Maestría en Ingeniería

Curso de Arquitectura de Software

Sesión 8

Fernando Barraza A.
fbarraza@javerianacali.edu.co

Objetivos

- Objetivo: Brindar al estudiante un conocimiento general sobre los métodos de análisis y diseño de AS

- Temas:
  - Modelado de Arquitecturas
  - Metodologias de Análisis y Diseño de AS
  - Arquitectura basada en escenarios (SAAM)
  - Architecture Tradeoff Analysis Method (ATAM)
  - Active Design Review (ADR's)
  - Active Review for Intermediate Design (ARID)
  - Quality Attribute Workshops (QAW) - QASAR
  - Attribute-Driven Design (ADD)

Modelado de Arquitecturas

- **Un modelo de arquitectura define la estructura del sistema a un alto nivel**
  - Componentes y sus interfaces, relaciones estructurales, responsabilidades de comportamiento, coordinación, control de flujo, comunicación y flujo de datos
- **El proceso de diseño de una arquitectura debe considerar**
  - Requerimientos no funcionales, aspectos de calidad, diferentes consumidores de los modelos, tecnologías disponibles.

Consideraciones típicas

- Que dominio de aplicación vamos a desarrollar?
- Es nuestra aplicación de misión crítica para el negocio?
- Quienes son nuestros competidores?
- Cual es la plataforma candidata?

Objetivo de la metodología

- Ayuda a partir un proyecto grande en porciones manejables
- Ayuda a la comunicación entre los miembros del equipo
- Ayuda a recordar en que etapa nos encontramos
- Mejores entregables y comunicación con el personal de post-venta

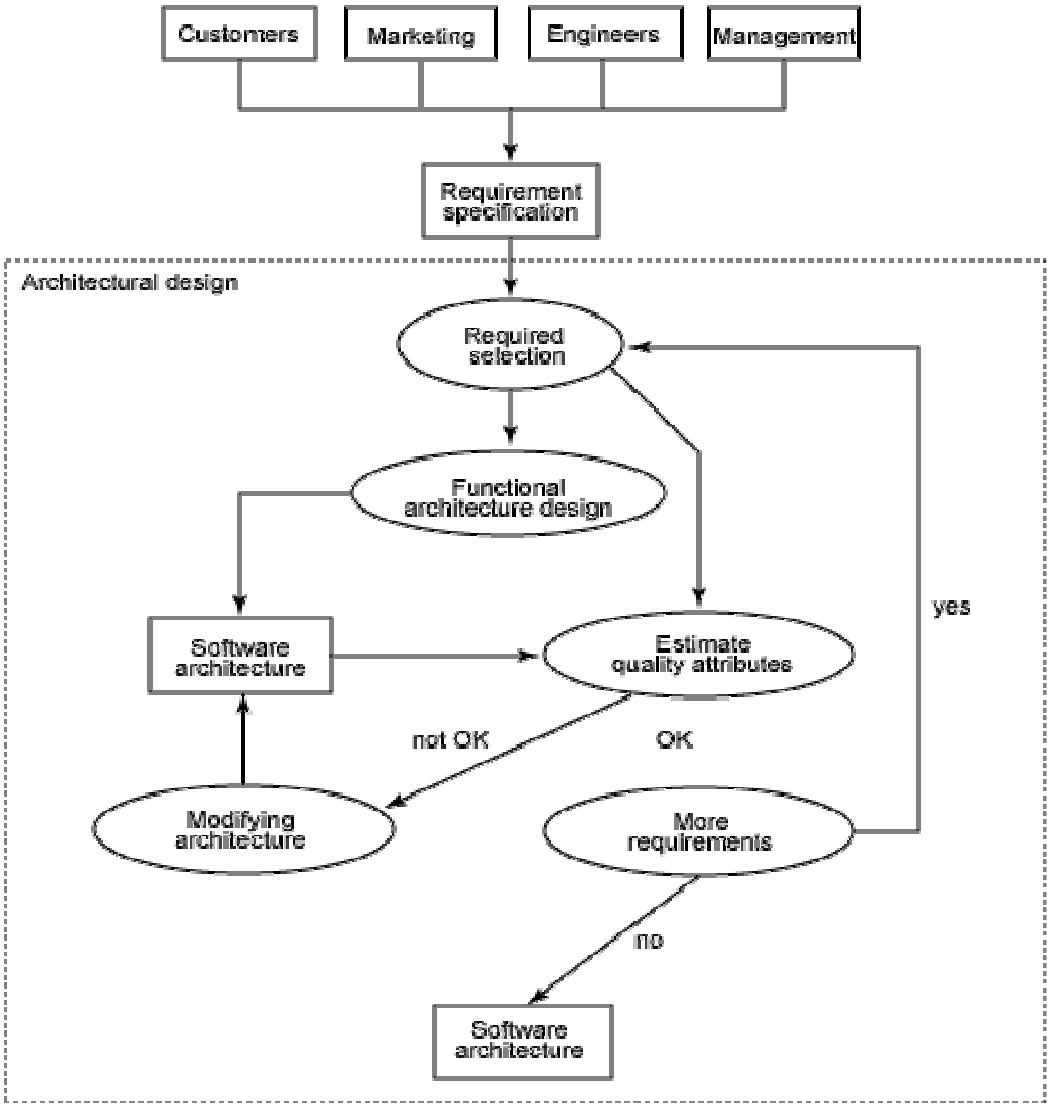
Metodologías para diseño de Arquitecturas

- No son el lenguaje para describirlas y representarlas
- No son el proceso de desarrollo de software
- Son específicas dependiendo del dominio del problema

Metodologías para diseño de AS

- Existen diferentes aproximaciones
  - Iterativas e incrementales
  - Adaptación y generalización
  - Dirigidas hacia el desarrollo
  - Basadas en Escenarios

Proceso de diseño de una arquitectura



Diferentes objetivos de las metodologías en AS

- Ayudar a identificar los requerimientos del sistema hacia la AS
- Analizar los criterios de calidad de la AS
- Evaluar y comparar AS
- Documentar la AS

Scenario-Based Software Architecture

- Se centra en los métodos para el análisis y evaluación de Software soportados sobre sus atributos de calidad
- Los métodos más conocidos son:
  - SAAM, Software Architecture Analysis Method,
  - ATAM, Architecture Trade-off Analysis Method
  - CBAM, Cost Benefit Analysis Method
  - ALMA, Architecture Level Modifiability Analysis
  - FAAM, Family Architecture Analysis Method

SAAM

- SAAM fue el primer método de analisis de arquitectura de software ampliamente divulgado.
- Su objetivo es medir la “modificabilidad” de la AS
- Sus creadores buscaron un método capaz de expresar los diferentes atributos de calidad de AS (modificabilidad, flexibilidad, mantenibilidad, etc.) por medio de escenarios que se evaluan contra los actuales
- Se ha probado su utilidad midiendo atributos como modificabilidad, portabilidad, integrabilidad, asi como funcionalidad
- El método permite identificar los puntos débiles y fuertes, junto con los puntos donde la arquitectura falla en cumplir los requerimientos

Pasos en el método SAAM

- Paso 1: Desarrollar Escenarios:
  - Es un ejercicio de “brainstorm” con el alcance para identificar el tipo de actividades que el sistema debe soportar.
  - Las actividades con sus modificaciones son llamadas “Escenarios”
  - El reto es capturar todos los “usos” y usuarios del sistema, todos sus atributos de calidad y los niveles asociados que el sistema debe alcanzar con miras a los futuros cambios.

Pasos en el método SAAM

- Paso 2: Describir Arquitecturas
  - Se presentan las arquitecturas candidatas.
  - La notación de la AS debe ser bien entendida por los participantes en el proceso y debe indicar tanto la representación estática del sistema (componentes, sus interconecciones y su relación con el ambiente operativo) como su comportamiento.

Pasos en el método SAAM

- Paso 3: Clasificar y priorizar escenarios
  - Los escenarios son clasificados en directos e indirectos (son equivalentes a los casos de uso en UML)
  - Un escenario directo es soportado por la arquitectura candidata porque es basado en sus requerimientos
  - Uno indirecto es una secuencia de eventos que implican cambios menores en la arquitectura
  - La priorización de los escenarios está basada en un proceso de votación

Pasos en el método SAAM

- Paso 4: Evaluar individualmente escenarios indirectos
  - En un escenario directo el arquitecto demuestra como el escenario puede ser ejecutado por la arquitectura
  - En uno indirecto el arquitecto describe como la arquitectura necesitaría cambiar para acomodarse al escenario
  - Por cada escenario indirecto debe identificarse las modificaciones arquitecturales necesarias para facilitar que el escenario evolucione hacia el nuevo sistema, estimando costos y esfuerzos para su implementación.

Pasos en el método SAAM

- Paso 5: Evaluación de escenarios de interacción
  - Cuando uno o mas escenarios son cambios requeridos sobre el mismo componente de la arquitectura, ellos deben interactuar.
  - En este caso, los componentes afectados nececitan ser modificados o divididos en sub-compnentes para poder evitar la interacción de los diferentes escenarios.

Pasos en el método SAAM

- Paso 6: Crear una evaluación general
  - Finalmente se le asigna un peso a cada escenario en términos de su importancia relativa con respecto al éxito del sistema
  - Los pesos están ligados a criterios como costo, riesgo, “time-to-market”, etc.
  - Se escoge la mejor evaluada para los posibles escenarios.

Architecture Tradeoff Analysis Method - ATAM

- En un método probado para la evaluación de As
- Se basa en la presencia en el proceso de los “consumidores” del diseño (stakeholders que pueden ser clientes, probadores, ingenieros o cualquiera que esté interesado en probar la suficiencia y robustez de la arquitectura)
- La arquitectura es analizada comparando las calidades requeridas con respecto a las propiedades conocidas de las aproximaciones usadas en la arquitectura, dentro de los escenarios de prueba (por análisis o experimentación)

Pasos en ATAM

ATAM Steps	Description
Present the ATAM	Evaluation team leader presents a method overview to the participants.
Present business drivers	Client or representative of system whose architecture is being evaluated presents the business drivers underlying the architecture.
Present architecture	Architect makes presentation.
Identify architectural approaches	Evaluation team catalogs architectural approaches used, as basis for subsequent analysis
Generate quality attribute utility tree	Participants build utility tree to identify quality attributes (and the scenarios that express them) of interest. Evaluation team facilitates.
Analyze architectural approaches	Evaluation team and architect perform analysis based on qualities desired and approaches used.
Brainstorm and prioritize scenarios	The architecture's stakeholders adopt an additional set of scenarios expressing architectural requirements. Evaluation team facilitates.
Analyze architectural approaches	Evaluation team and architect perform analysis based on these new scenarios.
Present results	Evaluation team leader makes presentation of analysis results, lists identified risks, sensitivity points, and tradeoffs in the architecture.

ADR´s

- Son técnicas utilizado principalmente para evaluar el diseño detallado de unidades coherentes de software, tales como módulos o componentes.
- Las preguntas a resolver son:
  - Calidad y completitud de la documentación
  - Suficiencia, robustez y ajustabilidad de los servicios provistos por el diseño.
- Los evaluadores son escogidos entre los “consumidores” del diseño y de la documentación.

ADR’s y su efectividad

- Alguna inquietudes al diseño de AS:
  - Conocer si el diseño es implementable. Es decir, si los programadores pueden seguir la AS.
  - Está el diseño conceptualmente coherente?

Active Reviews for Intermediate Designs (ARID)

- Es un híbrido entre ADR y ATAM
- Al igual que ATAM brinda una forma para evaluar diseños de forma preeliminar.
- ATAM se enfoca en evaluar toda la arquitectura y no una parte de ella, lo cual es contrario en ARID.

Características para pensar en ARID

- Cuando el diseño no está lo suficientemente bien documentado.
- Cuando los programas fueron diseñados con el significado que sus nombres y parámetros tienen, pero no hay información detallada sobre:
  - Excepciones que pueden levantarse por cada programa
  - Que podría pasar si los programas fueron llamados en estados “ilegales”
  - Como declarar variables de algún tipo
  - Como los programas en procesos diferentes se intercomunican

Comparación de técnicas

Conventional Design Review Questions	Active Design Review Instructions
Are exceptions defined for every program?	Write down the exceptions that can occur for every program.
Are the right exceptions defined for every program?	Write down the range or set of legal values of each parameter. Write down the states under which it is illegal to invoke the program.
Are the data types defined?	For each data type, write down • an expression for a literal value of that data type; • a declaration statement to declare a variable for that type; • the greatest and least values in the range of that data type.
Are the programs sufficient?	Write a short pseudo-code program that uses the design to accomplish {some defined task}.
Is the performance of each program adequately specified?	For each program, write down its maximum execution time and list the shared resources that it may consume.

Pasos en ARID

Phase 1: Pre-meeting	Step 1: Identify reviewers
	Step 2: Prepare design presentation
	Step 3: Prepare seed scenarios
	Step 4: Prepare for the review meeting
Phase 2: Review meeting	Step 5: Present ARID method
	Step 6: Present design
	Step 7: Brainstorm and prioritize scenarios
	Step 8: Perform review
	Step 9: Present conclusions

Comparación de métodos

	ATAM	ADR	ARID
Artifact examined	Architecture	Architectural and/or design documentation	Conceptual design approach, with embryonic documentation
Who participates	Architect, stakeholders	Experts in the design realm, consistency and completeness checkers	Lead designer, stakeholders
Basic approach	Elicit drivers and qualities, build utility tree, catalog approaches, perform approach-based analysis.	Construct review questionnaires, assign reviewers tasks, analyze their results to evaluate quality	Present design, elicit desired uses, have reviewers as a group use the design.
Outputs	Identified risks, sensitivity points, and tradeoff points	Errors, gaps, and inconsistencies	Issues and problems preventing successful usage

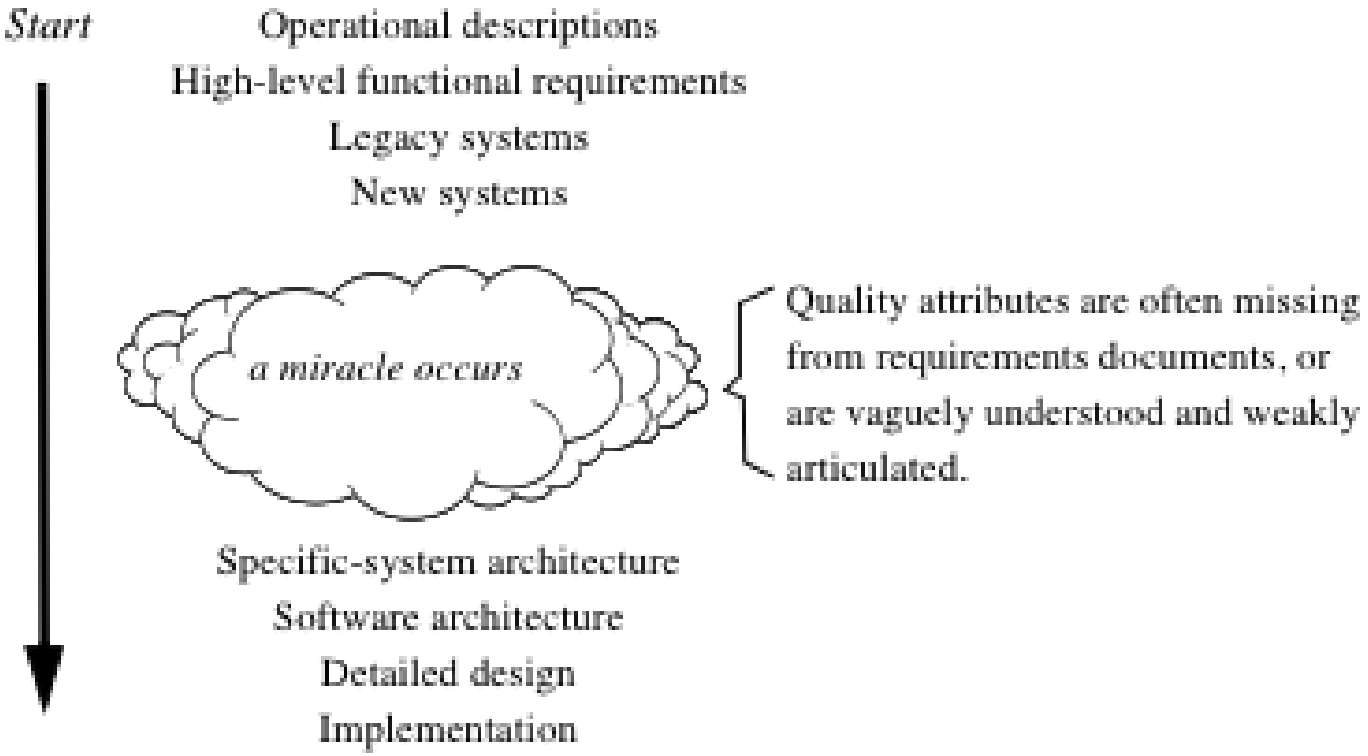
Técnicas complementarias

- Mejoran el desempeño del método cuando las condiciones no son del todo propicias para su implementación
- Se siguen de forma paralela o intercalada con el método principal

Quality Attribute Workshop - QAW

- Busca el descubrimiento de atributos de calidad críticos del software en las fases iniciales de su desarrollo.
- Provee una forma para identificar dichos atributos y clarificar los requerimientos del sistema antes que la arquitectura de software ha sido creada

Caso típico entre requerimientos y software



Pasos de QAW

1. Presentación de QAW
2. Presentación del misión y visión del negocio
3. Presentación del plan de arquitectura
4. Identificación de puntos críticos de la AS
5. Lluvia de ideas de escenarios
6. Consolidación de escenarios
7. Priorización de escenarios
8. Refinamiento de escenarios

Beneficios de QAW

- Resultados:
  - Lista de puntos críticos de las AS
  - Una lista de escenarios “en bruto”
  - Una lista refinada de los escenarios priorizados
- Estos resultados ayudan a:
  - Actualizar la visión de la arquitectura
  - Refinar los requerimientos de software y del sistema
  - Guiar el desarrollo de prototipos
  - Ejercitar simulaciones
  - Influenciar el orden en que la AS es desarrollada
  - Describir la operación del sistema

Ejemplo de un escenario refinado

Scenario Refinement for Scenario N		
Scenario(s):		When a garage door opener senses an object in the door's path, it stops the door in less than one millisecond.
Business Goals:		safest system; feature-rich product
Relevant Quality Attributes:		safety, performance
Scenario Components	**Stimulus:**	An object is in the path of a garage door.
	Stimulus Source:	object external to system, such as a bicycle
	Environment:	The garage door is in the process of closing.
	Artifact (If Known):	system's motion sensor, motion-control software component
	Response:	The garage door stops moving.
	Response Measure:	one millisecond
Questions:		How large must an object be before it is detected by the system's sensor?
Issues:		May need to train installers to prevent malfunctions and avoid potential legal issues.

Attribute Driven Design (ADD)

- Es un método para definir una arquitectura de software basando el proceso de diseño en los atributos de calidad que el software debe cumplir
- El proceso sigue una descomposición recursiva, donde cada fase en la descomposición, atributos de calidad son escogidos para satisfacer un conjunto de escenarios.
- La funcionalidad es utilizada para “instanciar” el componente y los tipos de conectores provistos por las primitivas.

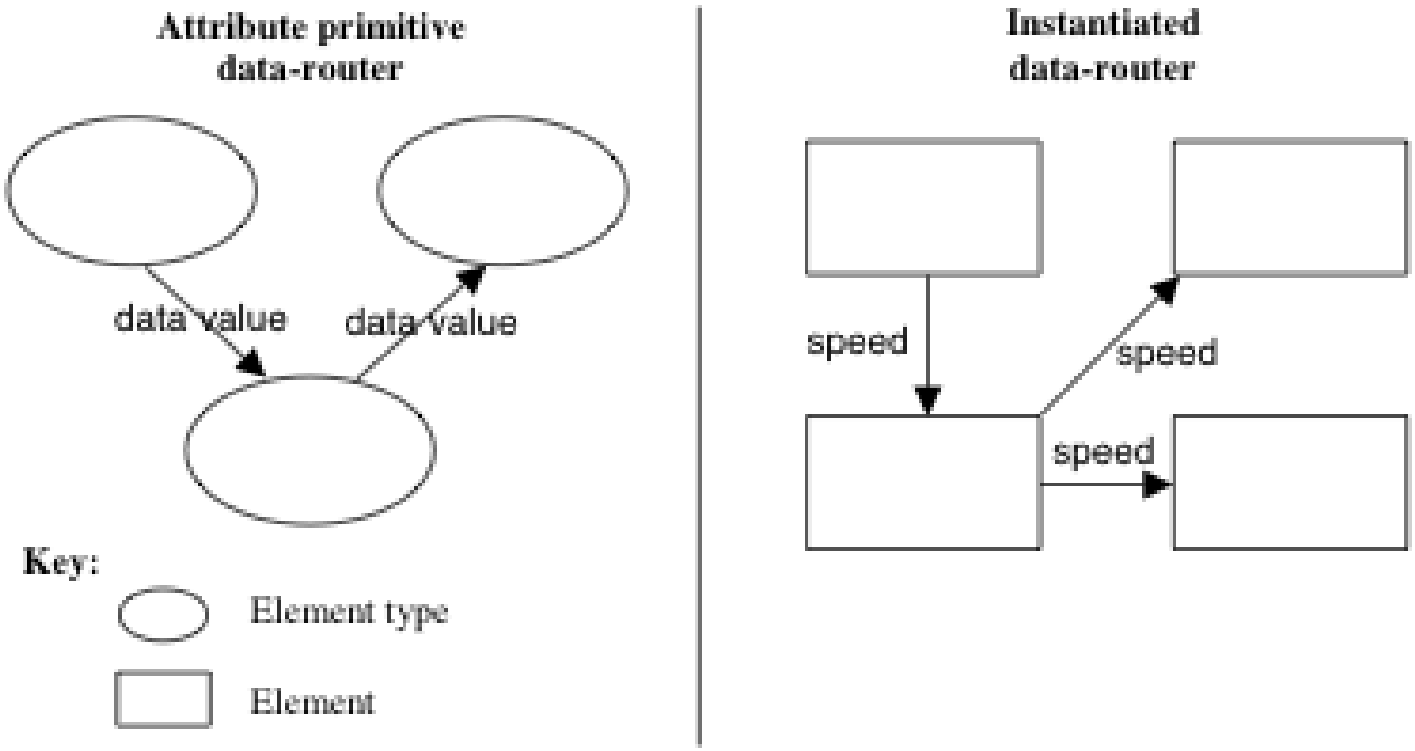
Primitivas de atributos

- Son una colección de conectores que colaboran para alcanzar algunas metas de un atributo de calidad (expresado como un escenario de calidad) y es lo mínimo con respecto al alcance de esas metas.
- Ejemplos son un enrutador de datos o un cache y los componentes que los accesan.

Ejemplos

- Un enrutador de datos protege a productores de las adiciones y cambios de los consumidores y viceversa, limitando el conocimiento que tienen unos de otros. Esto afecta o contribuye con la “modificabilidad” del sistema.
- Un cache reduce el tiempo de respuesta proveyendo una copia de los datos cercanos a los que la función necesita. Esto contribuye o afecta el desempeño del sistema.

Instanciación de atributos



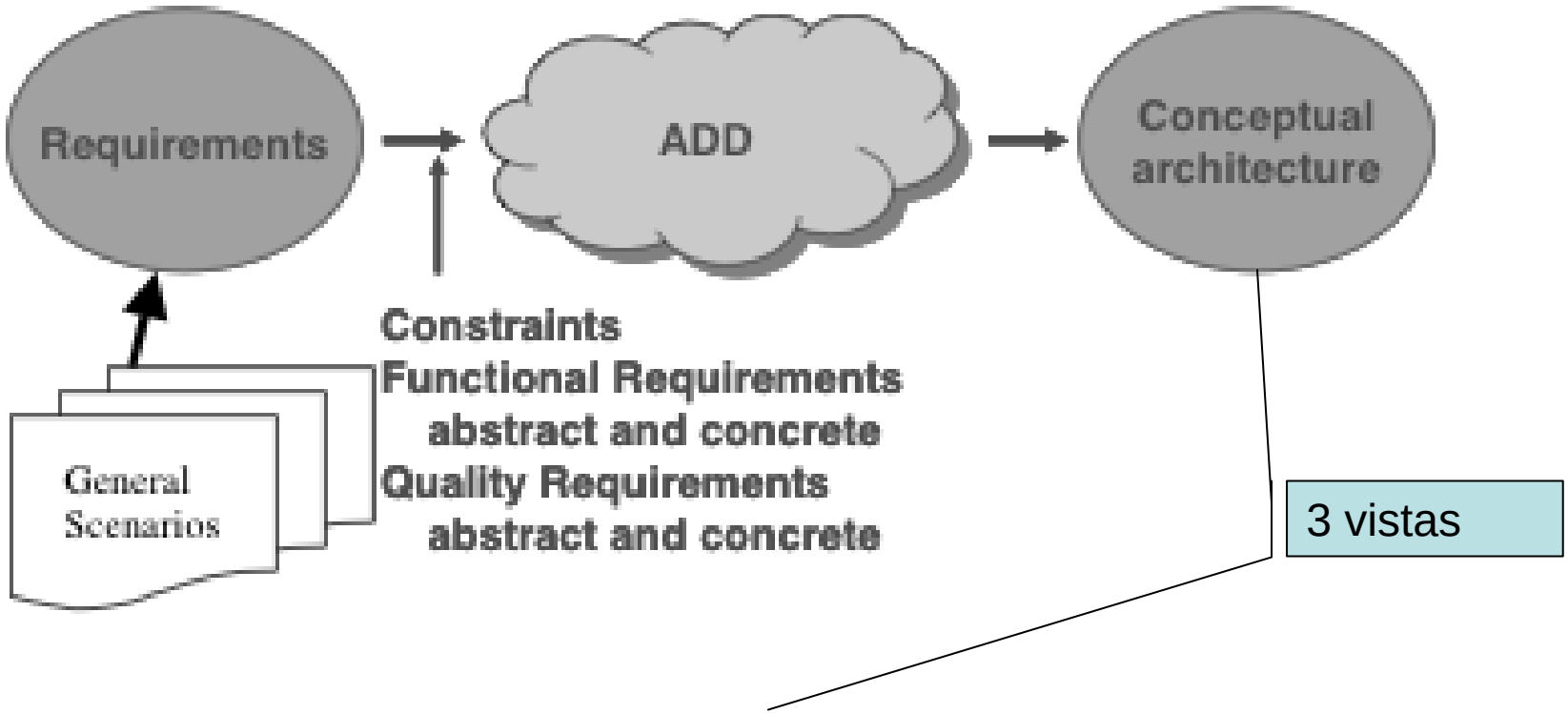
Ejemplos de Primitivas de atributos

Performance	Modifiability	Security	Availability	Testability	Usability
Load balancing	Data Router	Encryption	Ping/Echo	Monitors	Separation of command from data
Priority assignment	Data repository	Integrity	Voting	Backdoor	Separation of data from the view of that data
Fixed priority scheduling	Virtual machine	Firewalls	Recovery blocks	open APIs	Replication of commands
Cyclic Executive	Interpreter	Mirroring of databases	Atomic transactions		Recording
Client-Server		Audit trail	Checkpoints		Explicit models for Task, User, System

Ciclo de vida de ADD

- ADD toma los requerimintos, calidad y funcionalidad como entrada
- Esto significa que ADD está en el ciclo de vida despues de la fase de análisis de requerimientos, pero no significa que esta fase debe acabar para empezar ADD
- La salida de ADD es una arquitectur conceptual. Esto significa que ADD no completa el diseño de arquitectura pero provee las bases para hacerlo.

Ciclo de vida ADD



•The module view shows the structural elements and their relations.
•The concurrency view shows the concurrency in the system.
•The deployment view shows the deployment of functionality onto the execution hardware.

Créditos

- **Quality Attribute Workshops (QAWs), Third Edition**. Mario R. Barbacci, Robert Ellison, Anthony J. Lattanze, Judith A. Stafford, Charles B. Weinstock, William G. Wood.
- **Quality Attribute Design Primitives and the Attribute Driven Design Method.** Len Bass, Mark Klein, and Felix Bachmann
- **Active Reviews for Intermediate Designs.** Paul C. Clements

Créditos

- Software Engineering Institute:
  - Len Bass
  - Mark Klein
  - Felix Bachmann
  - Paul C. Clements.
  - Mario R. Barbacci
  - Robert Ellison
  - Anthony J. Lattanze
  - Judith A. Stafford
  - Charles B. Weinstock
  - William G. Wood
- Department of Mathematics and Computing Science, Technical University Eindhoven & Department Software Architectures, Philips Research:
  - Mugurel T. Ionita
  - Dieter K. Hammer
  - Henk Obbink