Arquitectura de Sistemas de Software

Departamento de Ciencia de la Computación
Escuela de Ingeniería – PUC
Hans Findel {hjfindel@uc.cl}

Introducción

¿Por qué este ramo?

- Aumenta la complejidad del software actual
  - Sistemas distribuídos, con escalabilidad masiva, sobre diversas plataformas, transparente para el usuario, ...
- Existe conocimiento de arquitecturas de referencia, frameworks y patrones para usar como base y mejorar la calidad del diseño
- Garantizar calidad, reducir costos, permitir flexibilidad

Motivación

- Típicamente los sistemas desarrollados atienden intereses puntuales
  - Lista de requisitos/requerimientos
  - Modelo de dominio
  - Desarrollo de software

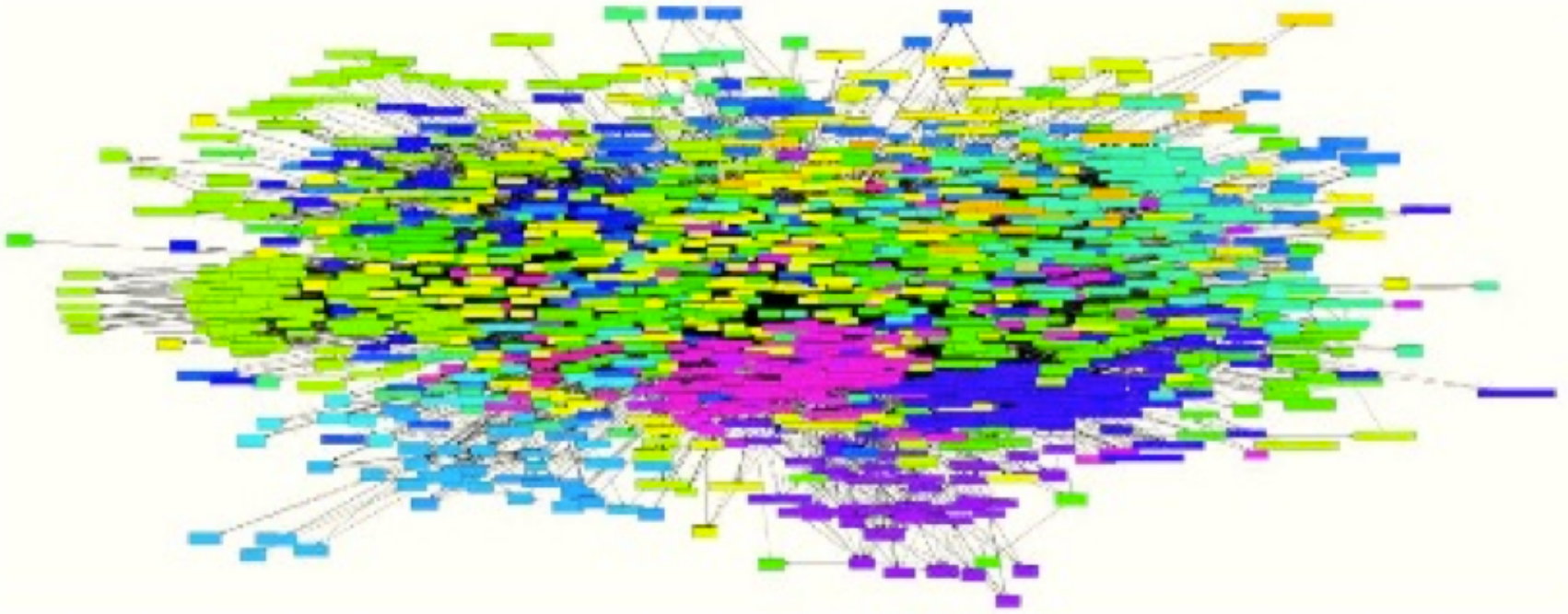
- Aparecen cambios!
  - Modificaciones, adaptaciones, extensiones, parches, …

- Resultado : Big ball of Mud

Big ball of mud

JDK 1.5

- 1315 classes in 229 packages all depend on each other



Problemas de la Ingeniería de Software

- Qué problemas existen en Ing. de Software?

  - Esenciales

  - Accidentales

Problemas de la Ingeniería de Software

- (Accidentales)
- Existe una solución
  - Hay que encontrar una solución “elegante”
- Mejora de productividad
  - Programación y abstracciones
    - Lenguajes de programación de alto nivel
  - Resultados de las decisiones de programación toma mucho tiempo
  - Programas heterogéneos
    - IDEs (Integrated Development Environment)
    - Apoyo al desarrollo

Problemas de la Ingeniería de Software

- (Esenciales)
- Solo existen soluciones parciales
  - Complejidad
    - Crece de manera no lineal
  - Conformidad
    - Sistema operativo, hardware, …
  - Cambiabilidad (adaptabilidad)
    - Nuevas aplicaciones, usuarios, máquinas, estándares, leyes, hardware, …
  - Intangibilidad
    - No hay leyes físicas, no hay una presentación obvia

Arquitectura de software

- Conjunto de las principales decisiones de diseño sobre el sistema
  - Estructura
  - Conducta
  - Interacción
  - Propiedades no funcionales

- Principal: Decisiones que afectan a todo el sistema

Similaridades y diferencias con Arquitectura (Ing. Civil)

- Requisitos, planos, construcción, uso
- Satisfacción del cliente
- Tareas especializadas
- Puntos de revisión
- El proceso no es tan importante como la arquitectur, pero
  - el software es más maleable que el material sólido
  - el software tiene un aspecto temporal y cambiará

Arquitectura de Software

- No se especifican detalles –de implementación– pero si los componentes que cargan con la responsabilidad de la implementación
  - Componentes de grano grande y subsistemas, no clases ni algoritmos
- Va del análisis a la implementación
- Objetivo: Manejar la complejidad y cumplir con los “atributos de calidad”
- Tomar decisiones que afectan a todo el sistema

Arquitectura de software

Arquitectura de software = {Elementos, *qué* Forma, *cómo* Lógica *por qué* }

- Involucra:
  - Elementos a partir de los cuáles se construye un sistema
  - Interacciones entre esos elementos
  - Patrones que guían su composición
  - Restricciones sobre esos patrones
  - Abstracciones, de-composición, composición, estilo y *estética*

Arquitectura de software

- “La organización fundamental de un sistema se ve reflejada en sus componentes, relaciones entre ellos y el ambiente, y los principios que gobiernan su diseño y evolución.”

Recommended Practice for Architectural Description of Software-Intensive Systems

ANSI/IEEE Std 1471-2000

- Captura la **estructura** del sistema en términos de componentes y cómo interactúan
- Define reglas de diseño y evolución globales al sistema
- Componentes, módulos, objetos o cualquier otra unidad de software interrelacionada

Principios

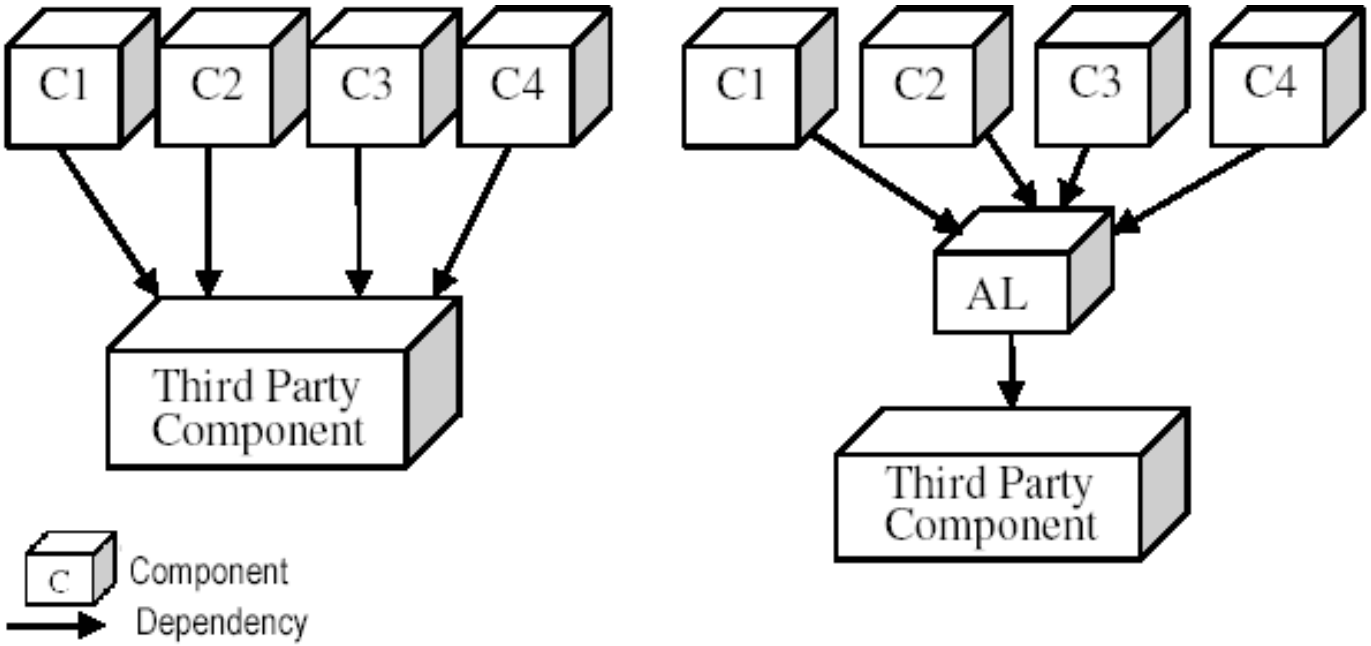
- Cohesión
- Acoplamiento
- Consistencia
- YAGNI (you aren’t goin to need it)
- KISS (keep it simple and stupid)
- Calidad
- Testing
- Diseño (frameworks, patrones, metáforas, … the big picture … pero sin funcionalidad innecesaria)
- etc…

Cohesión y acoplamiento

- Cohesión
  - La funcionalidad relacionada debe ir junta, dentro de un módulo
  - Las partes dentro de un módulo deben trabajar juntas
  - Cohesión débil?

- Acoplamiento
  - Interdependencia entre módulos
  - Acoplamiento débil ?

Capas de Abstracción (layers)



- Trade-offs
  - Cohesión y Dependencia.
  - Control y Comunicación de datos (latencia!).
- Patrones.

Roles del Arquitecto

- Juntar requisitos
- Documentar
- Diseñar
- Comunicar
- Liderar el desarrollo

Roles del Arquitecto

- Las mejores arquitecturas son producto de
  - Solo una "mente"
  - Un equipo pequeño y estructurado
    - Rechtin, Systems Architecting: Creating & Building Complex Systems, 1991, p21
- Todo proyecto debe tener exactamente 1 arquitecto identificable (1 responsable)
  - Para proyectos más grandes, el arquitecto principal debe/puede estar respaldado por un arquitecto por sub-equipo
    - Booch, Object Solutions, 1996

Habilidades esperadas

- Capacidad de desarrollo de software (experto)
- Expertise en el dominio (“experto del tema”)
- Comunicador
- Estratega
- Consultor
- Líder
- Tecnólogo
- Estimador de costo
- “Animador”/Motivador
- Político
- Vendedor

Arquitecto como desarrollador

- Debe entender las dificultades del desarrollo de software
  - Principios
  - Métodos y técnicas
  - Metodologías
  - Herramientas
- Debe entender las ramificaciones de las decisiones arquitectónicas
  - No debe vivir en una "torre de marfil"
  - Restricciones (de implementación) de sus decisiones

Arquitecto como “experto del tema”

- No basta con que sea (buen) programador
- Problemas con el dominio del tema
  - Madurez
  - Estabilidad
  - Usuarios y sistemas
- Puede afectar de gran manera a la arquitectura diseñada y la implementada de la solución
  - Capacidad de escalar
  - Capacidad de evolucionar
  - Capacidad de …
- Requiere de artefactos para entender el problema
  - No la solución (al menos no en este rol)

Arquitecto como estratega

- Desarrollar una arquitectura "elegante" no es suficiente
  - La tecnología es solo una parte de la solución
  - La arquitectura debe ser adecuada para la organización
- Debe ajustarse a la organización
  - Estrategia de Negocios
    - Al "rationale" detrás de la misma
  - A las prácticas del negocio
  - A los ciclos de planificación
  - Debe ajustarse (o funcionar bien) con los procesos del negocio
- Debe estar consiente de la competencia
  - Productos
  - Estrategias
  - Procesos

Arquitecto como comunicador

- Es casi la mitad del trabajo
- Debe
  - Escuchar las preocupaciones de los Stakeholders
  - Explicar la arquitectura de la solución
  - Negociar compromisos y fechas
- Necesita capacidades para
  - Escribir bien
  - Hablar (convencer)
  - Presentar

Arquitecto como comunicador

- Managers
  - Para entregar mensajes importantes
    - La arquitectura es útil e importante
    - Asegurar el apoyo a lo largo del proyecto
  - Debe escuchar las preocupaciones acerca de
    - Costos
    - Fechas / deadlines
- Desarrolladores
  - Convencerlos de la efectividad (de la arquitectura)
  - Justificar decisiones sub-óptimas (locales)
  - Responder a problemas con
    - Herramientas
    - Métodos
    - Desiciones de diseño
- Otros arquitectos de software
  - Asegurar y mantener la consistencia del lenguaje
  - Asegurar las propiedades (deseadas) y capacidad de evolución del sistema

Arquitecto como comunicador

- Ingenieros de sistema
  - Coordinar requerimientos y soluciones
  - Explicar cómo se enfrentan los principales problemas
- Clientes (como en Cliente / Servidor)
  - Determinar las necesidades
  - Explicar cómo la arquitectura responde a dichas necesidades
- Usuarios
  - Determinar las necesidades
  - Explicar cómo la arquitectura responde a dichas necesidades
  - Escuchar los problemas
- "Área de marketing"
  - Ayudar a obtener y determinar los objetivos y directrices
  - Explicar cómo la arquitectura responde a dichos objetivos

Arquitecto como líder

- Debe ser un líder técnico
  - En base a su conocimiento y sus logros
  - Inspirar respeto a través de sus ideas, expertise, palabras y acciones
  - No puede apoyarse solo en su posición en el organigrama
- Asegurar que se sigan las reglas, guías y decisiones de diseño
- Mejorar la productividad y calidad al agregar/incorporar
  - Nuevas ideas, soluciones y técnicas
  - Mentores y nuevos personas al proyecto
- Tomar decisiones y ayudar en asegurar la implementación

Arquitecto como tecnólogo

- Entender los enfoques de desarrollo
  - Basado en objetos y en componentes
- Entender las tecnologías fundamentales
  - Redes y sistemas operativos
  - Middleware
  - Seguridad
  - Bases de Datos
  - Interfaces gráficas (GUI)
- Estar al día con las corrientes
  - E.g., CORBA, COM/DCOM, JavaBeans, UML, XML
- Demostrar expertise en
  - Modelar el sistema
  - Análisis de trade-off de la arquitectura
  - Amarrar requerimientos de sistema a la solución arquitectónica

Arquitecto como Estimador de Costos

- Entender las ramificaciones financieras de las decisiones arquitectónica
  - Green-field vs. Brown-field
  - Costo de adoptar COTS
  - Costo de desarrollar para el reuso
  - Estabilidad financiera y posición de la compañía en el rubro
- La solución tecnológica "superior" no es siempre la más apropiada
  - Impacto en el costo y el cronograma
- Aproximaciones de estimaciones de costos son suficientes (en general)
  - Se puede entrar en mayor detalle cuando las opciones se han reducido a un pequeño grupo

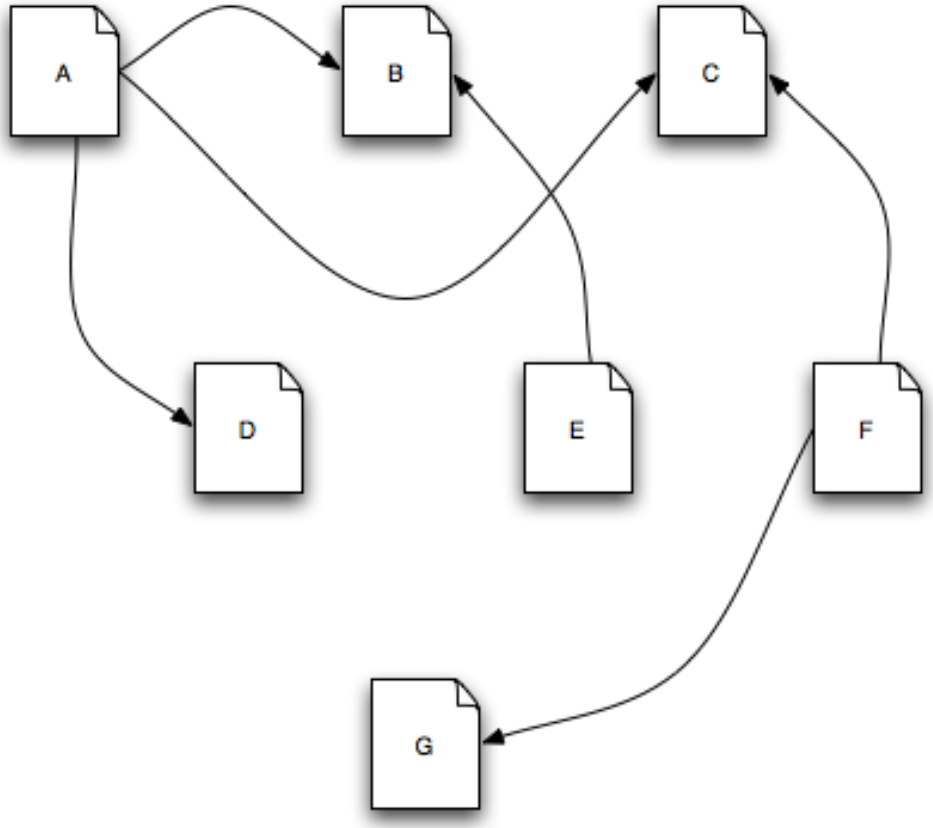
Arquitecto como vendedor

- Por las razones anteriores, el arquitecto debe vender:
  - La visión general
  - La solución tecnológica
  - Principales propiedades arquitectónicas
  - Principales propiedades del sistema que se obtendrá
  - Perfil de costos y cronograma
  - Importancia de apegarse a la arquitectura
    - "Costos" de desviarse de la arquitectura

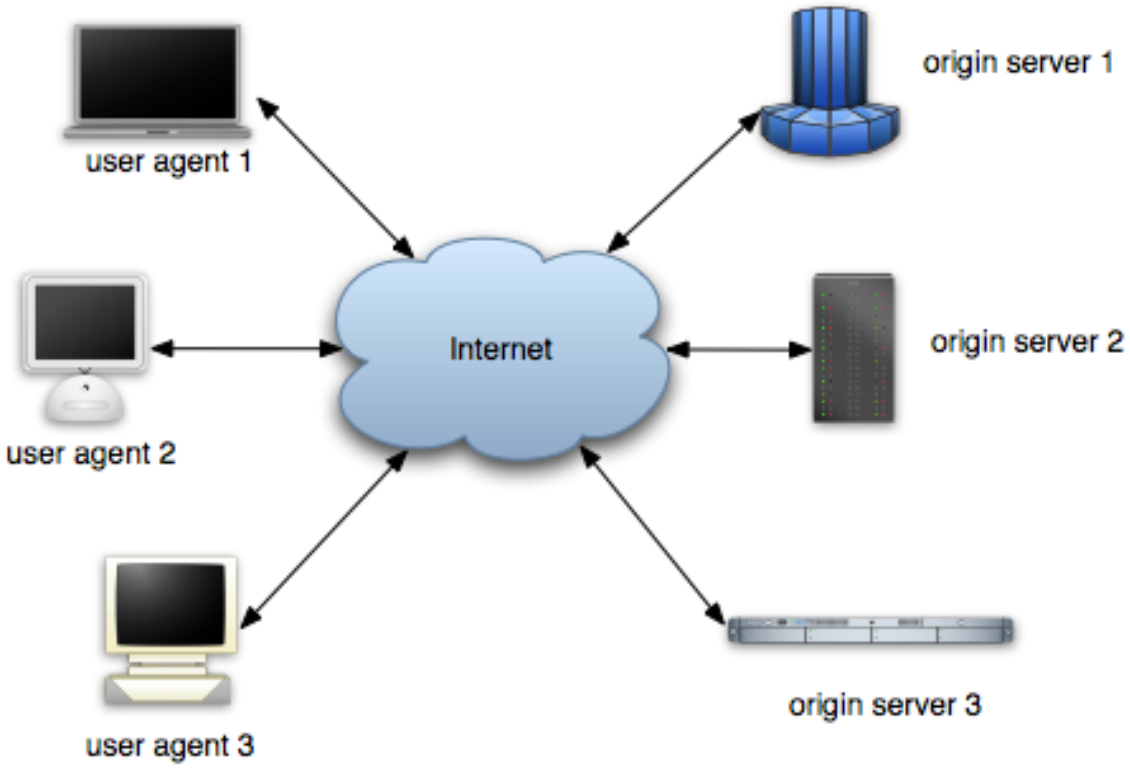
Rol del equipo de arquitectura

- Definir la arquitectura de software
- Mantener la integridad de la arquitectura de software
- Explorar y mitigar los riesgos asociados al diseño
- Proponer orden y contenidos de las iteraciones
- Coordinar y co-existir con otros equipos
- Asistir en las decisiones del proyecto
- Asistir en las definiciones del producto

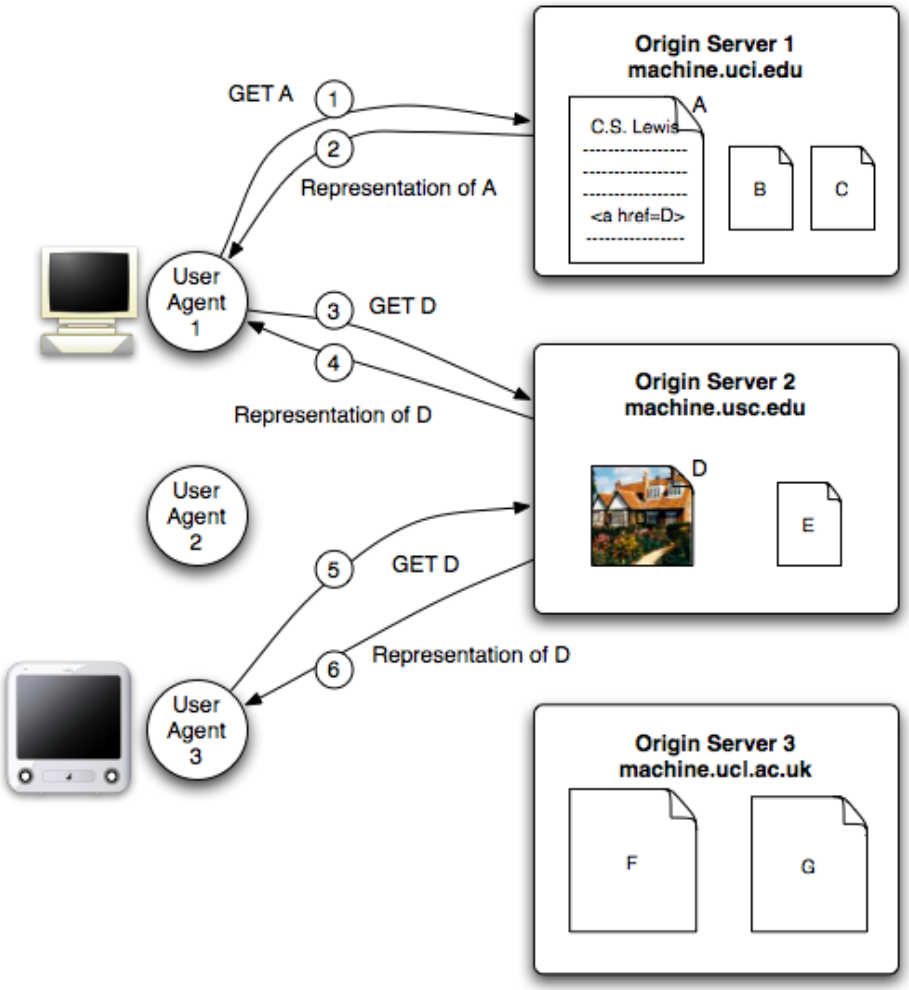
Ejemplo - Comunicando la Arquitectura de la WWW



Ejemplo - Comunicando la Arquitectura de la WWW



Ejemplo - Comunicando la Arquitectura de la WWW



Proyecto

- Construir un prototipo siguiendo un estilo arquitectónico predeterminado

- Entrega 1 : Diseño (NFR, Patrones, UML 2.0, Tecnología)
- Entrega 2 : Diseño + Desarrollo inicial + Deploy (parcial)
- Entrega 3 : Funcionalidad completa + Deploy
- Entrega 4 : Deploy + Desarrollo + Testing + Presentación

Evaluación

- Ayudantías no obligatorias
- Tareas obligatorias
- Teoría (NT)
  - 2 Interrogaciones
  - (NE) Examen final (obligatorio y reprobatorio) (20%)
- Práctica (NP)
  - Entregas individuales
  - Todos aplican todas las tecnologías (rotación)
  - NO MYSQL, NO PHP, NO RoR
- NE ≥ 4.0, NP ≥ 4.0, NT ≥ 4.0

Bibliografía

- Richard N. Taylor, “Software Architecture, Foundations, Theory and Practice”, Wiley, 2010.
- Spinellis & Gousios, “Beautiful Architecture”, O’Reilly, 2009.
- Ian Gorton, “Essential Software Architecture”, Springer, June, 2006.
- Martin Fowler et al., “Patterns of Enterprise Application Architecture”, Addison Wesley, 2002.

Fechas

- I1: Jueves 26 Septiembre
- I2: Jueves 17 Octubre

- Examen: Martes Noviembre/Diciembre ?, 09:00