UNIVERSIDAD TECNICA
FEDERICO SANTA MARIA

Departamento de Informática
Universidad Técnica Federico Santa María

Ingeniería de Software

Arquitectura de Software

Cristian Orellana (por Hernán Astudillo) & Gastón Márquez
Departamento de Informática
Universidad Técnica Federico Santa María
<hernan@inf.utfsm.cl, gaston.marquez@sansano.usm.cl>

Motivación

Ejemplo: i-Banco Casero

- Problema (conocido)
  - Operaciones bancarias desde la casa
  - Vía internet (Web)
  - Base de datos externa conocida
- Caso de uso
  - Consultar saldo, usando usr/pwd & registrando en log

Motivación - Fase 0

Fase 0 – Análisis

- Solución simple
  - Cliente: browser Web
  - Servidor: scripts CGI (programas invocados según URL)
  - Comunicación vía HTTP
  - Seguridad: usuario y contraseña, vía formas HTML
  - Auditable: bitácora (“logs”)
  - Datos: BD detrás del servidor Web

Fase 0 – Encuesta

- Encuesta
  - Número de clases
  - Esfuerzo de construcción

Fase 0 – Problema

- Problema tecnológico
  - Operaciones transaccionales (atómicas pero compuestas)
  - ...pero HTTP es “stateless”: cada invocación parte de cero
- Solución
  - ‘Sesiones’: pasar estado, o guardar estado y pasar clave; ambos casos pueden usar URL, forma, cookie...

Descripción de arquitecturas

- Notaciones similares a diseño (p.ej. UML, '4+1')
  - También existen notaciones formales (ADLs)
- Vistas del sistema a construir
  - Vistas estáticas (clases, asociaciones, capas...)
  - Vistas dinámicas (estados, secuencia, colaboración...)
  - Vistas funcionales (casos de uso, interfaces...)
- Vistas del proceso a seguir
  - Subsistemas y componentes (unidades de trabajo)
  - Dependencias entre unidades (para planificar; Gantt/Pert...)
- Vistas son definidas por tipo de 'lector'
  - *¿Quién lee especificaciones de arquitectura?*
    - Mejor dicho, *¿A quién le importa lo que hace el arquitecto?*
- 'Stakeholders': los afectados (lectores y evaluadores)

Motivación - Fase 1

Fase 1: Masificación

- Problema (conocido)
  - El sistema se vuelve popular
  - Muchos usuarios
- Problemas sistémicos
  - Lento e inestable
    - CGI levanta procesos -> memoria y velocidad
  - Colisiones en bitácoras y datos
    - Problemas de concurrencia
  - Difícil de monitorear
- Requisitos
  - Mismo caso de uso
  - 1000 usuarios

Fase 1 – Encuesta

- Estimaciones
  - Número de clases
  - Esfuerzo de construcción

Fase 1 – Análisis

- Soluciones (complementarias)
  - Procesos residentes en memoria (por sesión)
  - Control de concurrencia (pesimista, probablemente)
  - Monitoreo dinámico y global
  - Replicación de servidores (balanceo de carga)
  - Caché de datos (read-only)

Problemas “sistémicos”

- Mayor escala provoca problemas sistémicos
  - Escalabilidad, rendimiento, disponibilidad (estabilidad), confiabilidad, seguridad...
  - No puede identificarse un lugar específico ‘culpable’
  - No son defectos de programación, sino malas decisiones sobre protocolos, políticas de replicación, concurrencia...
- Impacto de requisitos
  - El arquitecto privilegia las propiedades sistémicas por sobre la “funcionalidad” (tareas)
  - En general, con sistemas grandes es más difícil satisfacer las propiedades de ejecución que ejecutar la tarea misma
- Errores de arquitectura (o diseño) son caros
- Foco del arquitecto
  - Requisitos sistémicos (descripción y solución)

Motivación - Fase 2

Fase 2: PyMEs / sucursales

- Problema (conocido)
  - El sistema se expande a usuarios corporativos
    - Se recibe depósitos del empleador a empleados
- El sistema atrae usuarios internos
  - Usuarios conectados vía intranet, rápida, segura, con plataforma uniforme
  - Problemas de responsabilidad y acceso
- Requisitos
  - Casos de uso: (1) ver balance, (2) depositar sueldo

Fase 2 – Encuesta

- Estimaciones (usando una arquitectura pre-empaquetada)
  - Número de clases
  - Esfuerzo de construcción

Fase 2 – Análisis

- Notas
  - Usuarios: personas y organizaciones (con representantes)
  - Seguridad: separar autenticación y autorización
    - Autenticación: certificados (extranet) y login (intranet)
    - Autorización: modelo de ‘apoderado’(s) por empresa
- Problema
  - Aspectos especializados del dominio exigen modelos más ricos
  - Estos modelos requieren administración (como parte del sistema)
- Soluciones
  - Autorización
    - Modelo: roles y derechos
    - Implantación: directorios centralizados (‘manejo de identidad’)

Arquitecturas sistematizadas

- Problemas recurrentes, usualmente coexistentes
- Pre-empaquetar tecnologías
  - Sesiones, control de concurrencia, monitoreo...
- 'Arquitecturas de línea de productos'
  - Factorización y reuso intra-organización
- 'Arquitecturas de referencia' para aplicaciones
  - 'Servlets': objetos-procesos (esquema MVC)
  - ASP (Active Server Pages): appsWeb
- 'Arquitecturas de referencia'
  - CORBA (objetos distribuidos heterogéneos; casi S.O.)
  - RM-ODP (Open Distributed Processing-Reference Model)

Motivación - Fase 3

Fase 3: Integración

- Problema (grande al fin)
  - El banco decide incorporar pago de cuentas y transferencias
    - Por usuarios individuales y/o corporativos
- Transacciones con efectos en terceros
  - No-repudiables
  - Mal uso puede afectar al banco, no sólo a los usuarios
- Conexión con sistemas legado
  - Antiguos, pero funcionando (y bien)
  - Plataformas heterogéneas

Fase 3 – Solución

- *Modelar in situ*

Fase 3 – Notas

- Problemas de 'integración'
  - Integridad de datos: transacciones coordinadas y/o distribuidas
  - Compatibilidad de formatos, plataformas, protocolos...
  - Disponibilidad: tolerancia a fallas
  - En general: sistemas y políticas heterogéneos
- Posibles soluciones
  - Construcción ad-hoc (p.ej. 'heartbeat', 'logs'...)
  - Servicios Web (XML, SOAP, UDDI...)
  - 'Message brokers': mensajes+filas confiables (ej: IBM MQ)
  - Sistema operativo (o BD) distribuido (p.ej. Oracle)
  - Combinación de elementos y/o productos

Temas a considerar y reflexión

Generando arquitecturas [1]

- Generación de arquitecturas alternativas
  - Identificar políticas (propiedades de alto nivel)
  - Determinar combinaciones de mecanismos suficientes
- Dimensiones y valores (+/- ortogonales)
  - Integridad: vía transacciones
  - Tolerancia a falla: vía replicación
  - Comunicación: vía enlaces (modo, medio, formato, meta...)

Generando arquitecturas [2]

- Dimensiones y valores (+/- ortogonales)
  - Integridad: vía transacciones
    - Semánticas: nivel de serialización, anidamiento, sagas...
    - Concurrencia: lock, timestamp, file system, BD...
  - Tolerancia a falla: vía replicación
    - Del sistema: redundancia activa, 'failover' (master)...
    - Del estado: propagado dinámicamente, periódicamente, 'cacheado'...
  - Comunicación: vía enlaces (modo, medio, formato, meta...)
    - Modo: push, pull...
    - Medio: punto-punto/multicast/broadcast, síncrona/asíncrona, mensajes vs. 'streams'...

Evaluando arquitecturas

- Implantando arquitecturas
  - Solución H0: programar (flexible/ameno vs. costo/tiempo)
  - Tecnologías con capacidades suficientes (J2EE, .NET...)
  - Productos: comercial vs. Open Source (riesgo vs. costo)
- Problema
  - Evaluar arquitecturas & comparar alternativas
- *¿Qué significa 'mejor'?*
  - Criterios de optimalidad: efectividad, costo, simplicidad, futura integración o crecimiento, marketing...
  - Sólo los 'stakeholders' pueden realmente escoger (no son decisiones técnicas)
- *Una buena arquitectura es una buena descripción de una buena solución*
  - "Buena": responde las preguntas de los 'stakeholders'

Reflexión

- ¿Qué hemos hecho?
- Introducción paulatina de fuentes de complejidad
  - Madurez tecnológica
  - Escala (tamaño)
  - Complejidad del dominio
  - Heterogeneidad
- Introducción paulatina de temas de arquitectura
  - Descripción de arquitecturas
  - Problemas sistémicos
  - Arquitecturas sistematizadas
  - Evaluación y comparación
- Enfoque a vuelo de pájaro
  - Sin notación detallada
  - Sin detalles técnicos (de problemas ni soluciones)

Contexto

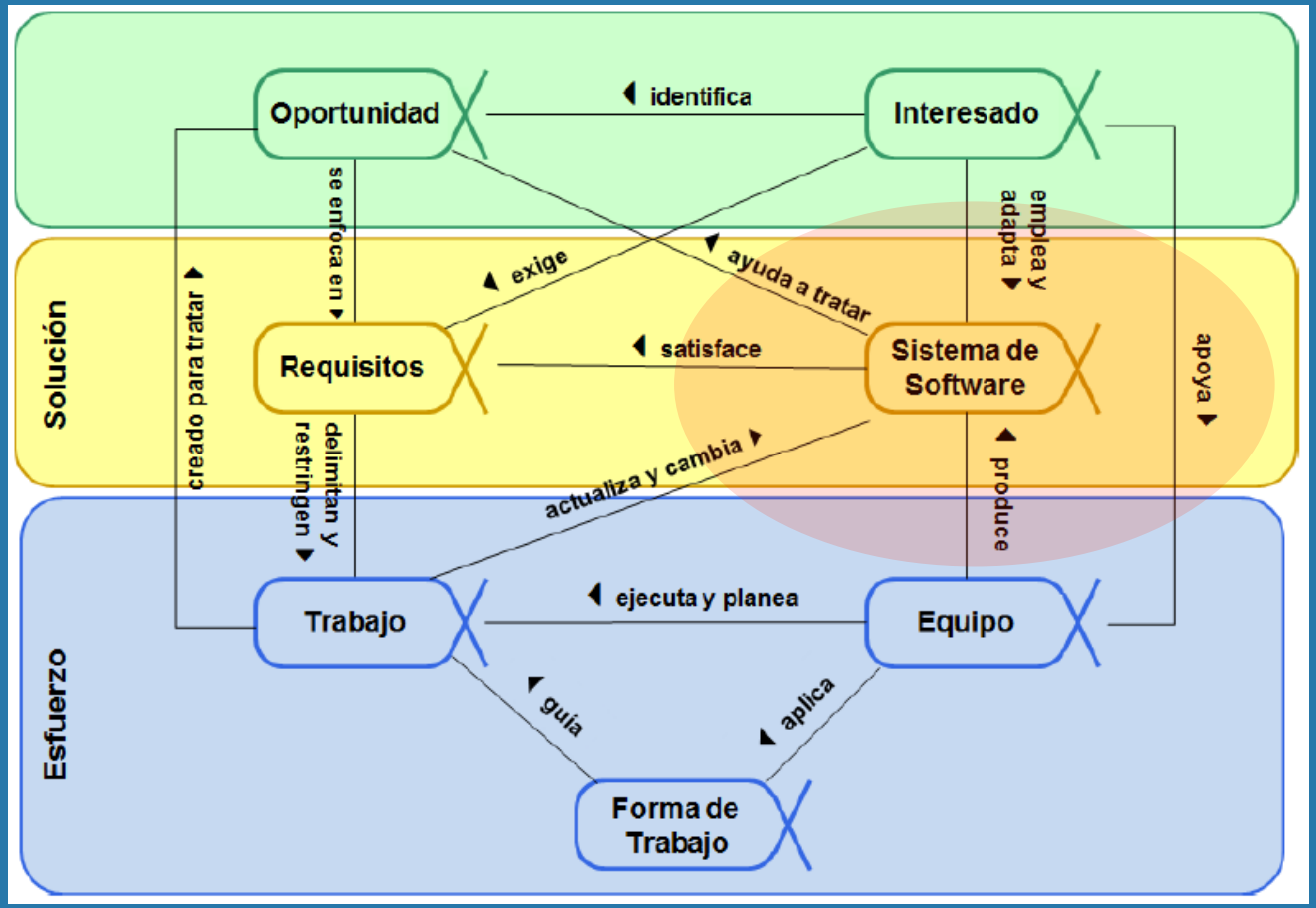
Contexto [1]

- Arquitectura
  - Antigua disciplina de la ingeniería
- Software
  - Nuevo paradigma para la sociedad
- Típica excusa de la comunidad
  - La industria de software joven y única
- Aún así, la economía del mundo se basa en productos de software

Contexto [2]

- ¿Hacia a dónde apunta la Arquitectura de Software?
  - *La complejidad del software aumenta*
    - Desarrollo de Software: miles de personas por años
    - Muchos sistemas de software
  - *Diseñando software*
    - Más allá de algoritmos/estructura de datos
    - Nuevo tipo de problema: ***estructura general del sistema***

Contexto [4]


Oportunidad
identifica
Interesado
se enfoca en
exige
ayuda a tratar
empleа y adapta
Solución
creado para tratar
Requisitos
satisface
Sistema de Software
apoya
delimitan y restringen
actualiza y cambia
produce
Esfuerzo
Trabajo
ejecuta y planea
Equipo
guía
aplica
Forma de Trabajo

Arquitectura de Software

Produciendo Sistemas de Software

- El criterio ha cambiado
  - La necesidad de aplicaciones de software ha aumentado
  - ¿Cómo especificar requisitos para nuevos productos e implementar software rápido y barato?
  - Productos de software lanzados muy rápido al mercado
    - ¿Qué ocurre con la calidad?
  - Nuevo criterio→¿Tiene una buena Arquitectura de Software, entendida por los stakeholders y desarrolladores?

¿Qué es Arquitectura de Software? [1]

- Según [Bass et.al], *Software Architecture is the structure or structures of the system, which comprise software elements, the externally visible properties of those elements, and the relationships among them.*
- La Arquitectura de Software (AS) define elementos
  - Que tienen propiedades externas visibles. Por ejemplo: servicios, características de rendimiento, control de fallos, recursos compartidos, otros.
  - Relaciones entre elementos
  - Estructuras

¿Qué es Arquitectura de Software? [2]

- La AS puede tener más de una estructura
  - Arquitectura de Software no es una sola estructura
- Cada sistema de software tiene una arquitectura
- El comportamiento de los elementos es parte de la AS
- Pero, la definición anterior no menciona la calidad de AS

Arquitectura de Software

1. Provee un plan de diseño de un sistema
2. Es una abstracción que ayuda a administrar la complejidad de un sistema

- AS está limitada por
  - La falta de formas estandarizadas para representar arquitecturas
  - La falta de métodos analíticos para predecir arquitecturas que resultan en una implementación a partir de requerimientos

AS como un plan de diseño

- Es un plan estructural que describe
  - Los elementos de un sistema
  - Cómo van a rendir juntos
  - Cómo van a trabajar juntos para satisfacer los requerimientos
- Se usan bosquejos durante el proceso de desarrollo
- Se establecen las expectativas del
  - Cliente
  - Personal administrativo

AS como abstracción

- AS no es una descomposición completa del sistema
- AS debe describir los distintos niveles de granularidad
  - ¿Cómo los elementos satisfacen las necesidades de los requisitos?
  - Interacciones entre los elementos
  - Dependencias de los elementos en la plataforma de ejecución
- Resolver los *trade off* puede liderar
  - Sacrificar algunas cualidades deseadas
  - Comprometer algunos requerimientos

Conceptos de AS [1]

- **Patrón de Arquitectura**
  - Define tipos de elementos y cómo interactúan
  - Algunas veces definen el mapeo de la funcionalidades a elementos de arquitectura
- **Dominio de una arquitectura**
  - Define tipos de elementos y cómo interactúan
  - Se aplican a un dominio en particular
  - Se definen en cómo la funcionalidad del dominio es mapeada a elementos de arquitectura

Conceptos de AS [2]

- **Product-line architecture**
  - Aplicar para un conjunto de productos de una compañía
  - Define tipos de elementos, cómo ellos interactúan, como la funcionalidad del productos es mapeada a ellos
  - También definen algunas de las instancias de los elementos de arquitectura
    - Por ejemplo, componentes *error-reporting* debería ser común a muchos productos de la línea de productos

Conceptos de AS [3]

- **Arquitectura de Software**
  - Se aplica a un sistema
  - Describe tipos de elementos, cómo interactúan, cómo la funcionalidad del producto es mapeada a él
  - Describe las instancias que existen en el sistema
  - Detalla el nivel de especificación necesitado para diseñar un sistema

Conceptos de AS [4]

- **ADL (Architectural Description Language)**
  - Entrega la notación para elementos de arquitectura
  - Comúnmente utilizado en la comunidad de investigación
  - En teoría, los arquitectos pueden usar ADL para representar las definiciones explicadas anteriormente
  - En la práctica, no todos los ADL pueden representar las defincciones
  - Algunas herramientas soportan ADL
    - ABACUS
    - ACME/ADML
    - ByADL
    - Rapide

Arquitectura de Software y atributos de calidad

Arquitectura y atributos de calidad

- Los atributos de calidad se consideran en
  - Diseño, implementación, desarrollo
- La arquitectura es crítica para satisfacer muchos atributos de calidad
  - Estos atributos son diseñados y evaluados en cada nivel de arquitectura
- La arquitectura no alcanza estos atributos
  - Son alcanzados mediante detalles (implementación)

Atributos de calidad

- Atributos del sistema
  - Availability, modifiability, performance, security, testability, usability
- Atributos del negocio
  - Time-to-market, otros
- Atributos de arquitectura
  - Integridad conceptual, otros

Escenarios de los atributos de calidad [1]

- Un requerimiento de atributo de calidad contiene:
  1. Fuente de estímulo (*Source of stimulus*)
  2. Estímulo (*Stimulus*)
  3. Ambiente (*Enviroment*)
  4. Artefacto (*Artifact*)
  5. Respuesta (*Response*)
  6. Medición de la respuesta (*Response measure*)

Escenarios de los atributos de calidad [2]

- *Fuente de estímulo*
  - Humanos, computadores, sistemas generadores de estímulos
- *Estímulo*
  - Condición que necesita ser evaluada cuando llega al sistema
- *Ambiente*
  - Las condiciones en que el sistema está (sobrecargado, ejecutándose, otros)

Escenarios de los atributos de calidad [3]

- *Artefacto*
  - Algo que es estimulado (el sistema entero o parte del sistema)
- *Respuesta*
  - Actividad tomada una vez que el estímulo llega
- *Medición de la respuesta*

  Respuesta que debe ser medible de alguna manera para que el requerimiento pueda ser testeado

Availability [1]

- Relacionado con la usabilidad
- Se preocupa de las fallas del sistema y de sus consecuencias
- Preocupaciones
  - ¿Cómo el sistema detecta una falla?
  - ¿Con cuanta frecuencia el sistema falla?
  - ¿Qué ocurre cuando el sistema falla?
  - ¿Cuánto tiempo deberá estar el sistema no operativo?
  - Otros

Availability [2]

Source	Internal/external to system
Stimulus	Fault: omission, crash, timing, response
Artifact	System's processors, communication channels, persistent storage, processes
Environment	Normal operation or degraded mode
Response	Detect event and record it/notify appropriate parties/disable event sources causing faults/ failures/ be unavailable for an interval/ continue
Response measure	Time interval of available system, availability time, time interval of degraded mode, repair time

Modifiability [1]

- Relacionado con el costo del cambio
  - ¿Qué puedo cambiar del artefacto?
  - ¿Quién o quienes pueden hacer el cambio?
- ¿Qué puedo cambiar?
  - Aspecto del sistema: agregar, borrar, modificar
  - Funcionalidades del sistema
  - La plataforma
  - El ambiente del sistema
  - Las cualidades del sistema
  - Capacidad del sistema
  - Otros

Modifiability [2]

Source	End user, developer, system administrator
Stimulus	Wishes to add/delete/modify/vary functionality, QA, capacity, etc
Artifact	System UI, platform, environment, system that interoperates with target system
Environment	Runtime, compile time, build time, design time
Response	Locates place in architecture to modify, makes modification w/o affecting other func., tests modif., deploys modif.
Response measure	Costs in terms of number of elements affected, effort, money; extent to which this affects other QAs, functions

Performance [1]

- Relacionado con el tiempo
  - ¿Cuánto tiempo se toma el sistema para responder cuando un evento ocurre?
- Eventos
  - Interrupciones, mensajes, peticiones de usuarios
- Eventuales problemas
  - ¿Qué ocurre si existe un gran numero de eventos? (Black Friday, por ejemplo)

Performance [2]

Source	Independent sources (possibly from within system)
Stimulus	Periodic or stochastic or sporadic events occur
Artifact	System
Environment	Normal mode, overload mode
Response	Processes stimuli; changes level of service
Response measure	Latency, deadline, throughput, jitter, miss rate, data loss

Security [1]

- Mide la capacidad del sistema
  - De resistir usos no autorizados
  - Proveer servicios autorizados a usuarios legítimos
- Ataque: intento de quebrar la seguridad
  - Intento no autorizado para acceder a datos o servicios
  - Intento no autorizado para modificar
  - Intento para denegar servicios a usuarios legítimos

Security [2]

Source	Individual/system: identity, internal/external, authorization, access
Stimulus	Try to: display data, change/delete data, access system services, reduce availability
Artifact	System services, data within system
Environment	On/offline, (dis)connected, firewalled or open
Response	various
Response measure	various

Testability [1]

- Facilidad en donde un software puede ser demostrado sus faltas mediante testing
- Probabilidad en que el software fallará en la siguiente etapa de ejecución
- Mediciones de respuestas
  - Efectividad del test
  - Cuanto la satisfactibilidad del test anterior

Testability [2]

Source	Unit developer, increment integrator, system verifier, client acceptance tester, system user
Stimulus	Analysis, architecture, design, class, subsystem integration completed, system delivered
Artifact	Design part, code part, complete application
Environment	At design time, at development time, at compile time, at deployment time
Response	Provides access to state values; provides computed values; prepares test environment
Response measure	Percent executable statements executed, probability of failure if fault exists, time to perform tests, length of longest dependency chain in a test, length of time to prepare test environment

Usability [1]

- Preocupado de
  - Cuan fácil es para el usuario realizar sus tareas
  - Soporte de usuario
- Los problemas de usabilidad generalmente son descubiertos durante el prototipo (presentación del Entregable I ☺) y las pruebas finales
- Si los problemas de usabilidad son descubiertos tarde, más caro sale repararlos

Usability [2]

Source	End user
Stimulus	Wants to learn system features, use system efficiently, minimize impact of errors, adapt system, feel comfortable
Artifact	System
Environment	At runtime and configure time
Response	Various
Response measure	Task time, number of errors, number of problems solved, user satisfaction, user knowledge gain, ratio of successful operations to total operations, amaount of time/data lost

Patrones y tacticas de Arquitectura

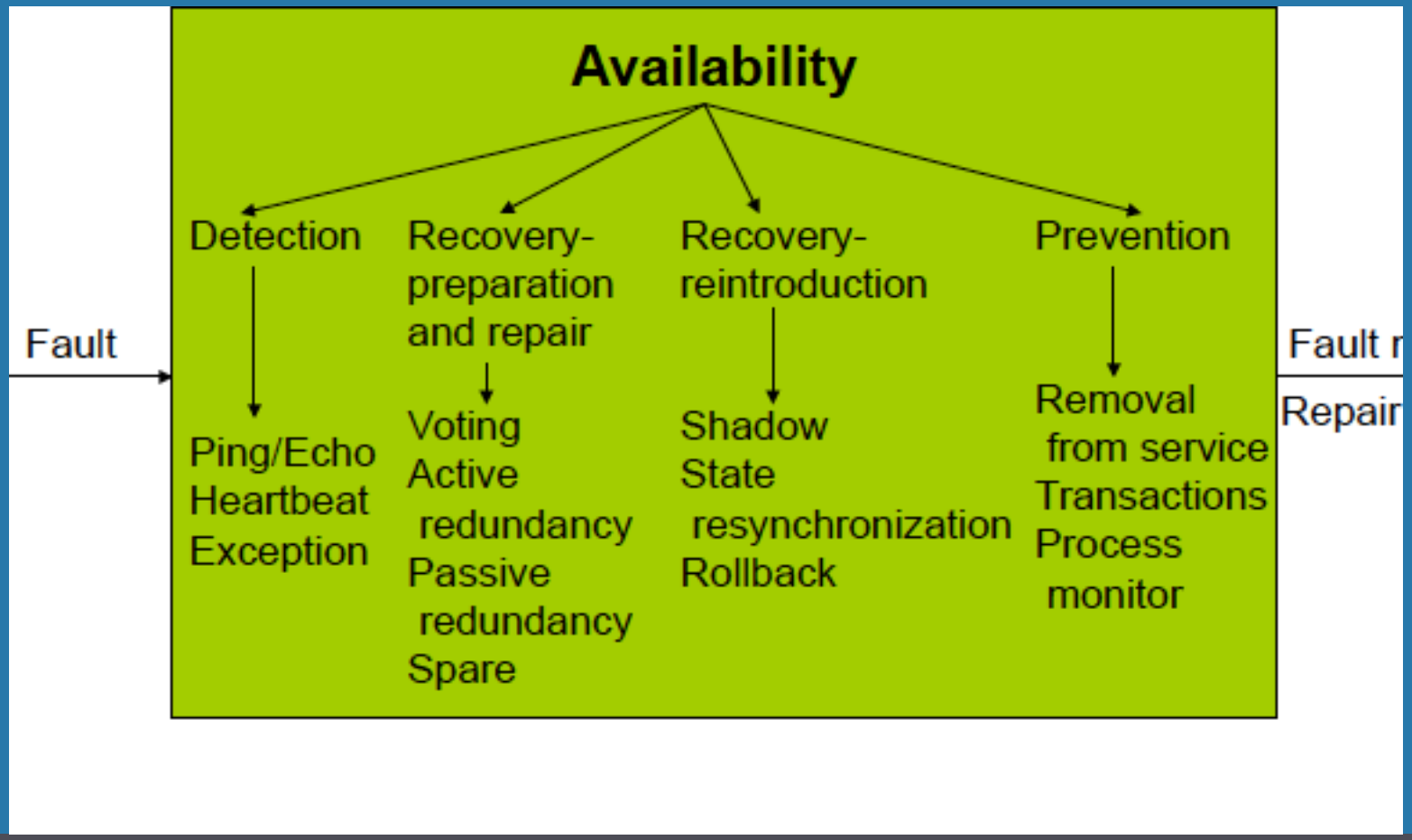
Tácticas

- Cualidades que son obtenidas a través de las decisiones de diseño
- ¿Qué decisiones de diseño son necesarias para obtener una cualidad específica?
- Tácticas
  - Decisiones de diseño que influencian el control de los atributos de calidad
- Diseño del sistema= colección de decisiones de diseño
  - Decisiones de diseño
    - Asegura el logro de algunos sistemas
    - Ayuda al control de QA

Availability [1]

Portion of Scenario	Possible Values
Source	Internal/external: people, hardware, software, physical infrastructure, physical environment
Stimulus	Fault: omission, crash, incorrect timing, incorrect response
Artifact	System's processors, communication channels, persistent storage, processes
Environment	Normal operation, startup, shutdown, repair mode, degraded operation, overloaded operation
Response	Prevent the fault from becoming a failure Detect the fault: • log the fault • notify appropriate entities (people or systems) Recover from the fault • disable source of events causing the fault • be temporarily unavailable while repair is being effected • fix or mask the fault/failure or contain the damage it causes • operate in a degraded mode while repair is being effected
Response Measure	Time or time interval when the system must be available Availability percentage (e.g. 99.999%) Time to detect the fault Time to repair the fault Time or time interval in which system can be in degraded mode Proportion (e.g., 99%) or rate (e.g., up to 100 per second) of a certain class of faults that the system prevents, or handles without failing

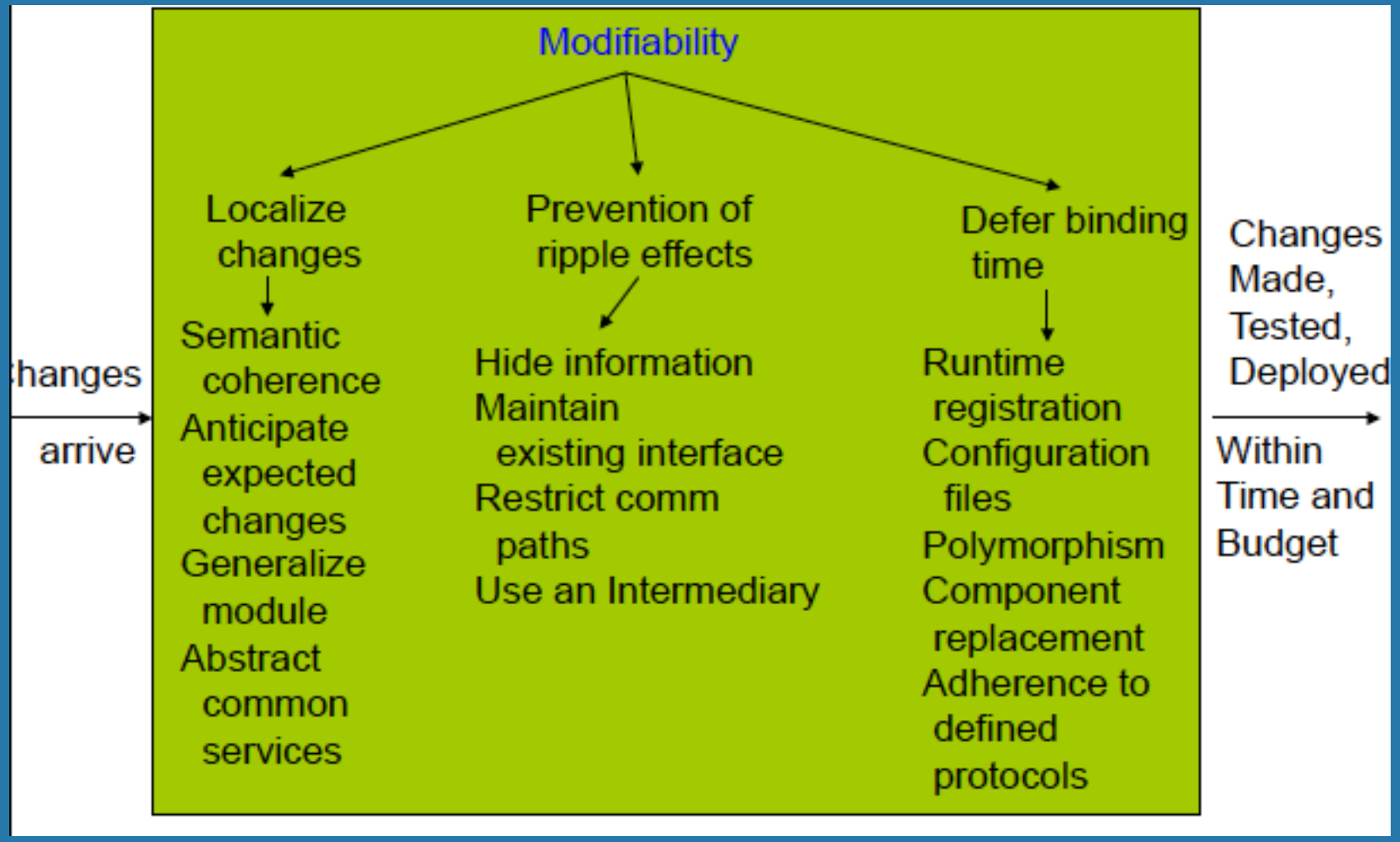
Availability [2]



Modifiability [1]

	Possible Values
Source	End user, developer, system administrator
Stimulus	A directive to add/delete/modify functionality, or change a quality attribute, capacity, or technology
Artifacts	Code, data, interfaces, components, resources, config., …
Env.	Runtime, compile time, build time, initiation time, design time
Response	One or more of the following: . make modification . test modification . deploy modification
Response Measure	Cost in terms of: . number, size, complexity of affected artifacts . effort . calendar time . money (direct outlay or opportunity cost) . extent to which this modification affects other functions or quality attributes . new defects introduced

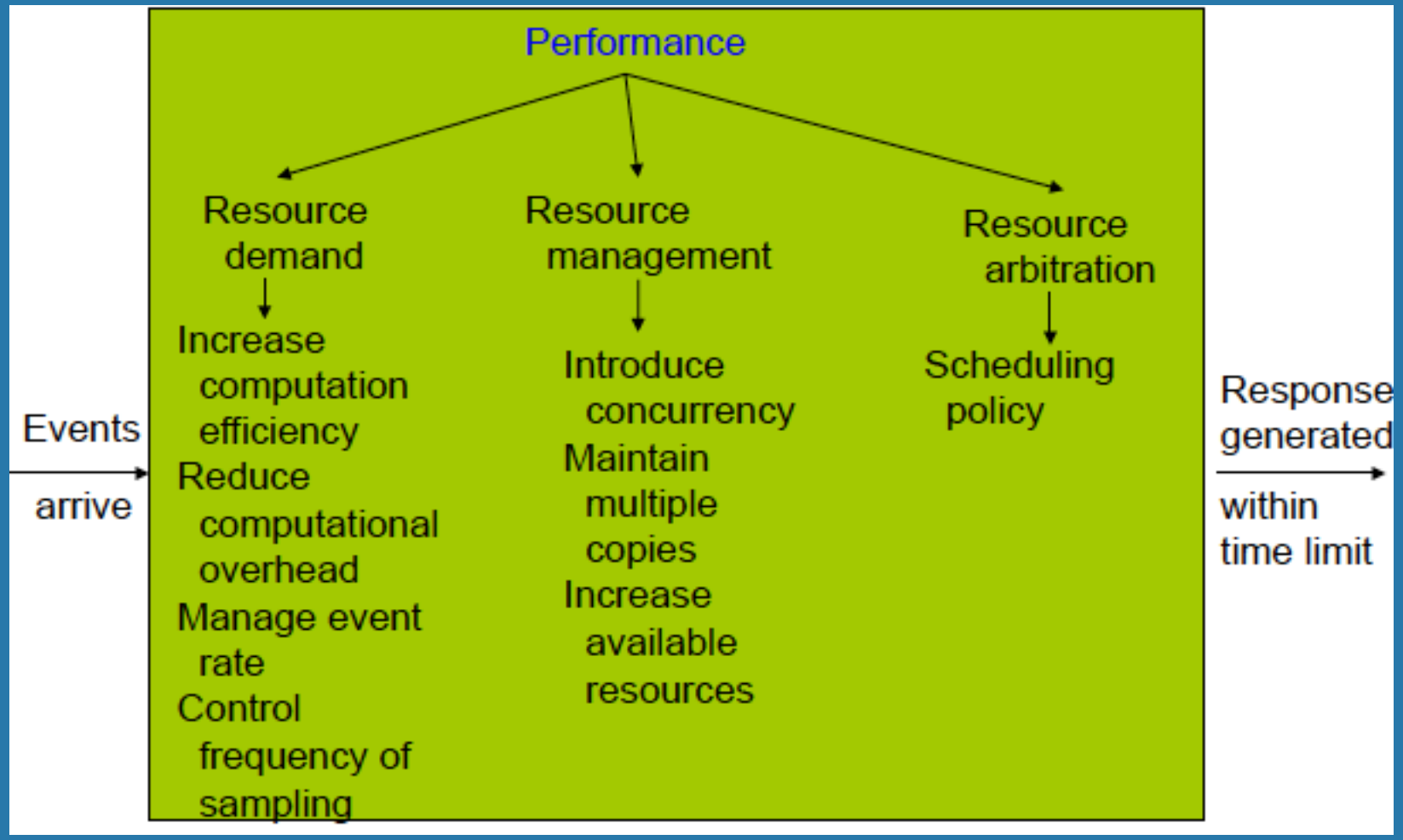
Modifiability [2]



Performance [1]

Portion of Scenario	Possible Values
Source	Internal or external to the system
Stimulus	Arrival of a periodic, sporadic, or stochastic event
Artifact	System or one or more components in the system.
Environment	Operational mode: normal, emergency, peak load, overload.
Response	Process events, change level of service
Response Measure	Latency, deadline, throughput, jitter, miss rate

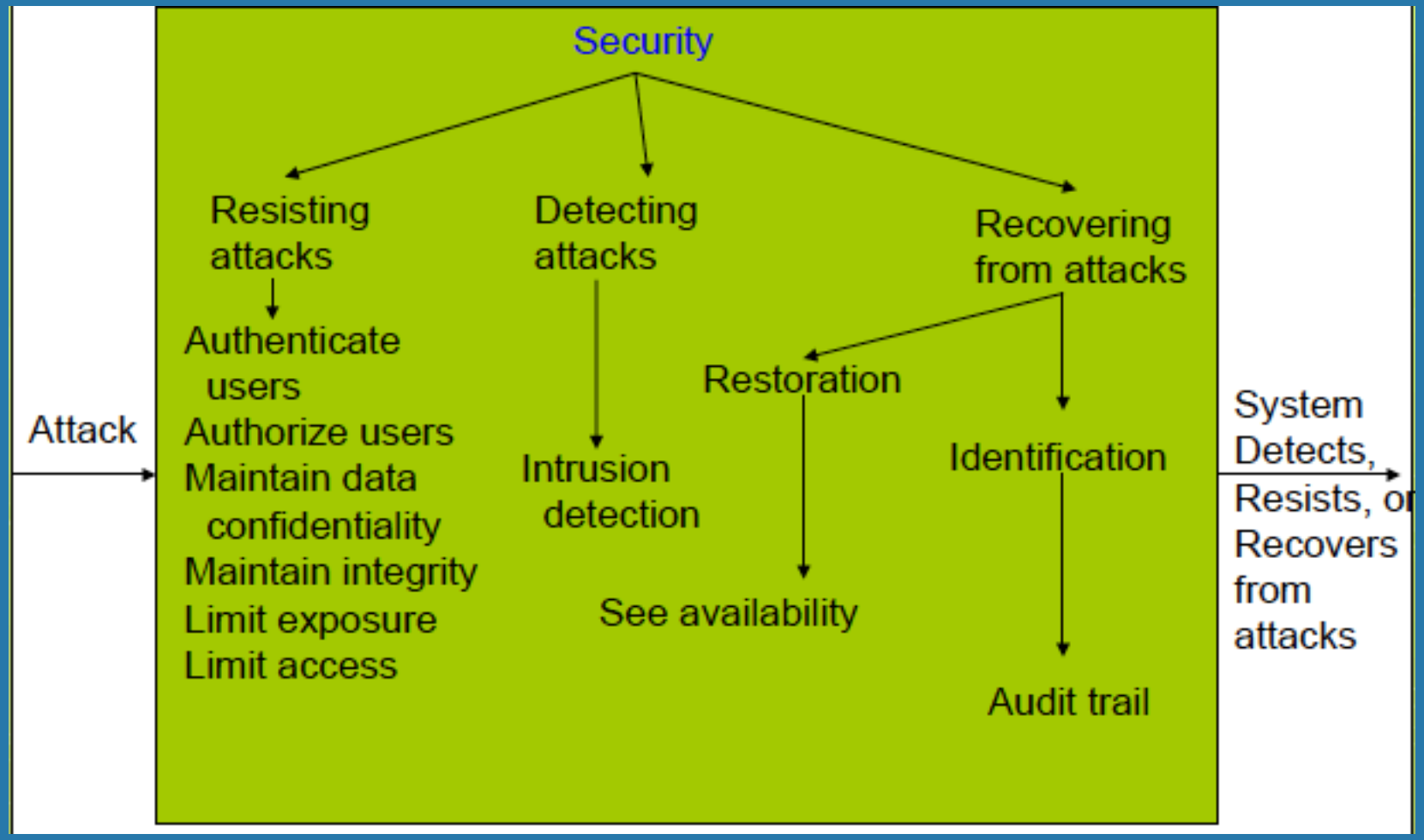
Performance [2]



Security [1]

Portion of Scenario	Possible Values
Source	Human or another system which may have been previously identified (either correctly or incorrectly) or may be currently unknown. A human attacker may be from outside the organization or from inside the organization.
Stimulus	Unauthorized attempt is made to display data, change or delete data, access system services, change the system's behavior, or reduce availability.
Artifact	System services; data within the system; a component or resources of the system; data produced or consumed by the system
Environment	The system is either online or offline, connected to or disconnected from a network, behind a firewall or open to a network, fully operational, partially operational, or not operational
Response	Transactions are carried out in a fashion such that • data or services are protected from unauthorized access; • data or services are not being manipulated without authorization; • parties to a transaction are identified with assurance; • the parties to the transaction cannot repudiate their involvements; • the data, resources, and system services will be available for legitimate use. The system tracks activities within it by • recording access or modification, • recording attempts to access data, resources or services, • notifying appropriate entities (people or systems) when an apparent attack is occurring.
Response Measure	One or more of the following • how much of a system is compromised when a particular component or data value is compromised, • how much time passed before an attack was detected, • how many attacks were resisted, • how long does it take to recover from a successful attack, • how much data is vulnerable to a particular attack

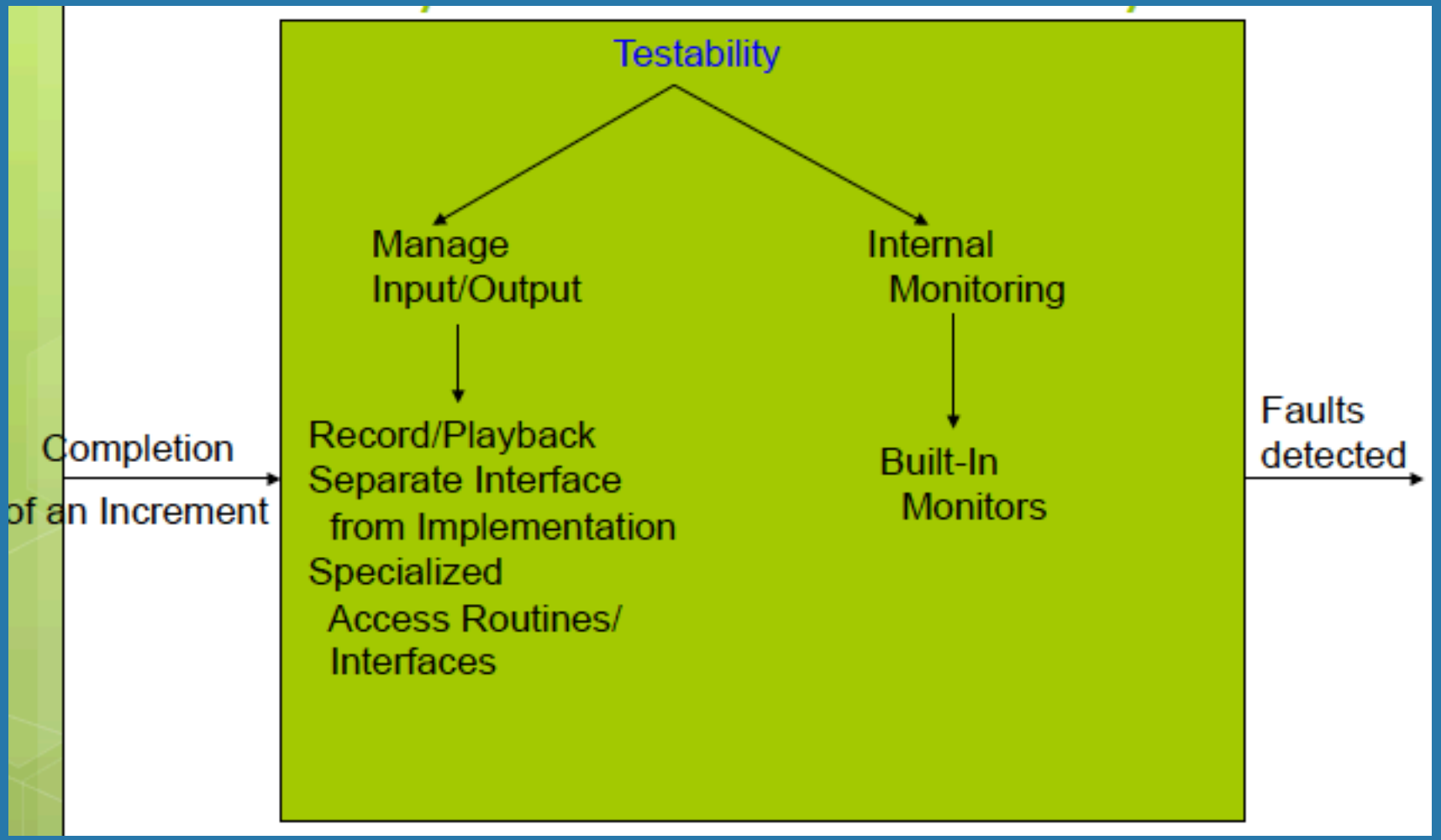
Security [2]


Security
Resisting attacks
Detecting attacks
Recovering from attacks
Authenticate users
Authorize users
Maintain data confidentiality
Maintain integrity
Limit exposure
Limit access
Intrusion detection
Restoration
Identification
See availability
Audit trail
Attack
System Detects, Resists, or Recovers from attacks

Testability [1]

Portion of Scenario	Possible Values
Source	Unit testers, integration testers, system testers, acceptance testers, end users, either running tests manually or using automated testing tools
Stimulus	A set of tests are executed due to the completion of a coding increment such as a class, layer or service; the completed integration of a subsystem; the complete implementation of the system; or the delivery of the system to the customer.
Environment	Design time, development time, compile time, integration time, deployment time, run time
Artifacts	The portion of the system being tested
Response	One or more of the following: execute test suite and capture results; capture activity that resulted in the fault; control and monitor the state of the system
Response Measure	One or more of the following: effort to find a fault or class of faults, effort to achieve a given percentage of state space coverage; probability of fault being revealed by the next test; time to perform tests; effort to detect faults; length of longest dependency chain in test; length of time to prepare test environment; reduction in risk exposure (size(loss) * prob(loss))

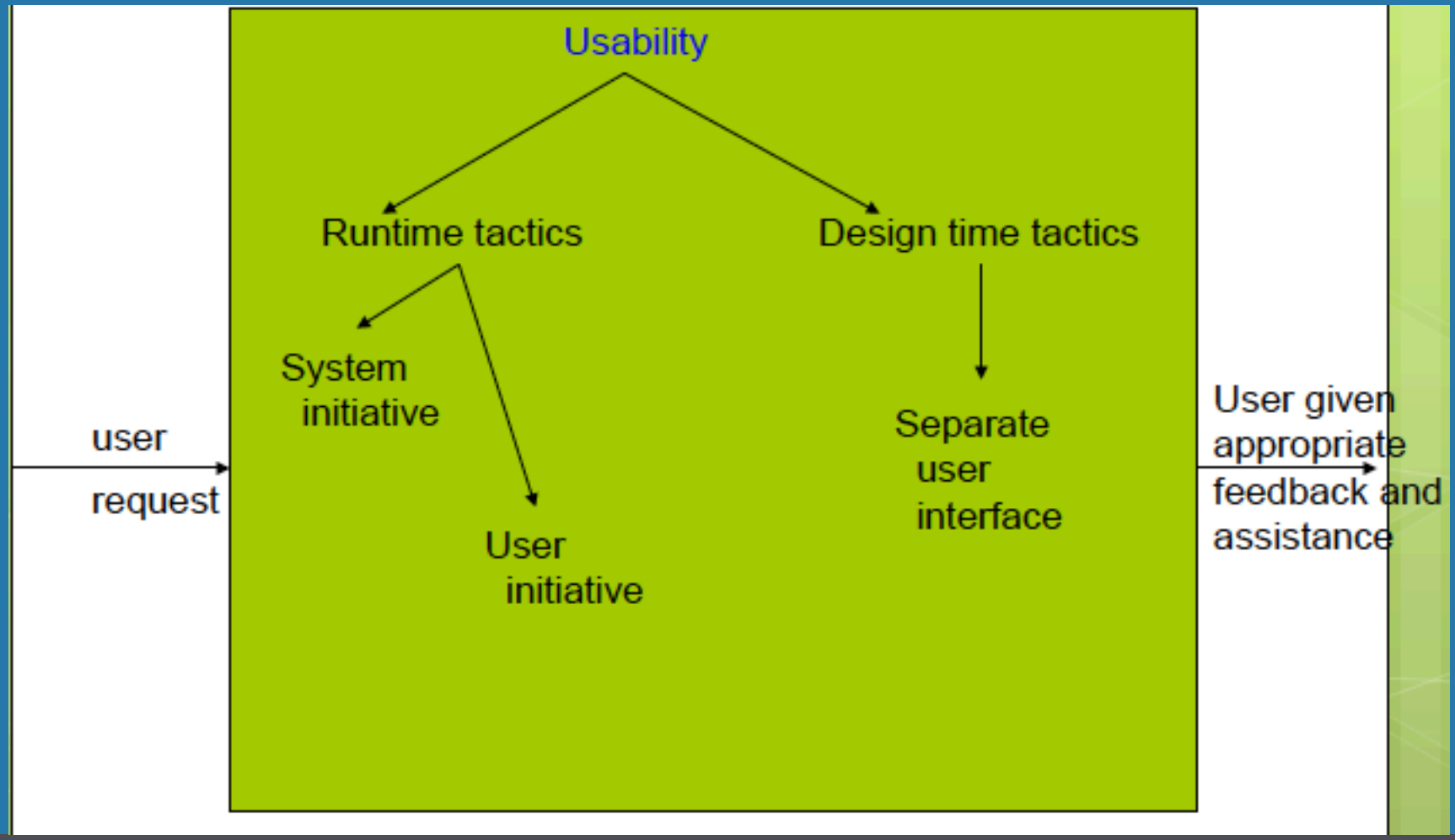
Testability [2]



Usability [1]

Portion of Scenario	Possible Values
Source	End user, possibly in a specialized role
Stimulus	End user tries to use a system efficiently, learn to use the system, minimize the impact of errors, adapt the system, or configure the system
Environment	Runtime or configuration time
Artifacts	System or the specific portion of the system with which the user is interacting.
Response	The system should either provide the user with the features needed or anticipate the user's needs.
Response Measure	One or more of the following: task time, number of errors, number of tasks accomplished, user satisfaction, gain of user knowledge, ratio of successful operations to total operations, or amount of time or data lost when an error occurs.

Usability [2]


Usability
Runtime tactics
Design time tactics
System initiative
User initiative
Separate user interface
user request
User given appropriate feedback and assistance

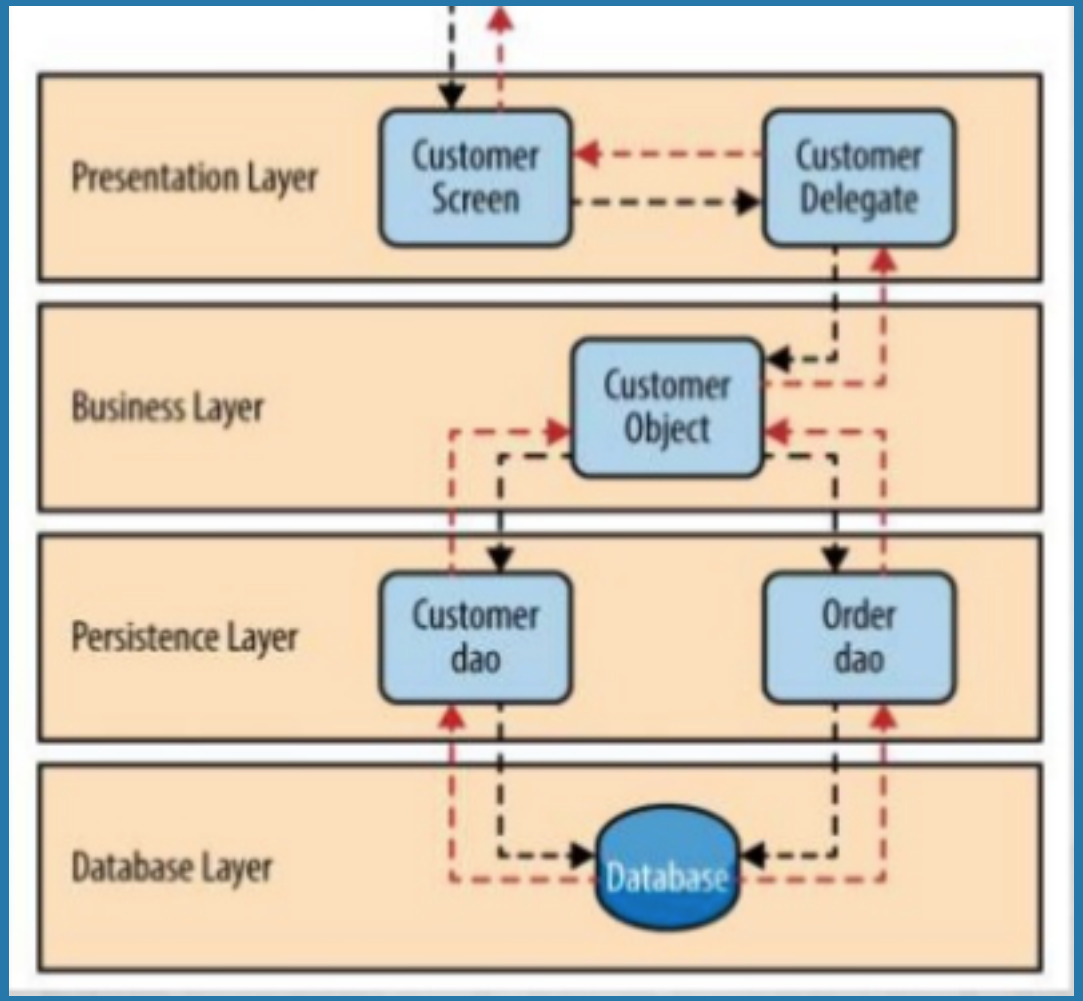
Patrones de Arquitectura

- Los Patrones de Arquitectura establecen las relaciones entre
  - **Un contexto** (una situación común)
  - **Un problema** (apropiadamente generalizado)
  - **Una solución** (una resolución arquitectónica apropiada para resolver el problema)
- Patrones de Arquitectura
  - Layered
  - Pipes and Filters
  - Blackboard
  - Broker
  - MVC
  - Presentation Abstraction Control
  - Microkernel
  - Reflection

Layered Pattern [1]

- Contexto: Todos los sistemas complejos necesitan ser desarrollados y cubiertos en sistemas independientes
- Problema: El software debe ser segmentado en la forma en que los módulos pueden ser desarrollados de manera separada.
- Solución: Layered Pattern divide el sistema en unidades llamadas layers. Cada layer es un grupo de módulos que ofrecen una cohesión de servicios.

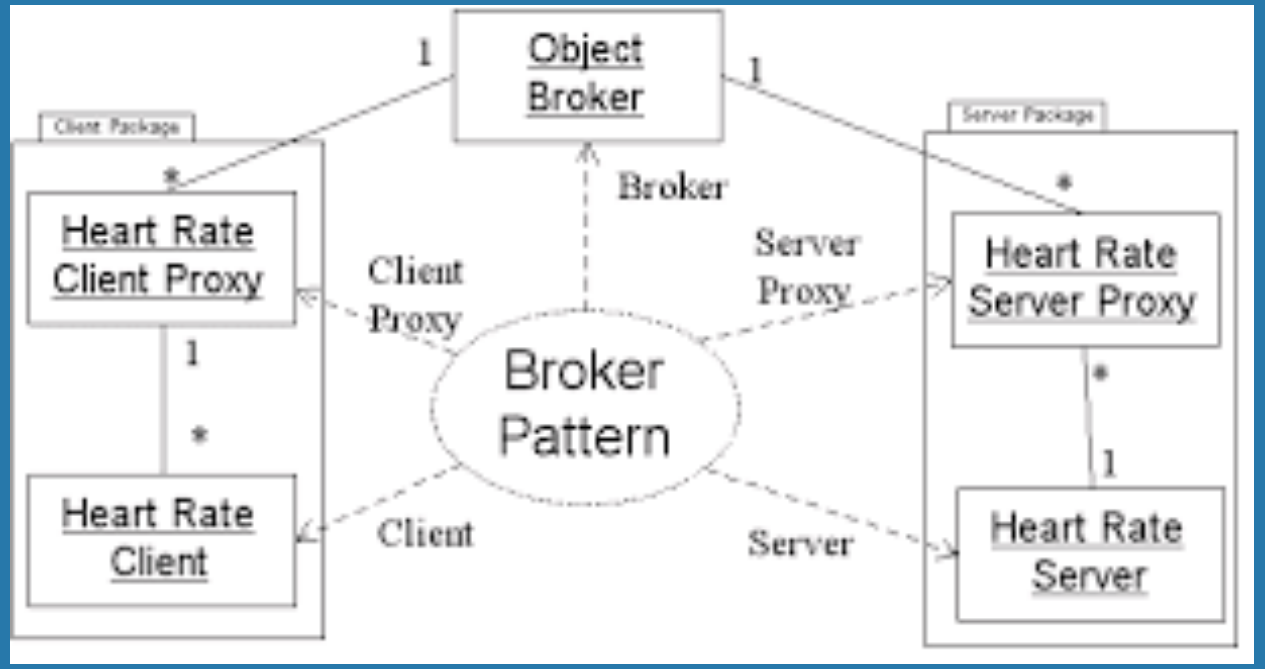
Layered Pattern [2]



Broker Pattern [1]

- Contexto: Muchos sistemas son construidos desde una colección de servicios distribuidos a través de múltiples servicios
- Problema: ¿Cómo hacer una estructura de software distribuida tal que el usuario no necesite saber la naturaleza del lugar y la ubicación de los proveedores del servicio?
- Solución: Broker Pattern separa los servicios del usuario (clientes) de los proveedores de servicios (servidor) insertando un intermediario, llamado broker.

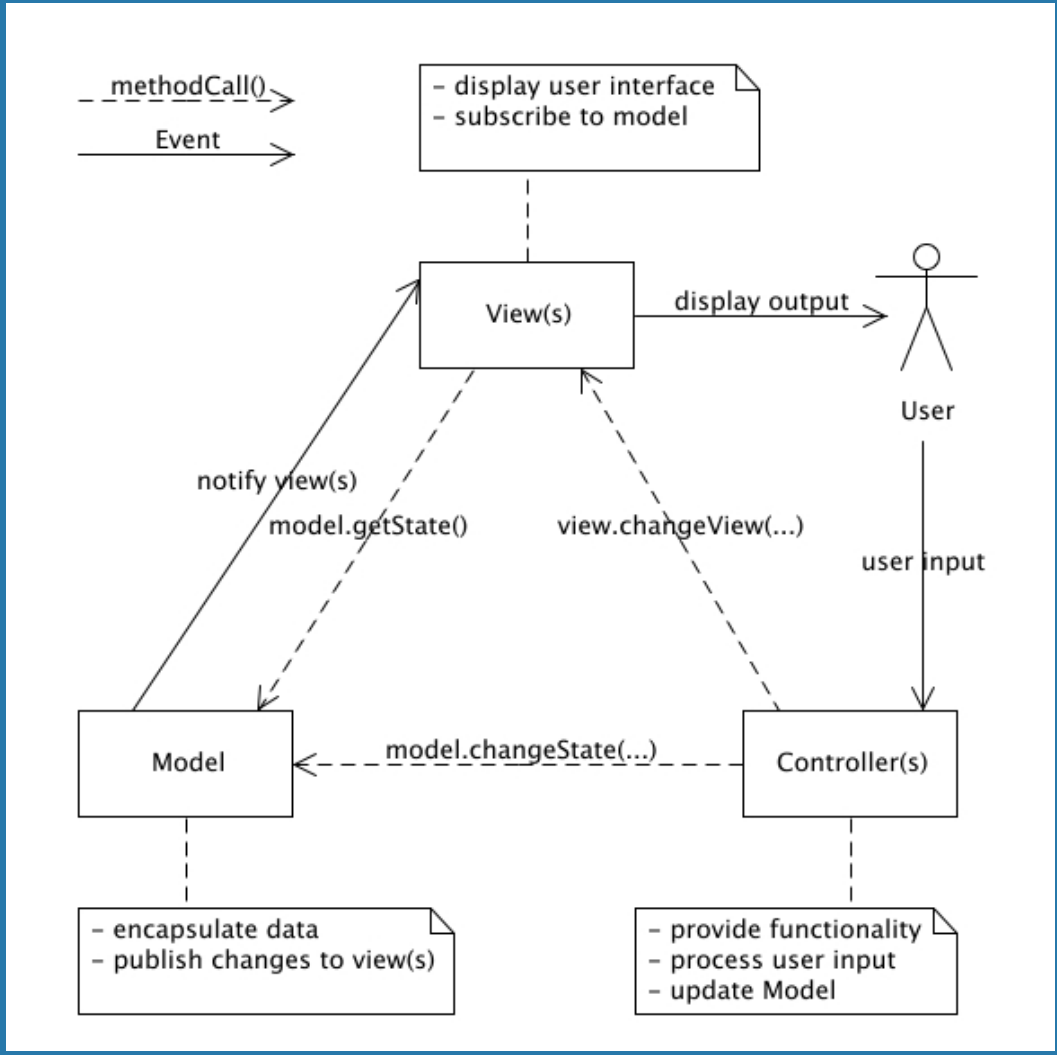
Broker Pattern [2]



Model-View-Controller Pattern [1]

- Conocido por ustedes ☺
- Contexto: La interfaz de usuario es típicamente la parte más modificable de una aplicación interactiva con usuarios.
- Problema: ¿Cómo poder hacer que una interfaz de usuario se mantenga separada de la funcionalidad de la aplicación pero que, a la vez, pueda responder a peticiones de ingreso, borrado o modificación de información?
- Solución: MVC pattern separa las funcionalidades de la aplicación en un modelo, una vista y un controlador.

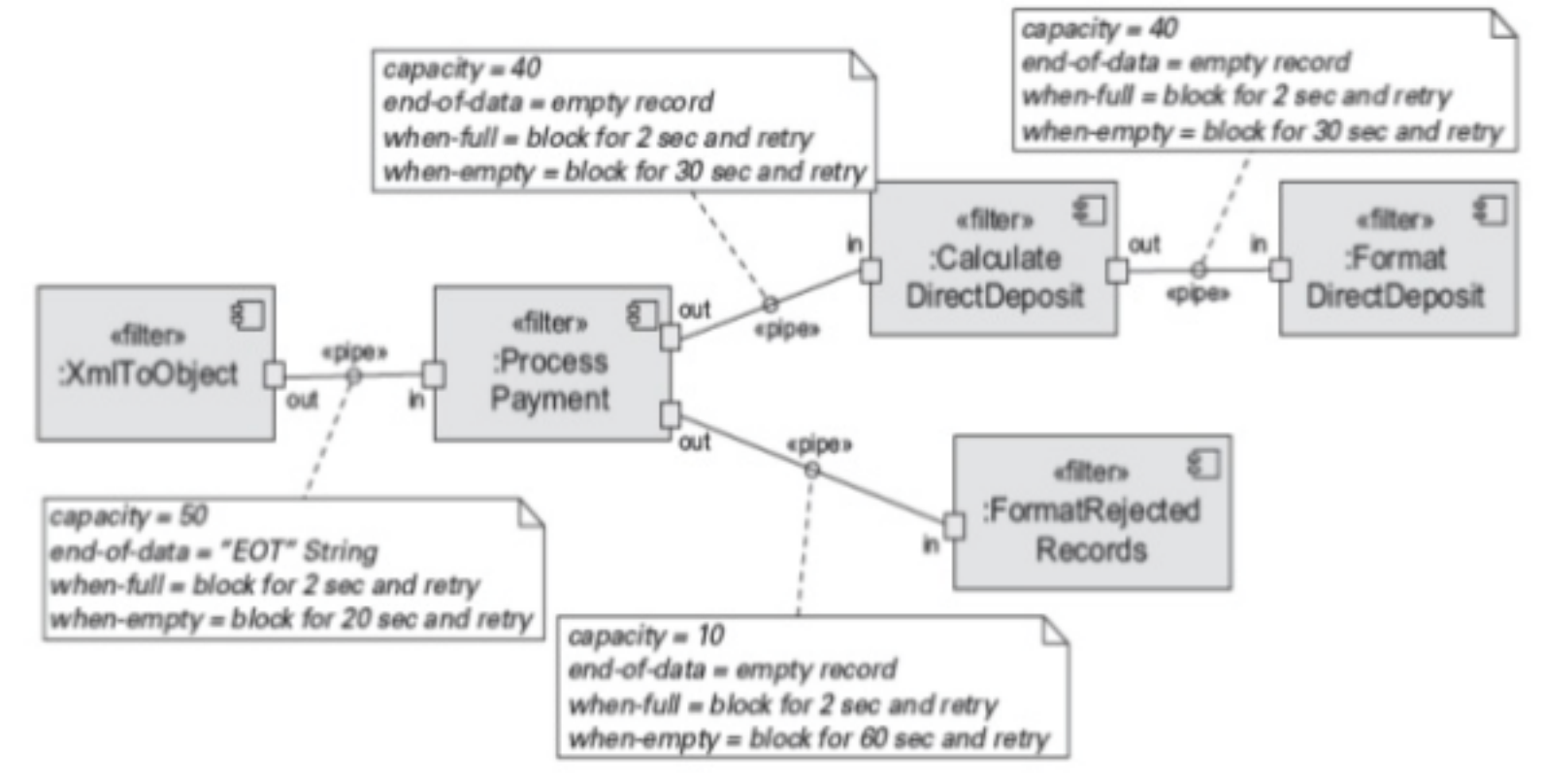
Model-View-Controller Pattern [2]



Pipe-and-Filter pattern [1]

- Contexto: Muchos sistemas requieren transformar streams de datos discretos desde una entrada hacia una salida
- Problema: El sistema necesita ser divido en componentes acoplados con simples mecanismos de interacción.
- Solución: Pipe-and-Filter pattern se caracteriza por las sucesivas transformaciones de streams de datos.

Pipe-and-Filter pattern [2]



Lectura recomendada

- Len Bass, Paul Clements and Rick Kazman. Software Architecture in Practice, Third Edition.

UNIVERSIDAD TECNICA
FEDERICO SANTA MARIA

Departamento de Informática
Universidad Técnica Federico Santa María

Ingeniería de Software

Arquitectura de Software

Cristian Orellana (por Hernán Astudillo) & Gastón Márquez
Departamento de Informática
Universidad Técnica Federico Santa María
<hernan@inf.utfsm.cl, gaston.marquez@sansano.usm.cl>