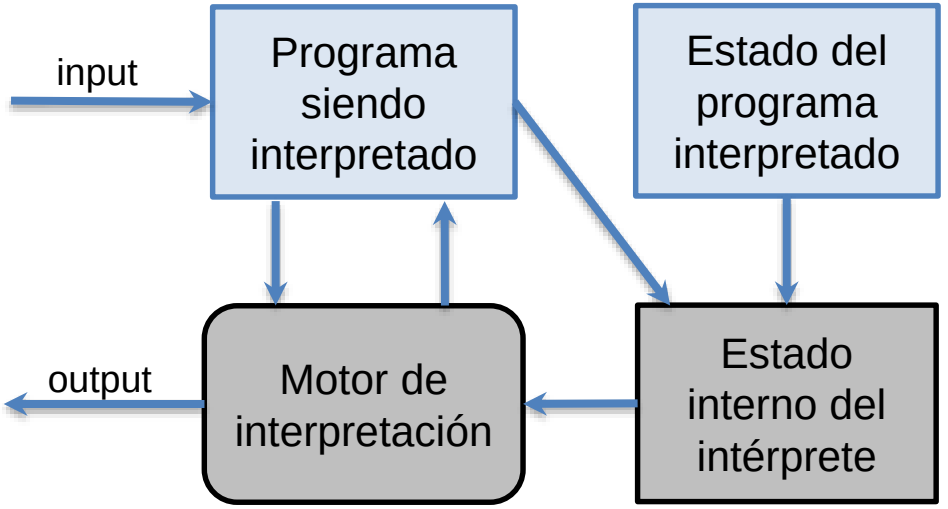
ESTILOS DE INTÉRPRETE

CARACTERÍSTICAS

- Interpretación dinámica y on-the-fly de comandos
- Los comandos son definidos en términos de comandos primitivos predefinidos.
- Proceso de interpretación
  - ✓ Comienza con un estado de ejecución inicial (datos iniciales)
  - ✓ Obtiene el primer comando a ejecutar
  - ✓ Ejecuta el comando sobre el estado de ejecución actual (probablemente modificando dicho estado)
  - ✓ Identifica el próximo comando a ejecutar (probablemente afectado por el resultado del comando anterior – por ejemplo condicionales)
  - ✓ Ejecuta el siguiente comando
- El estilo arquitectónico interprete básico implica la ejecución uno a uno de comandos.

3 – INTÉRPRETE BÁSICO

- Se puede decir que es similar al estilo basado en reglas: el motor de inferencia parsea el comando de entrada y lo resuelve en base a la base de conocimiento.
  - ✓ Toma un programa escrito en un lenguaje y lo interpreta a otro lenguaje para ejecutar una serie de comandos
- Facilita la codificación en un lenguaje de más alto nivel
- Ventajas
  - ✓ Portabilidad y flexibilidad de aplicaciones o lenguajes
  - ✓ Soporte dinámico de cambios
- Ejemplos
  - ✓ LISP, Perl
  - ✓ Fórmulas de Excel



INTÉRPRETE BÁSICO – ANÁLISIS 1

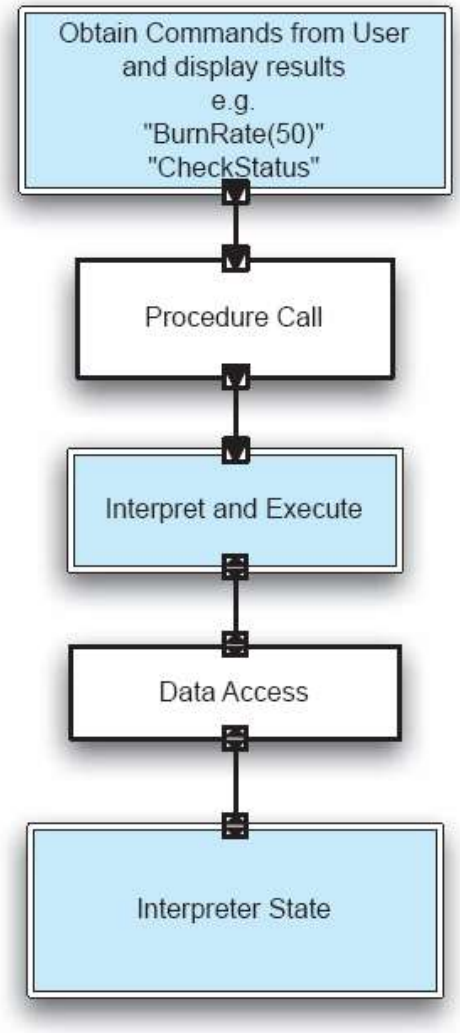
DESCRIPCIÓN	Parsea y ejecuta comandos de entrada, actualizando el estado mantenido por el intérprete
COMPONENTES	▪ intérprete de comandos ▪ estado del programa interpretado ▪ estado del intérprete ▪ la interfaz de usuario
CONECTORES	Los componentes están estrechamente conectados a través de: ▪ llamadas a procedimientos ▪ acceso a datos compartidos
ELEMENTOS DE DATOS	Comandos

INTÉRPRETE BÁSICO – ANÁLISIS 2

TOPOLOGÍA	3 capas altamente acopladas. El estado puede estar separado del intérprete.
CUALIDADES	▪ Es posible obtener un comportamiento altamente dinámico, donde el conjunto de comandos se va modificando. ▪ La arquitectura del sistema puede ser constante mientras se crean nuevas capacidades a partir de primitivas existentes
USOS TÍPICOS	end-user programing
PRECAUCIONES	Cuando se necesita un procesamiento rápido (el código interpretado tarda mucho más que el código ejecutable).

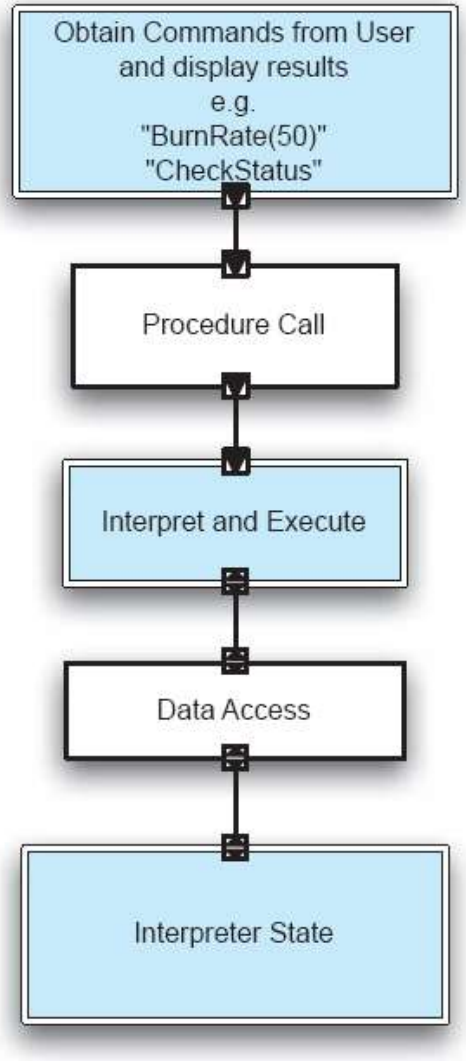
INTÉRPRETE BÁSICO – DISEÑO

- Para cada comando ingresado
  - ✓ Se procesa el código
  - ✓ Se actualiza el estado del intérprete
- Comandos:
  - ✓ directivas específicas considerando cómo sería manipulada la nave
  - ✓ comandos que actualizan estado
  - ✓ comandos que muestran información



INTÉRPRETE BÁSICO – DISEÑO (CONT.)

- Ejecución
  - ✓ El usuario ingresa BurnRate(50)
  - ✓ El intérprete toma el comando y su parámetro y lo interpreta como la cantidad de combustible a quemar
  - ✓ El intérprete calcula la necesidad de actualización de la altitud, nivel de combustible y velocidad.
  - ✓ Se simula el tiempo incrementándolo cada vez que ingresa un comando
  - ✓ Cuando el usuario entra el comando CheckStatus, se retorna el estado actual de la altitud, combustible, tiempo y velocidad.



4 – CÓDIGO MÓVIL

Algunas veces la interpretación no se puede resolver de manera local.

Variantes:

1) Código en demanda
2) Ejecución remota/evaluación
3) Agente móvil

4 – CÓDIGO EN DEMANDA

CARACTERÍSTICAS

- El cliente cuenta con los recursos y el poder de procesamiento
- El servidor mantiene el código a ser ejecutado
- El cliente le requiere el código al servidor y lo ejecuta

4 – EJECUCIÓN REMOTA/EVALUACIÓN

CARACTERÍSTICAS

- El cliente mantiene el código pero no tiene recursos para ejecutarlo (no tiene el “interprete” de software o no tiene capacidad de procesamiento)

4 – AGENTE MÓVIL

CARACTERÍSTICAS

- El iniciador tiene el código a ejecutar, pero no todos los recursos
- En forma autónoma decide migrar a otro nodo para obtener recursos adicionales

CÓDIGO MÓVIL– ANÁLISIS 1

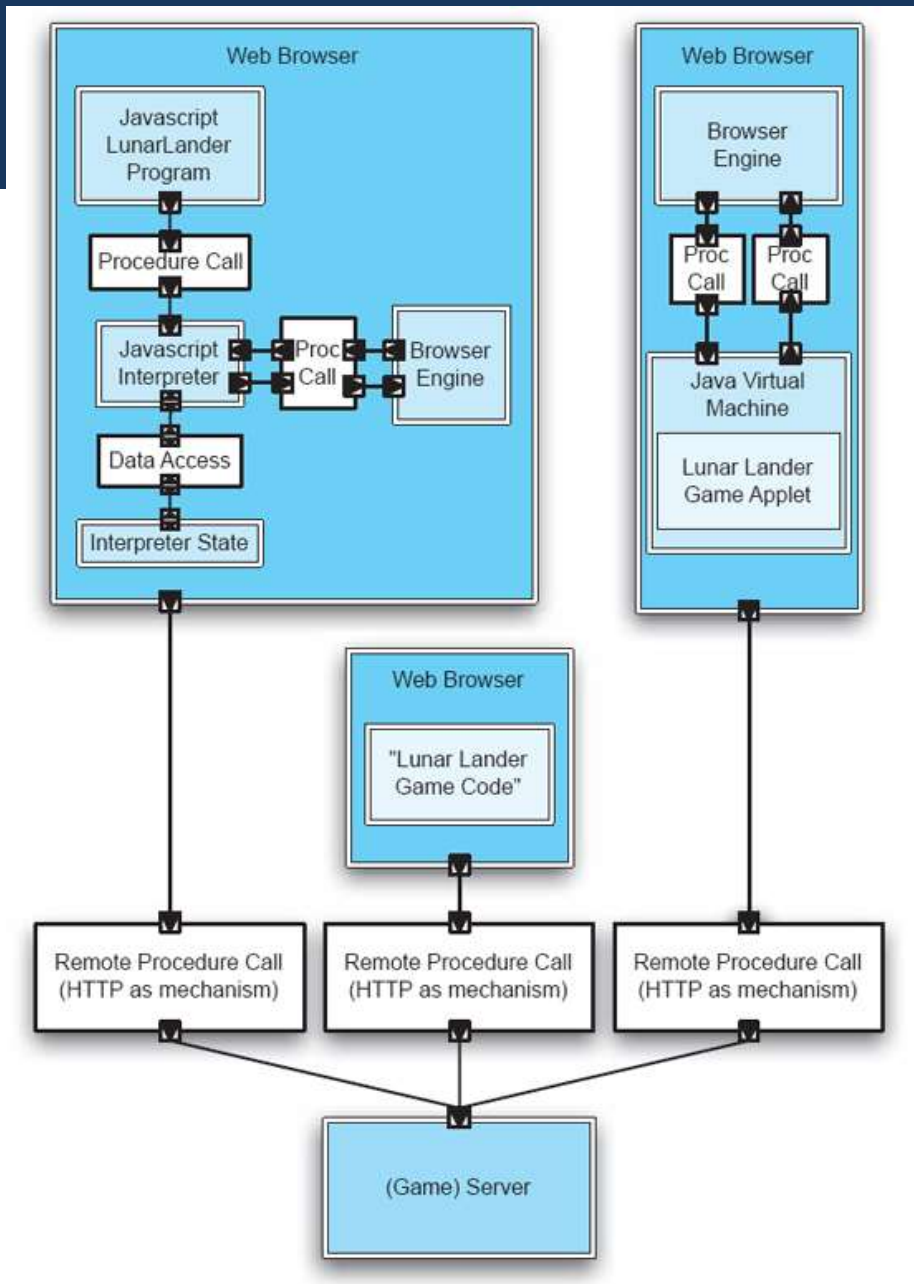
DESCRIPCIÓN	El código se mueve para ser interpretado en otro host.
COMPONENTES	▪ dock de ejecución (recepción y deployment de código y estado) ▪ intérprete/compilador de código
CONECTORES	Protocolos de red y elementos para empaquetar código y datos para transmisión
ELEMENTOS DE DATOS	▪ representación de código como datos ▪ estado del programa ▪ datos
TOPOLOGÍA	Red

CÓDIGO MÓVIL– ANÁLISIS 2

CUALIDADES	▪ Adaptabilidad dinámica ▪ Toma ventaja del poder de procesamiento del host ▪ Confianza - se incrementa al permitir migrar a un nuevo host de manera simple.
USOS TÍPICOS	Cuando se procesan grandes conjuntos de datos en locaciones distribuidas (es más eficiente que el código se mueva al lugar donde se encuentran esos datos)
PRECAUCIONES	▪ Seguridad - La ejecución de código importado abre la puerta a malware. ▪ Cuando el costo de transmisión excede al costo de ejecución ▪ Cuando las conexiones de red no están disponibles

CÓDIGO MÓVIL – DISEÑO

- Los clientes descargan por HTTP el código-por-demanda del Lunar Lander de distintas maneras:
  - ✓ versión Javascript
  - ✓ versión applet
  - ✓ otra versión
- Toda la lógica del juego se mueve a las máquinas cliente
- El servidor está más liberado
- Cada máquina cliente mantiene el estado del juego de manera independiente de otros clientes.



CONTENIDO

1 REPASO EJEMPLO COMÚN

2 CLASIFICACIÓN - ESTILOS SIMPLES

- ESTILOS DE MEMORIA COMPARTIDA
- ESTILOS DE INTÉRPRETE
- ESTILO DE INVOCACIÓN IMPLÍCITA
- ESTILO PEER TO PEER

UNA PROPUESTA

ESTILOS SIMPLES

Estilos influenciados por lenguajes tradicionales
- Programa principal y subrutinas ✓
- Orientación a objetos

Por capas
- Maquinas virtuales
- Cliente-servidor ✓

Estilos de Flujo de Datos
- Batch sequencial
- Pipe and Filters ✓

Memoria compartida
- Blackboard
- Basado en reglas ✓

Intérprete
- Intepreter
- Mobile Code ✓

Invocación Implícita
- Publish-Subscriber
- Event Based

Peer to Peer

ESTILOS COMPLEJOS

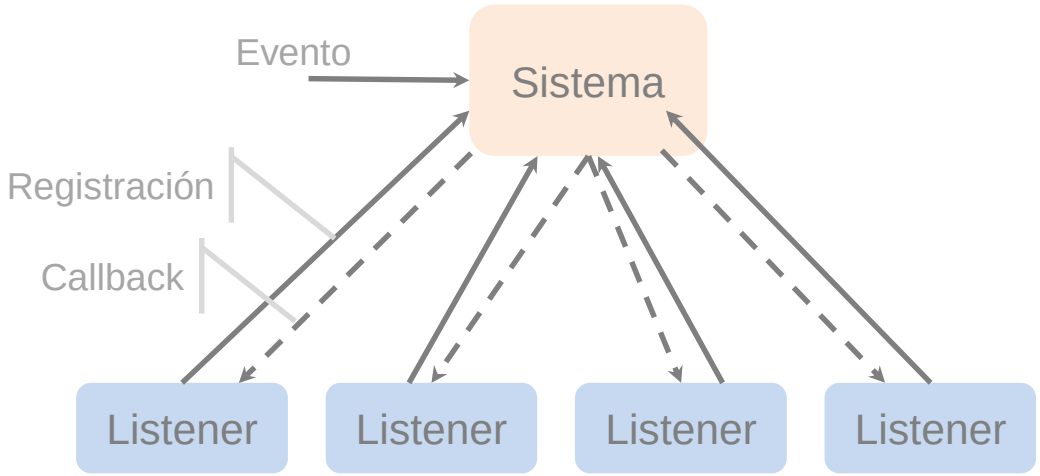
- C2
- Objetos distribuidos

Ref: *Software Architecture, Foundations Theory and Practice*

ESTILOS DE INVOCACIÓN IMPLÍCITA

CARACTERÍSTICAS

- Se anuncian los eventos en vez de invocarse métodos
- Los “listeners” se registran como interesados y asocian métodos (callbacks) con eventos.
- Al producirse un evento, el sistema invoca a todos los métodos registrados
- Quien anuncia el evento, no sabe a quién afectará
- No hay suposiciones sobre el orden de procesamiento en respuesta a eventos



5 – PUBLISH- SUBSCRIBER

CARACTERÍSTICAS

- Toma su nombre de la relación análoga entre publicadores de diarios y revistas y sus subscriptores
- Es usado para enviar eventos y mensajes a un conjunto desconocido de receptores.
- El conjunto de receptores es desconocido para el productor del evento, la correctitud del productor no puede depender de los receptores.
- Nuevos receptores puede agregarse sin cambios en el/los productor/es

5 – PUBLISH- SUBSCRIBER

ALCANCE

- Productores y consumidores de información están bien distinguidos
- Ejemplo: Servicios de ofertas laborales on-line

COMUNICACIÓN

- Invocaciones locales: Métodos y callbacks
- Invocaciones remotas: Protocolos de red y proxies

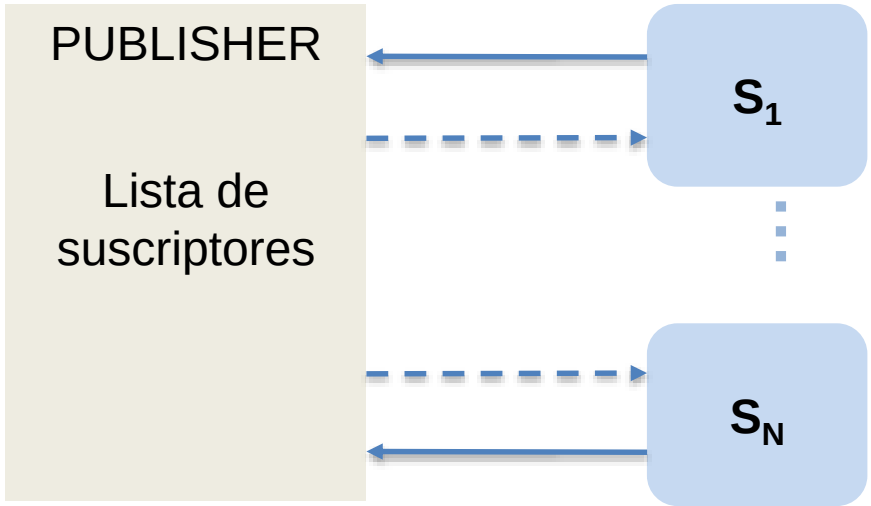
5 – PUBLISH- SUBSCRIBER - TIPOS

1) Basados en Listas
2) Basados en Broadcast
3) Basados en Contenido

5 – PUBLISH- SUBSCRIBER BASADOS EN LISTAS

CARACTERÍSTICAS

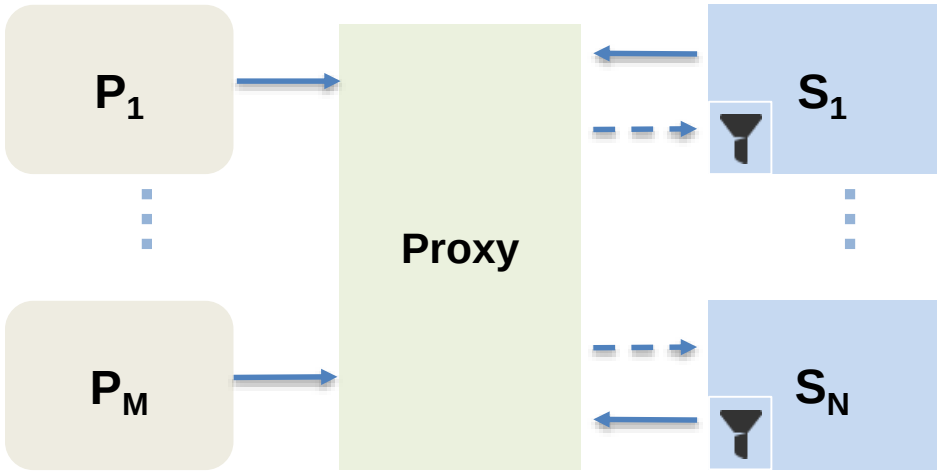
- Cada Publisher mantiene una lista de suscripciones



5 – PUBLISH- SUBSCRIBER BASADOS EN BROADCAST

CARACTERÍSTICAS

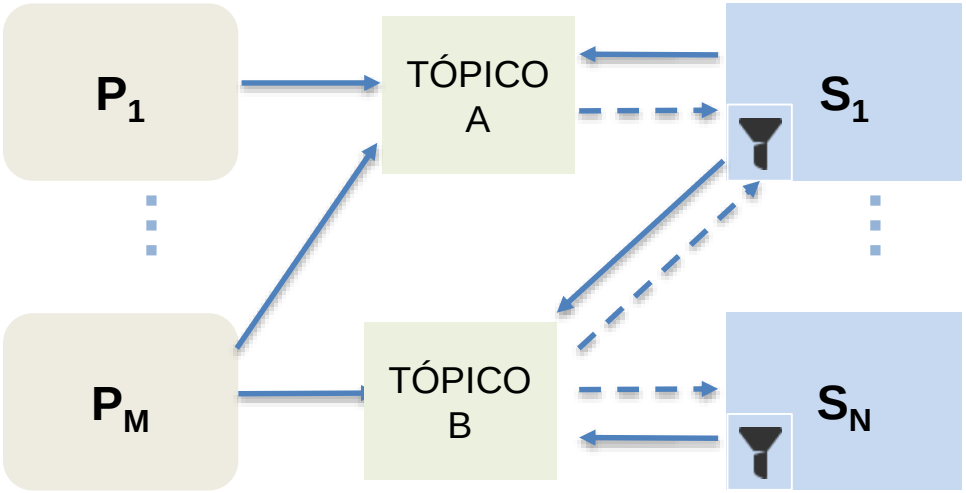
- Los Publishers tienen poco (o ningún) conocimiento de los Subscribers.
- Todos los eventos son emitidos a todos los Subscribers
- Los Subscribers deben filtrar los eventos que son de su interés



5 – PUBLISH- SUBSCRIBER BASADOS EN CONTENIDO

CARACTERÍSTICAS

- También conocida como “basada en tópicos”
- Los tópicos son tipos de eventos o mensajes predefinidos



PUBLISH- SUBSCRIBER

VENTAJAS	DESVENTAJAS
○ Desacoplamiento ○ Escalabilidad	○ Se agrega una capa de direccionamiento afectando la latencia ○ No se garantiza la entrega de mensajes, ni el orden en el que llegan ○ Disponiblidad

EJEMPLOS	
○ Interfaces de usuarios gráficas ○ Aplicaciones basadas en MVC ○ Ambientes de desarrollo extensibles	○ Listas de correo ○ Redes sociales

PUBLISH-SUBSCRIBER– ANÁLISIS 1

DESCRIPCIÓN	Subscribers se registran/desregistran para recibir mensajes o contenidos específicos. Cuando el Publisher publica, el mensaje es enviado a los Subscribers
COMPONENTES	○ publishers ○ subscribers ○ proxies para manejar la distribución
CONECTORES	Llamadas a procedimientos pueden ser usadas dentro de un programa. Protocolos de red son más frecuentes.
ELEMENTOS DE DATOS	○ suscripciones ○ notificaciones ○ información publicada
TOPOLOGÍA	Subscribers se conectan a Publishers en forma directa o pueden recibir notificaciones de intermediarios

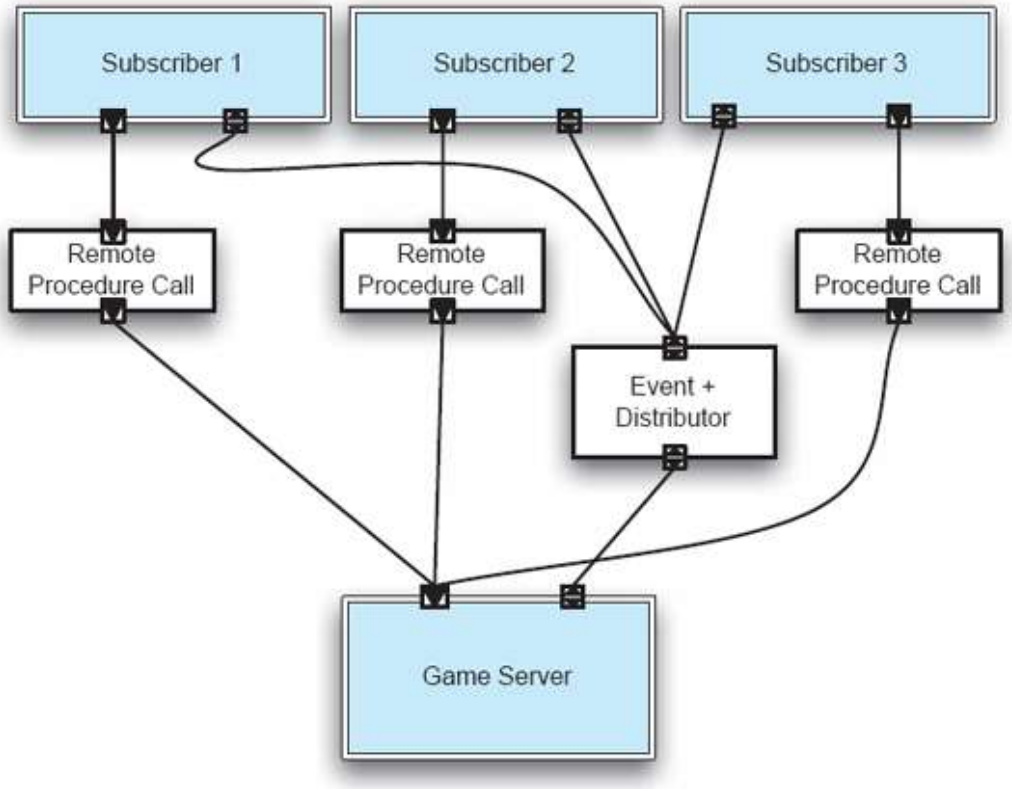
PUBLISH-SUBSCRIBER– ANÁLISIS 2

CUALIDADES	Altamente eficiente para distribuir información en un solo sentido con muy bajo acoplamiento de componentes.
USOS TÍPICOS	○ Distribución de noticias ○ GUIs ○ Juegos en red multi-player
PRECAUCIONES	Cuando la cantidad de Subscribers para un tópico es muy grande, un protocolo especial puede ser necesario.
VARIANTES	• Usos específicos del estilo puede requerir pasos particulares en la suscripción/desuscripción. • Soporte para el matching complejo de intereses y la información disponible puede ser provisto por los intermediarios.

PUBLISH-SUBSCRIBER – DISEÑO

- El Lunar Lander es instalado en varios hosts
- Los jugadores (subscribers) se registran en un servidor que publica:
  - ✓ Nuevos datos del terreno lunar
  - ✓ Nuevas naves
  - ✓ Ubicación (lunar) de otros jugadores

- Los jugadores reciben notificaciones cuando la información a la que se han registrado es actualizada.
  - ✓ Las notificaciones contienen la información en sí misma
  - ✓ Las notificaciones resultan sólo en un aviso de la novedad



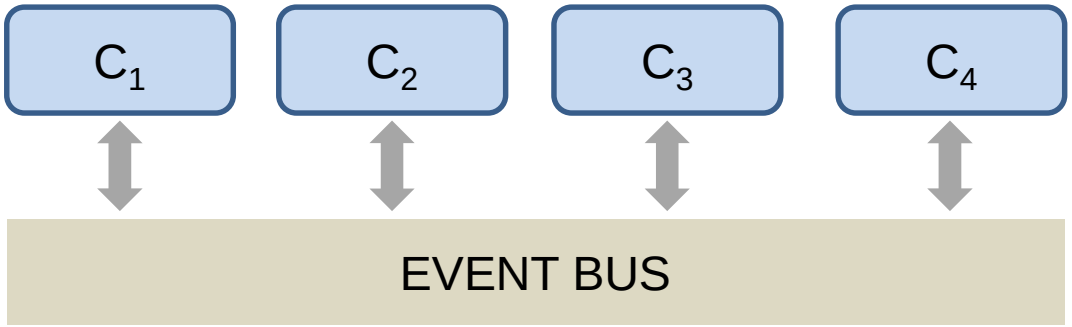
6 – BASADO EN EVENTOS

CARACTERÍSTICAS

- Componentes independientes comunicándose sólo enviando eventos a través de conectores a un event-bus

- Los componentes emiten eventos al event-bus en forma asincrónica, el cual luego los transmite a los otros componentes. Cada componente puede reaccionar ante la recepción de un evento, o ignorarlo.

- Los conectores se encargan de:
  - ✓ optimizar la distribución de eventos
  - ✓ la replicación de eventos (transparente al emisor y receptor)



6 – BASADO EN EVENTOS

OPTIMIZACIÓN

- Solamente distribuir eventos a quienes expresan interés en ellos
  - ✓ Similar a Publish-Subcriber
  - ✓ Diferencia: en Event-Based no hay clasificación como en Publisher y Subcriber, todos pueden emitir y recibir eventos.

TIPOS DE DISTRIBUCIÓN

- Pull (polling) - los componentes consultan al conector por eventos disponibles (bloqueante o no bloquente)
- Push - los eventos arriban a los componentes ni bien se producen

IMPLEMENTACIÓN

- Middlewares o librerías que resuelven el Event-Bus y los conectores (llamados Enterprise Service Bus – ESB)
  - ✓ Mule, JBoss ESB, Oracle Service Bus, Microsoft BizTalk Services, Windows Azure Service Bus, Spring Integration

BASADO EN EVENTOS – ANÁLISIS 1

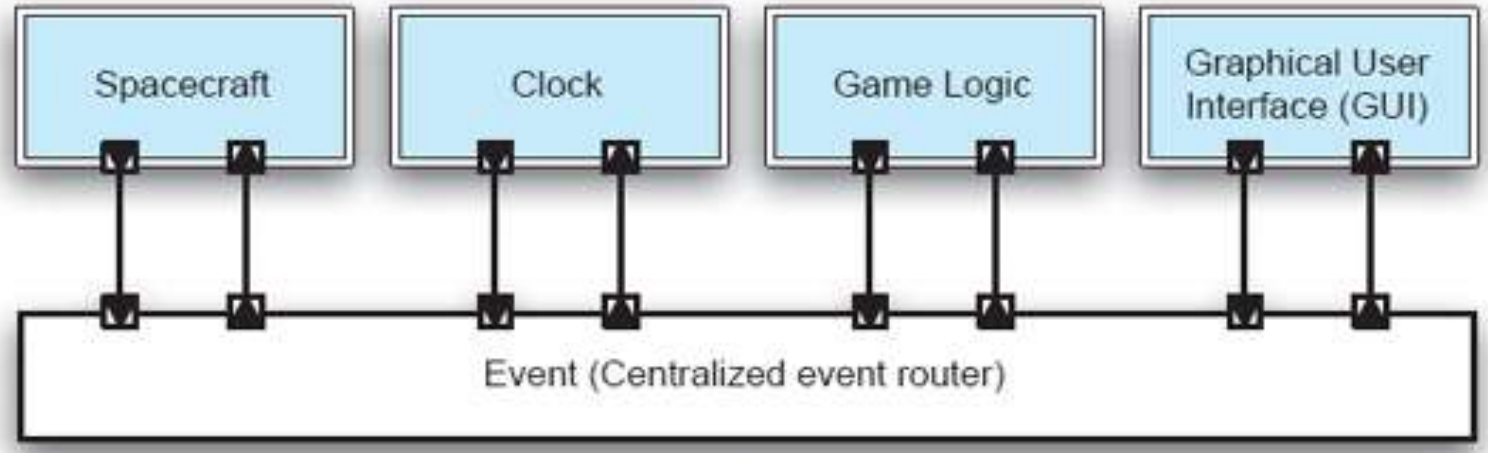
DESCRIPCIÓN	Componentes independientes que asincrónicamente emiten y reciben eventos comunicados a través de event-buses.
COMPONENTES	Generadores y/o consumidores de eventos independientes y concurrentes.
CONECTORES	Event-bus. Podría existir más de uno.
ELEMENTOS DE DATOS	Eventos
TOPOLOGÍA	Los componentes se comunican con el event-bus, no directamente entre ellos.

BASADO EN EVENTOS – ANÁLISIS 2

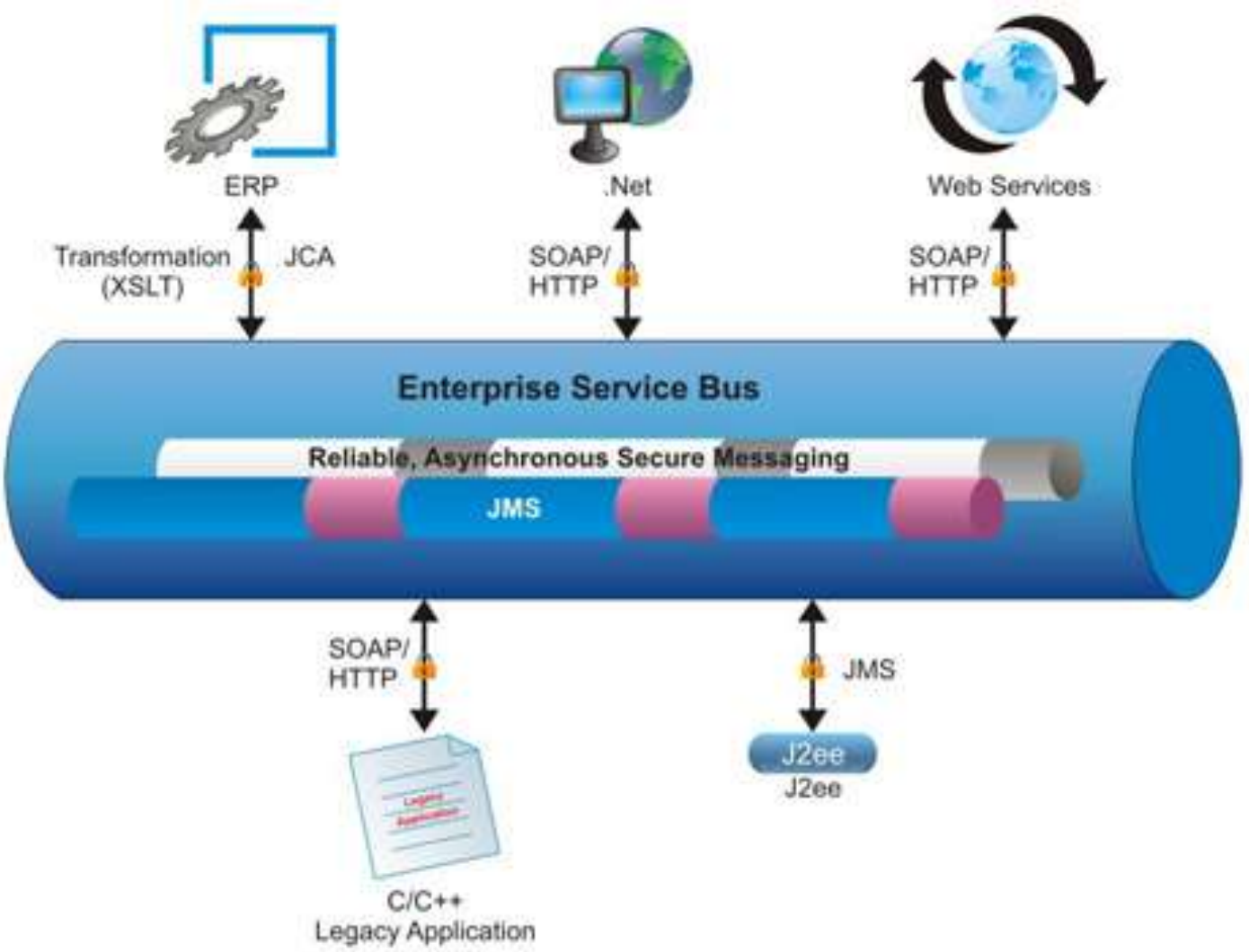
CUALIDADES	• Altamente escalable • Fácil de evolucionar • Efectivo para aplicaciones heterogéneas altamente distribuidas.
USOS TÍPICOS	• Software de UI • Aplicaciones de área amplia que involucran partes independientes (mercados financieros, logística, redes de sensado)
PRECAUCIONES	No existen garantías que un evento sea procesado, ni cuando lo será.
VARIANTES	Comunicación push o pull

BASADO EN EVENTOS – DISEÑO

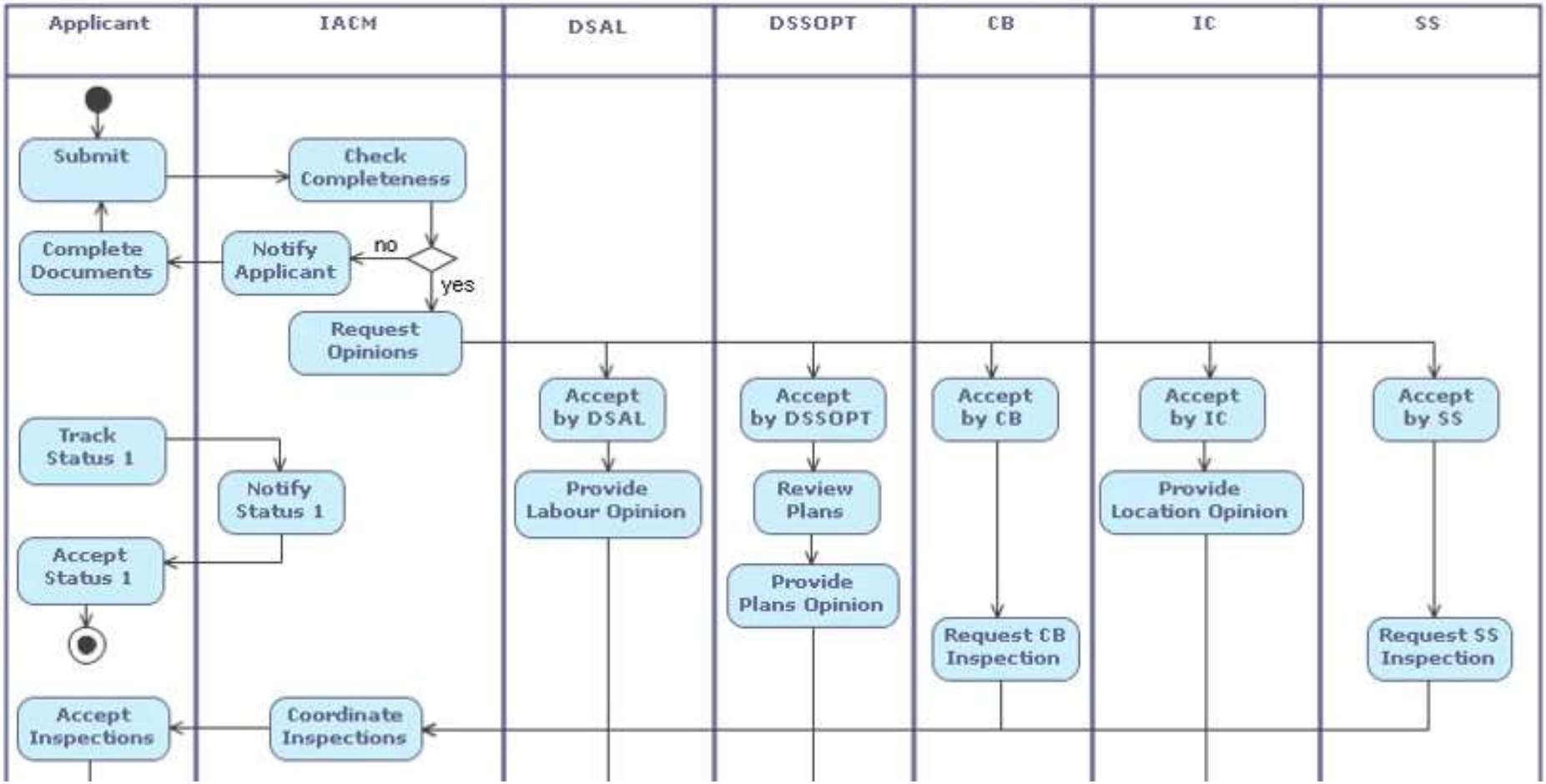
- Componentes
- Reloj: Maneja el juego
- Nave: Mantiene el estado de la nave (altitud, nivel de combustible, velocidad, aceleración)
- Interfaz Gráfica de Usuario (GUI): Maneja la pantalla.
  - ✓ Recibe eventos del estado de la nave para mostrar los datos en pantalla
  - ✓ Obtiene nuevas burn rates del usuario y emite notificaciones al event-bus
- Lógica de Juego: Recibe información sobre el estado de la nave y la cantidad de tiempo que ha pasado, determinando si el juego finalizó y, en ese caso, cuál fue el resultado final y el puntaje, notificándolo al event-bus.



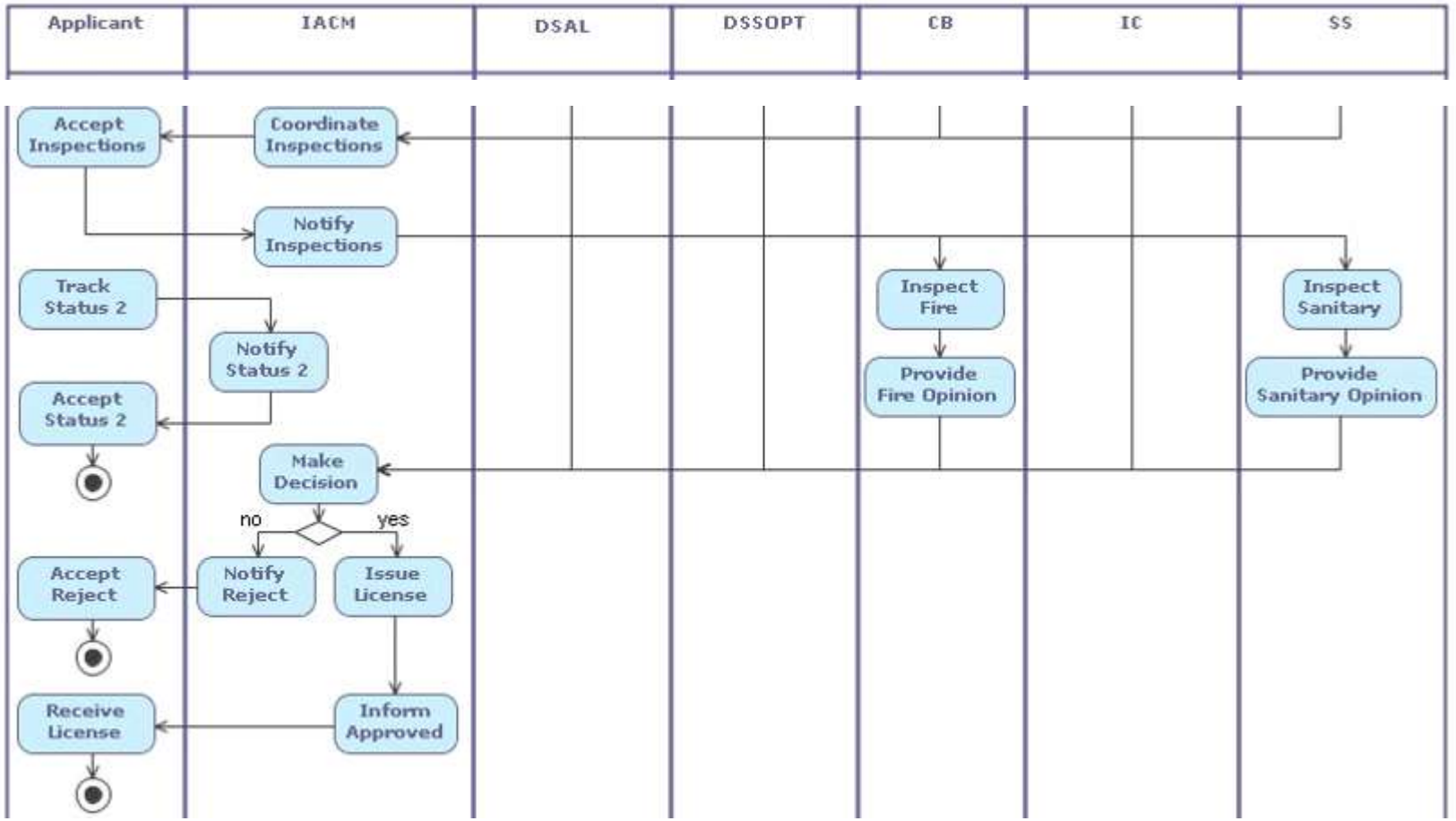
EJEMPLO – ENTERPRISE SERVICE BUS



EJEMPLO – LICENCIAS DE COMERCIO - 1



EJEMPLO – LICENCIAS DE COMERCIO - 2



CONTENIDO

1 REPASO EJEMPLO COMÚN

2 CLASIFICACIÓN - ESTILOS SIMPLES

- ESTILOS DE MEMORIA COMPARTIDA
- ESTILOS DE INTÉRPRETE
- ESTILO DE INVOCACIÓN
- ESTILO PEER TO PEER

UNA PROPUESTA

ESTILOS SIMPLES

Estilos influenciados por lenguajes tradicionales
- Programa principal y subrutinas ✓
- Orientación a objetos

Por capas
- Maquinas virtuales
- Cliente-servidor ✓

Estilos de Flujo de Datos
- Batch sequencial
- Pipe and Filters ✓

Memoria compartida
- Blackboard
- Basado en reglas ✓

Intérprete
- Intepreter
- Mobile Code ✓

Invocación Implícita
- Publish-Subscriber
- Event Based ✓

Peer to Peer

ESTILOS COMPLEJOS

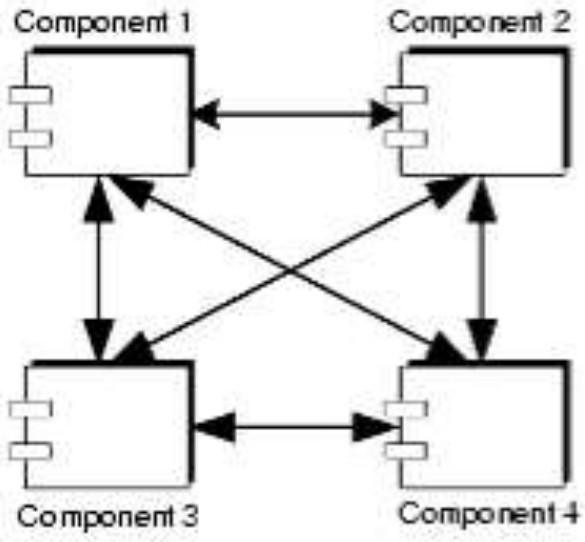
- C2
- Objetos distribuidos

Ref: *Software Architecture, Foundations Theory and Practice*

7 – PEER-TO-PEER 1

CARACTERÍSTICAS

- Consiste de una red de componentes autónomos y débilmente acoplados (pares) que colaboran para proveer un servicio.
- Todos los componentes son iguales y ninguno puede ser crítico para la salud del sistema
- Cada componente provee y consume los mismos servicios y usa el mismo protocolo

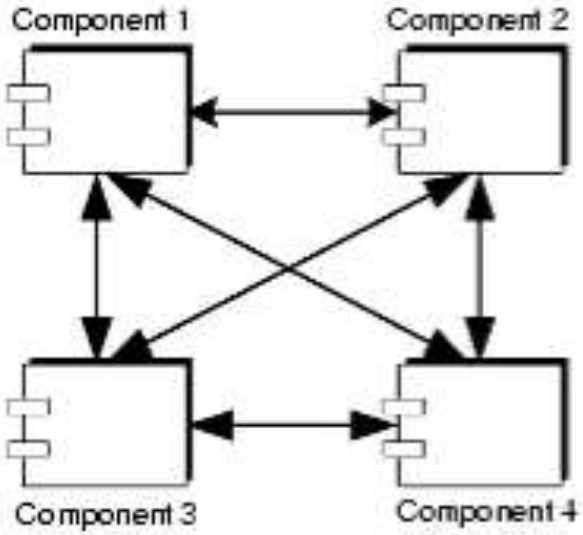


7 – PEER-TO-PEER 2

CARACTERÍSTICAS

- La información, por lo general, es mantenida localmente en cada componente.

- Interacción
  - ✓ Un componente puede interactuar con cualquier otro componente
  - ✓ La comunicación es típicamente una interacción requerimiento/respuesta
  - ✓ La interacción puede ser iniciada por cualquier parte (en el sentido client-server) y cada componente es tanto cliente como servidor



7 – PEER-TO-PEER – DESCUBRIMIENTO DE RECURSOS

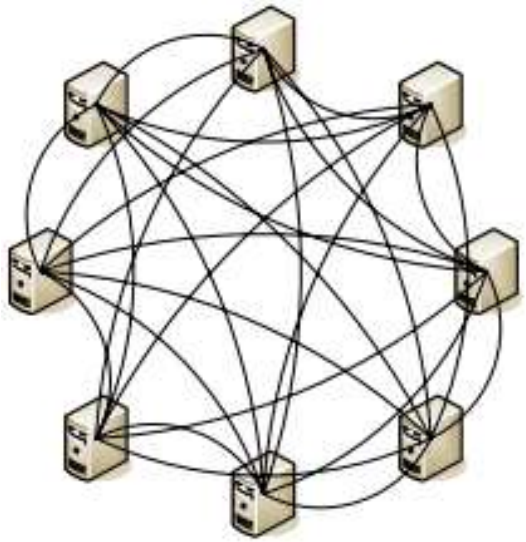
La ausencia de centralización hace que la búsqueda de recursos sea un tema importante en P2P.

Tres opciones:

- P2P Puro
- P2P Híbrido
- Servidor Índice

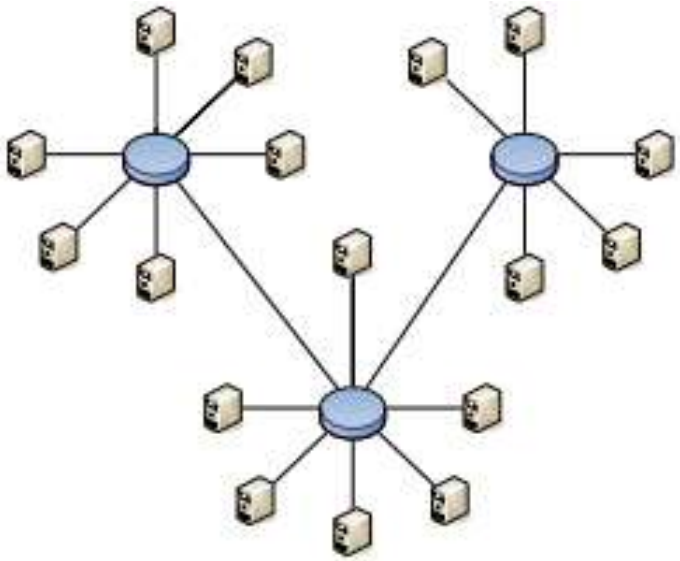
7 – PEER-TO-PEER – P2P PURO

- Una consulta es puesta en la red
- El requerimiento se propaga hasta que la información es descubierta o hasta que algún umbral de propagación es alcanzado.
- Si la información es localizada, el componente obtiene la dirección del otro componente y lo contacta



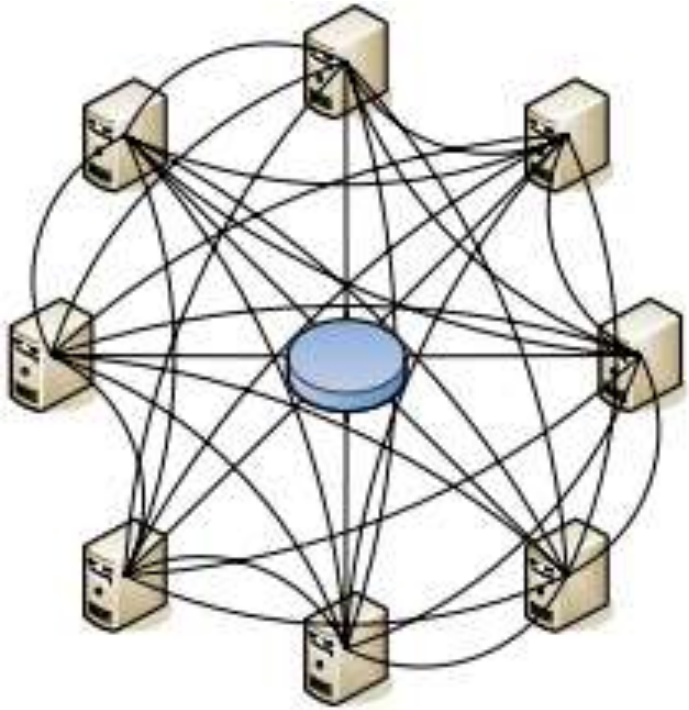
7 – PEER-TO-PEER – P2P HÍBRIDO

- Ciertos componentes juegan un rol especial, o bien localizando otros pares o proveyendo directorios para localizar información



7 – PEER-TO-PEER – SERVIDOR ÍNDICE

- Utiliza un servidor centralizado para indexar información y componentes.



7 – PEER-TO-PEER – PROS Y CONS

VENTAJAS	DESVENTAJAS
○ Escalabilidad: Los componentes pueden ser agregados o removido de la red sin un impacto significativo. ○ Disponibilidad: Si un componente deja de estar disponible, otros aun pueden proveer el servicio para completar la tarea. ○ Performance: La carga de cualquier componente actuando como servidor es reducida, ya que dicha carga es distribuida entre los componentes de la red.	○ Como un sistema peer-to-peer es descentralizado, algunas tareas son más complejas: manejar seguridad, consistencia de datos, disponibilidad de datos y servicios, backup y recuperación, etc. ○ Es difícil dar garantías porque los componentes van y vienen. Sin embargo, el arquitecto debe ofrecer probabilidades que las metas de calidad se cumplan, y que esas probabilidades serán mayores a medida que se incremente la población de componentes

PEER TO PEER – ANÁLISIS 1

DESCRIPCIÓN	Estado y comportamientos son distribuidos entre pares que pueden actuar tanto como clientes como servidores
COMPONENTES	Peers - componentes independientes que tienen su propio estado y control.
CONECTORES	Protocolos de red, generalmente propietarios.
ELEMENTOS DE DATOS	Mensajes de red.
TOPOLOGÍA	Red que puede tener conexiones redundantes entre peer Puede variar arbitraria y dinámicamente.

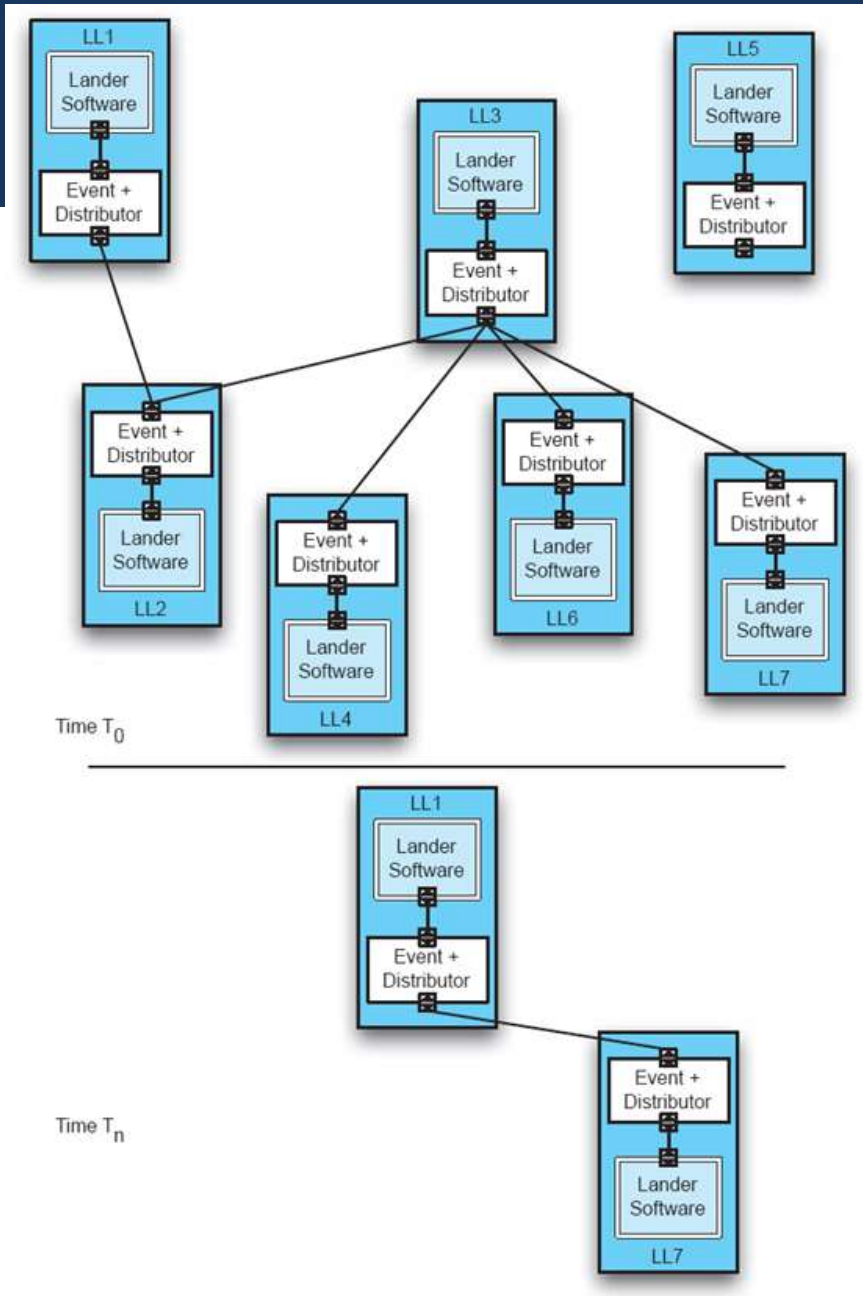
BASADO EN EVENTOS – ANÁLISIS 2

CUALIDADES	▪ Altamente robusto de cara a la falla de un nodo. ▪ Escalable, en términos de acceso a los recursos y poder de cómputo. ▪ Computación distribuida
USOS TÍPICOS	Cuando las fuentes de información y operación están distribuidas: ▪ File sharing ▪ Mensajería instantánea ▪ Grid computing
PRECAUCIONES	▪ Cuando la recuperación de la información es crítica en tiempo y no puede hacer frente a la latencia propuesta por el protocolo. ▪ Seguridad

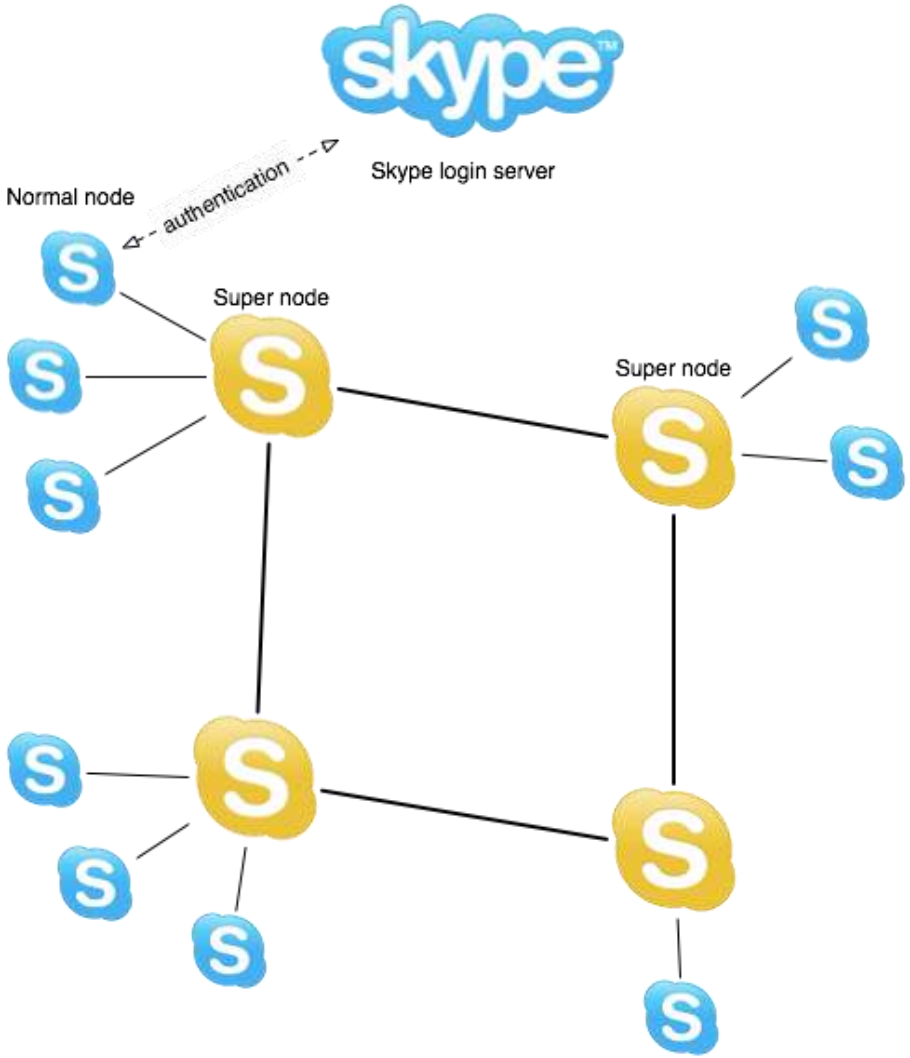
PEER-TO-PEER – DISEÑO

Cambios

- Un grupo de naves están intentando alunizar
- Una nave quiere saber si otra ya ha alunizado en un área especifica, para evitar la colisión.
- Cada nave está configurada para usa un protocolo P2P sobre la red.
- Un nave sólo podrá comunicarse con otras dentro de un radio limitado.



OTRO EJEMPLO - SKYPE



RESUMEN

ESTILOS SIMPLES

Estilos influenciados por lenguajes tradicionales
- Programa principal y subrutinas
- Orientación a objetos ✓

Por capas
- Maquinas virtuales
- Cliente-servidor ✓

Estilos de Flujo de Datos
- Batch sequencial
- Pipe and Filters ✓

Memoria compartida
- Blackboard
- Basado en reglas ✓

Intérprete
- Intepreter
- Mobile Code ✓

Invocación Implícita
- Publish-Subscriber
- Event Based ✓

Peer to Peer ✓

ESTILOS COMPLEJOS

- C2
- Objetos distribuidos

Ref: *Software Architecture, Foundations Theory and Practice*

RESUMEN – ESTILOS INFLUENCIADOS POR LENGUAJES

PROGRAMA PRINCIPAL Y SUBRUTINAS	
RESUMEN	El programa principal controla la ejecución del programa llamando a múltiples subrutinas
USARLO SI…	La aplicación es pequeña y simple
EVITARLO SI…	Se necesitan estructuras de datos complejas (por falta de encapsulamiento) Es probable que sucedan futuras modificaciones

ORIENTACIÓN A OBJETOS	
RESUMEN	Los objetos encapsulan el estado y acceden a funciones
USARLO SI…	El mapeo directo entre entidades externas y los objetos internos es sensible Existen muchas estructuras de datos complejas e interrelacionadas
EVITARLO SI…	La aplicación es distribuida en una red heterogénea Una fuerte independencia entre los componentes es necesaria Muy alta performance es requerida

RESUMEN – POR CAPAS

MÁQUINAS VIRTUALES	
RESUMEN	La máquina virtual, o una capa, ofrece servicios a las capas superiores
USARLO SI…	Muchas aplicaciones pueden basarse en una sola capa común de servicios. La implementación de una capa debe cambiar y se necesita una especificación de interface de servicio que sea resiliente
EVITARLO SI…	Se requieren muchos niveles (causa ineficiencia) Las estructuras de datos se deben acceder desde múltiples niveles

CLIENTE-SERVIDOR	
RESUMEN	Los clientes requieren servicios de un servidor
USARLO SI…	La centralización de cómputos y datos en una sola ubicación (el servidor) promueve la administración y la escalabilidad El procesamiento del usuario final es limitado a entrada de datos y presentación
EVITARLO SI…	La centralización presenta un solo punto de riesgo de falla El ancho de banda es limitado Las capacidades de la máquina cliente compiten con o exceden las del servidor

RESUMEN – DE FLUJOS DE DATOS

BATCH-SECUENCIAL	
RESUMEN	Programas separados, ejecutados secuencialmente con entradas por lotes
USARLO SI…	El problema es fácilmente formulado como un conjunto de varios pasos secuenciales
EVITARLO SI…	Interactividad o concurrencia entre los componentes es necesario o deseable Acceso random a datos es deseable

PIPE-AND-FILTER	
RESUMEN	Programas separados, llamados filtros, son ejecutados potencialmente de manera concurrente. Los “pipes” rutean los stream de datos entre los filtros.
USARLO SI…	(mismas condiciones que para Batch-Secuencial) .., y los filtros son útiles en mas de una aplicación Las estructuras de datos son fácilmente serializables
EVITARLO SI…	Se requiere interacción entre componentes

RESUMEN – MEMORIA COMPARTIDA

BLACKBOARD	
RESUMEN	Programas independientes, el acceso y la comunicación son exclusivamente a través del repositorio global, conocido como blackboard
USARLO SI…	Los cálculos se concentran en una estructura de datos común que cambia El orden de procesamiento es dirigido por los datos y determinado dinámicamente
EVITARLO SI…	Los programas tratan con partes independientes de los datos comunes La interface de los datos comunes es susceptible a cambiar Las interacciones entre los programas independientes requieren complejas regulaciones

BASADO EN REGLAS	
RESUMEN	Se usan hechos y reglas que se entran en la base de conocimiento para resolver una consulta
USARLO SI…	Los datos del programa y las consultas se pueden expresar como simples reglas sobre las cuales se pueden realizar inferencias
EVITARLO SI…	El número de reglas es extenso Se presentan interacciones entre las reglas Se requiere alta performance

RESUMEN – INTÉRPRETE

INTÉRPRETE	
RESUMEN	El intérprete parsea y ejecuta el stream de entrada, actualizando el estado mantenido por el intérprete
USARLO SI…	Se requiere un comportamiento altamente dinámico; y un alto grado de personalización del usuario final
EVITARLO SI…	Se requiere alta performance

CÓDIGO MÓVIL	
RESUMEN	El código es móvil, es decir, es ejecutado en un servidor remoto
USARLO SI…	Es más eficiente mover el procesamiento a un conjunto de datos que el conjunto de datos al procesamiento Es deseable personalizar dinámicamente el procesamiento de un nodo local a través de código externo
EVITARLO SI…	La seguridad del código móvil no puede ser asegurada Se requiere estricto control de las versiones del software desplegado

RESUMEN – INVOCACIÓN IMPLÍCITA

PUBLISH-SUBSCRIBE	
RESUMEN	Publishers desean hacer broadcast de mensajes a los suscriptores
USARLO SI…	Los componentes están muy débilmente acoplados Los datos de suscripción son pequeños y son eficientemente transportados
EVITARLO SI…	No se dispone de un middleware para soportar un alto volumen de datos

BASADO EN EVENTOS	
RESUMEN	Componentes independientes emiten y reciben eventos asincrónicamente comunicados a través de una red
USARLO SI…	Los componentes son concurrentes e independientes Los componentes son heterogéneos y distribuidos en red
EVITARLO SI…	Se requiere garantizar el procesamiento en tiempo real de los eventos

RESUMEN – PEER TO PEER

PEER-TO-PEER	
RESUMEN	Los “peers” mantienen estado y comportamiento y pueden actuar tanto como clientes como servidores
USARLO SI…	Los “peers” están distribuidos en una red, pueden ser heterogéneos y mutuamente independientes Se requiere robustez cuando pueden suceder fallos independientes y alta escalabilidad
EVITARLO SI…	La confianza de los “peers” independientes no puede ser asegurada o administrada No se dispone de nodos designados para dar soporte al descubrimiento de recursos

Elsa Estevez
ece@cs.uns.edu.ar