



Universidad de Oviedo



# Comunicando la Arquitectura del Software

2025-26



ARQUITECTURA  
DEL SOFTWARE

Jose Emilio Labra Gayo

# Contenidos

## Comunicando la arquitectura del software

- Objetivo de la documentación

- Personas interesadas

- Vistas

- Documentación y proyectos ágiles

- Directrices

## Enfoques de documentación

- Vistas 4+1 de Kruchten

- Vistas y más

- Modelo C4

- arc42

# Comunicando la arquitectura del Software

# Arquitectura es más que código

El código no cuenta la historia completa

Preguntas que el código no responde

- ¿Cómo encaja el software en el entorno existente?
- ¿Porqué se han elegido ciertas tecnologías?
- ¿Cuál es la estructura general del sistema?
- ¿Dónde están desplegados los componentes que se ejecutan?
- ¿Cómo se comunican los componentes?
- ¿Cómo y dónde se puede añadir nueva funcionalidad?
- ¿Qué patrones comunes o principios se utilizan?
- ¿Cómo funcionan los interfaces con otros sistemas?
- ¿Cómo se alcanza la seguridad/escalabilidad/... ?
- ...

# Objetivo de la documentación

Objetivo principal: Comunicar la estructura

Comprender la visión general del sistema

Crear una visión compartida: equipo y otras personas interesadas

Vocabulario común

**Describir** qué software se está construyendo y cómo

Facilitar conversaciones técnicas sobre características

Proporcionar mapa para navegar el código fuente

**Justificar** decisiones de diseño

**Ayudar** a desarrolladores nuevos que se unen al equipo

# Requisitos de documentación

Comprensible por las diferentes personas interesadas

*Stakeholders* técnicos y no-técnicos

Reflejar la realidad

Cuidado con separación modelo-código (model-code gap)

Adaptarse a cambios

Adaptarse a proyectos ágiles

Arquitectura evolutiva

# Reglas para buena documentación

Escribir desde el punto de vista del lector

Encontrar quienes serán los lectores y sus expectativas

Evitar repeticiones innecesarias (principio DRY)

Evitar ambigüedad

Explicar la notación (o utilizar notaciones estándar)

Utilizar leyendas o claves para diagramas

Utilizar una organización estándar o plantilla

Añadir Pendiente (TBD/To do) cuando sea necesario

Organizar para referencias/enlaces rápidos

Registrar las justificaciones de las decisiones

Mantener la documentación actual

# Espacio de Problema vs Solución

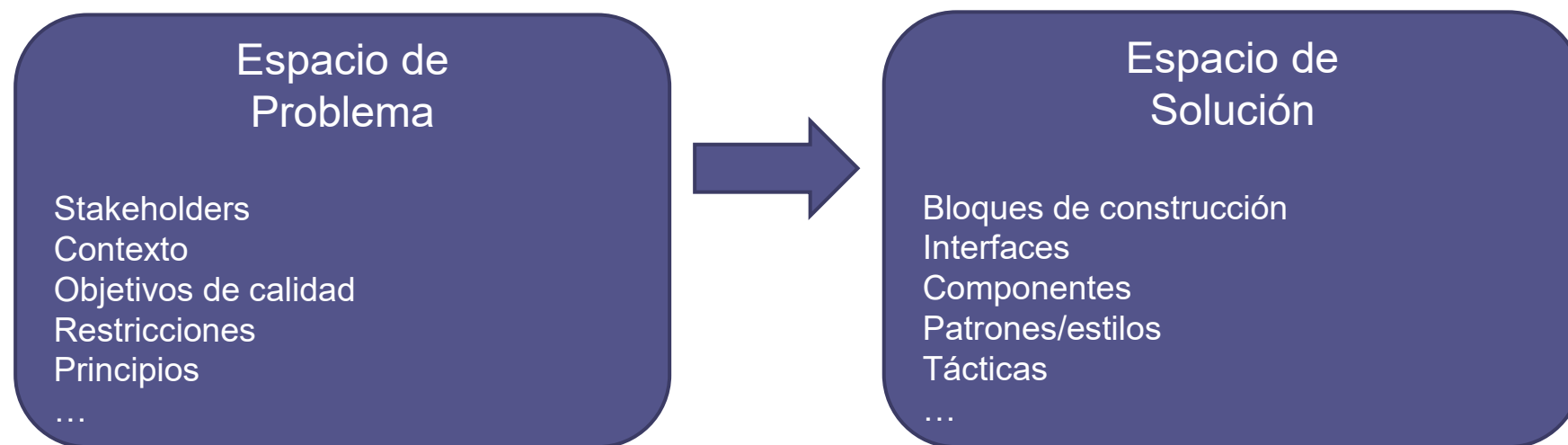
Arquitectura software = camino de problema a solución

Comprender el problema

Diseñar una solución

Justificar las soluciones propuestas

Registrar diferentes alternativas de diseño





# Vistas y puntos de vista

Arquitectura del software = entidad compleja

No puede describirse con una dimensión simple

Requiere varias vistas para diferentes *stakeholders*

Vista = Representación de un sistema respecto a algunas preocupaciones

Diferentes vistas sirven para diferentes objetivos

Punto de vista = Colección de patrones, plantillas y convenciones para construir una vista

Ejemplos: estructura, comportamiento, despliegue

Una vista es lo que ves

Un punto de vista es desde dónde estas mirando

# Documentando las vistas

## Introducción

- Descripción textual de la vista

## Diagrama(s)

- Añadir título descriptivo incluyendo estructuras representadas

- Crear una leyenda explicando significado de símbolos

  - No olvidarse de explicar líneas/flechas

## Lista de elementos y responsabilidades

- Dar nombres descriptivos

- Definir términos (inclusión de glosario)

## Justificación de decisiones

# Documentando vistas

Tratar de conseguir consistencia y simplicidad

Mantener los elementos consistentes

Colores, formas, flechas,...

Si se utiliza un esquema de color, seguirlo consistentemente

Chequear nombres entre vistas

Registrar posibles inconsistencias

Evitar demasiados detalles

Recordar ley de Miller

Persona media puede memorizar  $7 (\pm 2)$  elementos

# Herramientas para los diagramas

Bocetos

Herramientas de dibujo

Herramientas de diagramas basadas en Texto

Herramientas de modelado

Creación de modelo mediante Ingeniería inversa

Lenguajes de descripción de arquitecturas

# Bocetos

Habitual comenzar con boceto en papel o pizarra

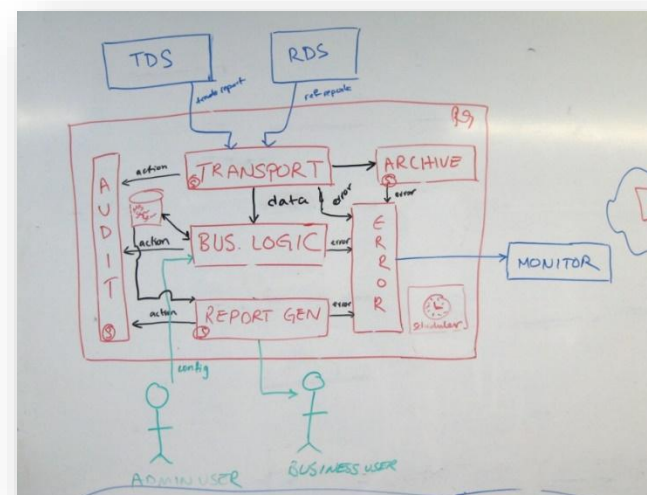
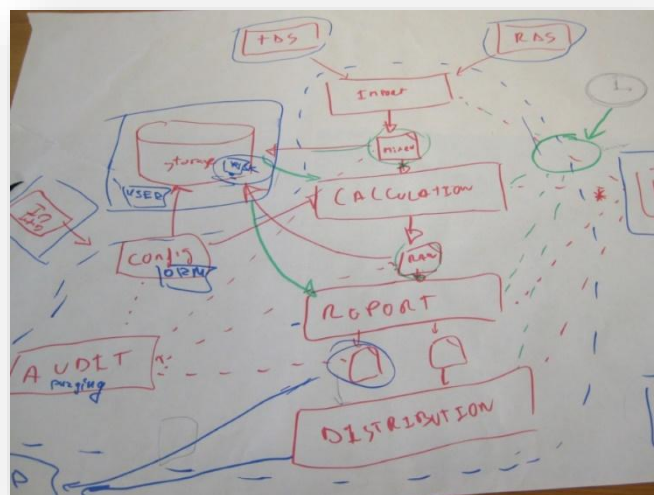
Forma muy buena de colaborar e intercambiar ideas

Normalmente tiene una vida breve

Pero antes o después, deben registrarse

Técnica sencilla hoy en día: Foto

...y posterior conversión a diagramas o modelos



Ejemplo de twitter/X:

# Herramientas de dibujo de diagramas

## Escritorio

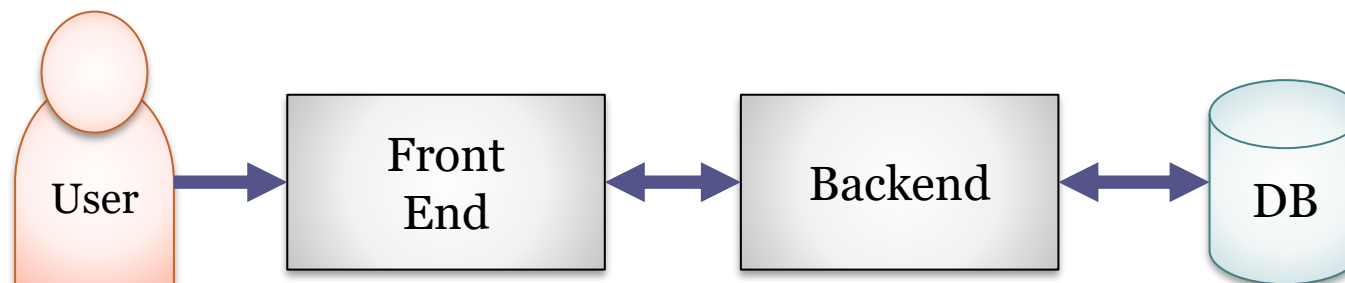
Microsoft Visio, Omnigraffle, SimpleDiagrams, ...

## Basadas en Web:

draw.io, gliffy, LucidChart,...

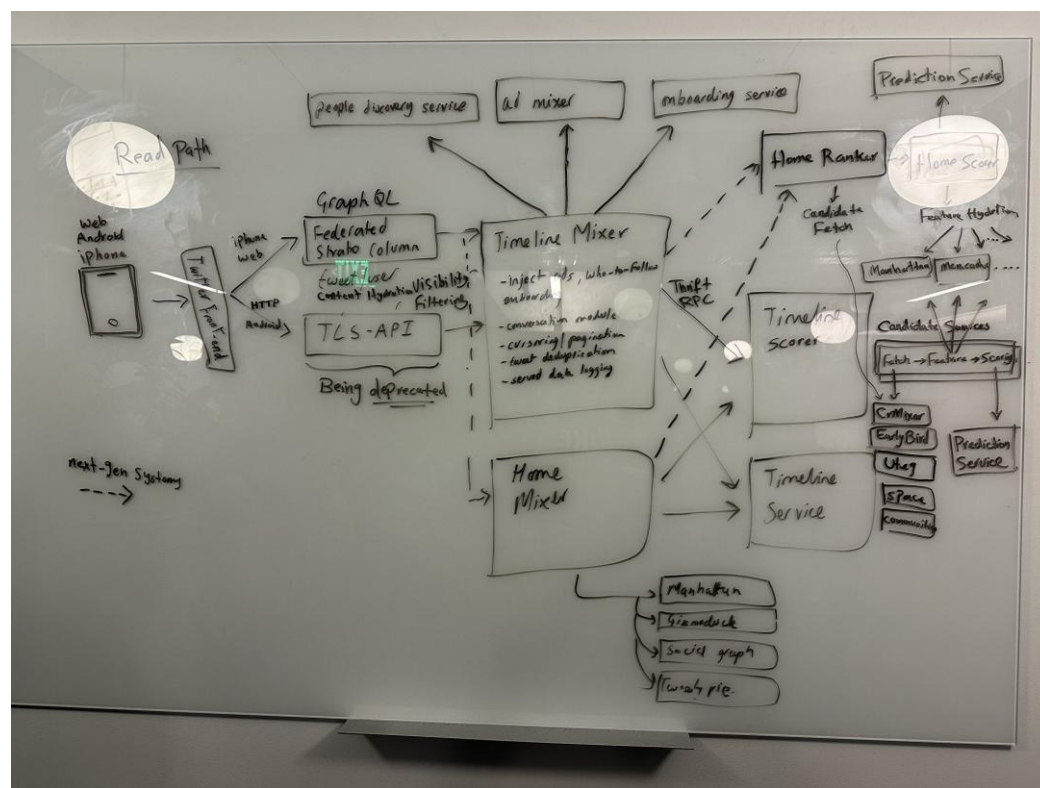
## Herramientas de dibujo propósito general:

Word, **Powerpoint**, Keynote,...



# Bocetos/Diagramas

## Tras la adquisición de Twitter por Elon Musk



Post de Elon Musk:

<https://x.com/elonmusk/status/1593899029531803649>

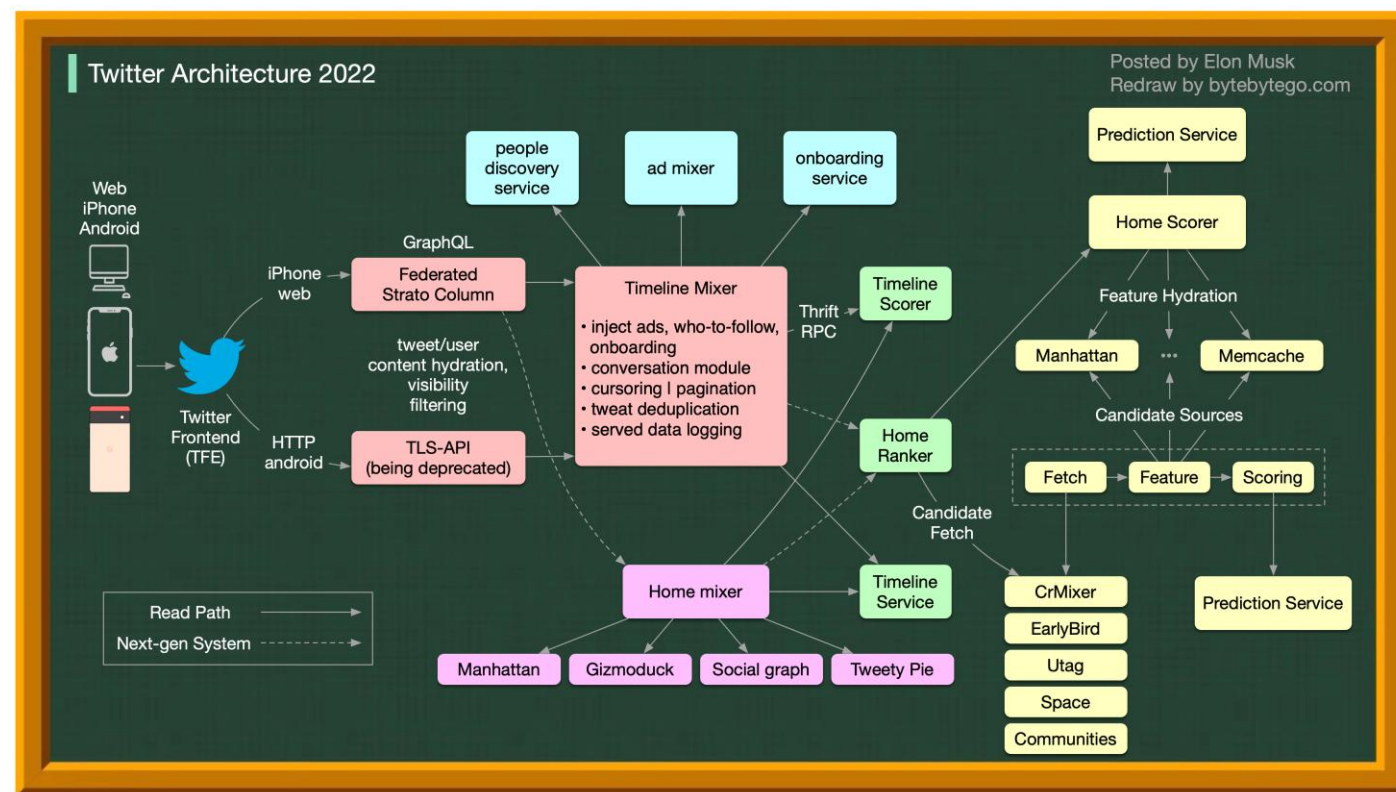


Diagrama de ByteByteGo

<https://x.com/alexxybyte/status/1594008281340530688>



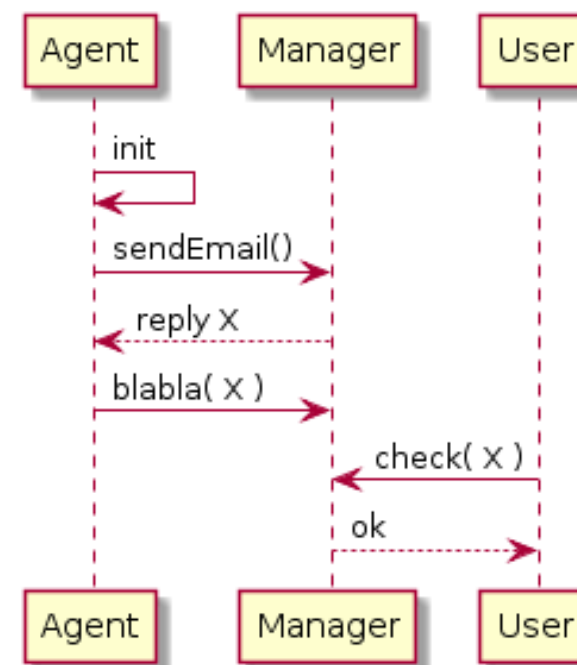
# Herramientas de diagramas basadas en texto

Normalmente basadas en UML

WebSequenceDiagrams, yUML, nomnoml

**PlantUML:** <http://plantuml.com/>

```
@startuml
Agent -> Agent : init
Agent -> Manager : sendEmail()
Agent <-- Manager : reply X
Agent -> Manager : blabla( X )
User -> Manager : check( X )
User <-- Manager : ok
@enduml
```



PlantUML Online: <https://www.planttext.com/>



# Herramientas de modelado

Permiten crear un modelo del Sistema software

Se generan representaciones visuals del modelo

Numerosas alternativas:

Enterprise Architect, **Visual Paradigm**, Archi, StarUML, ArgoUML, Modelio,...

Habitualmente permiten notaciones diferentes

UML, SysML, BPMN, ArchiMate

Útiles para diseño inicial *up-front*

Buenas para modificar y renombrar componentes

# Modelo mediante ingeniería inversa

Algunas de las herramientas anteriores de modelado permiten esta opción

Otras herramientas de análisis estático:

Structure101, NDepend, Lattix, Sonargraph,...

Crear el modelo a partir de código existente

Útil para visualizar bases de código existentes

Problema:

Los diagramas resultantes tienden a tener demasiados detalles

Difícil recuperar/visualizar la arquitectura

# Lenguajes de descripción de arquitecturas: ADLs

Definen formalmente la arquitectura de un sistema

Crear descripciones textuales en lugar de diagramas

Especificación formal

Describe la estructura y el comportamiento

Principalmente en entornos académicos

No muy populares en entornos industriales

Ejemplos:

xArch/xADL (<http://isr.uci.edu/projects/xarchuci/>)

ACME (<http://www.cs.cmu.edu/~able/>)

AADL (<http://www.aadl.info/>)

# Plantillas documentación arquitectura software

Varias posibilidades

Vistas 4+1 de Kruchten

Vistas y más

Modelo C4

Plantillas Arc42



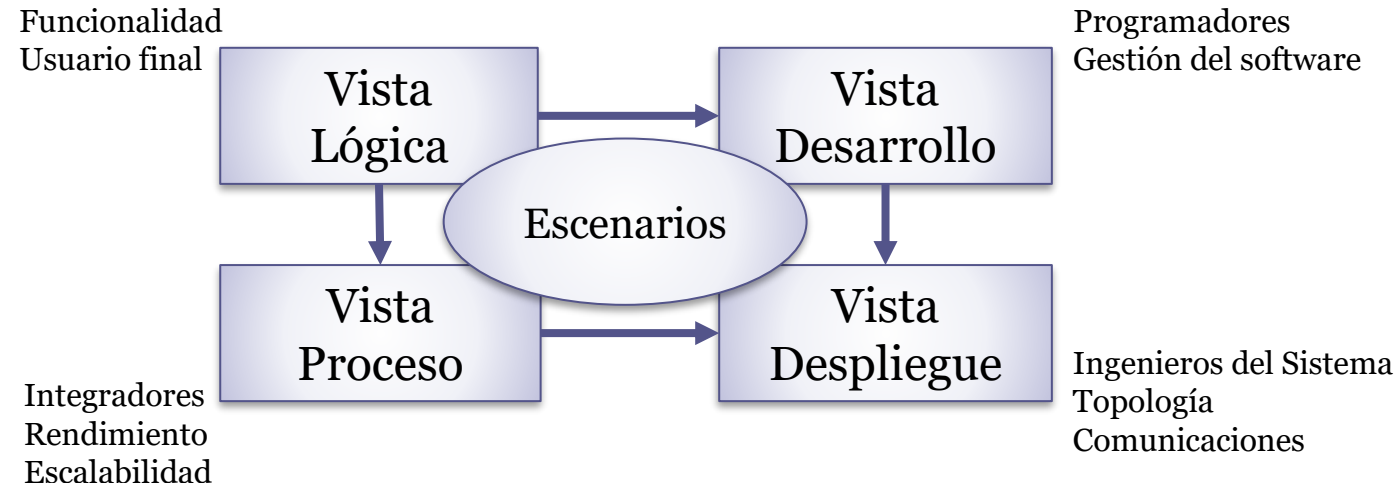
. . .

# Vistas 4+1 de Kruchten

Adoptadas en el *Rational Unified Process*

5 vistas

- 1 Vista **Lógica**: funcionalidad del sistema
- 2 Vista de **Desarrollo**: módulos, capas,...
- 3 Vista de **Proceso**: unidades de ejecución, concurrencia...
- 4 Vista **Física**: Topología de infraestructura y desarrollo
- (+1) Vista de **Escenarios**: casos de uso o escenarios



# Vistas y más (Views & beyond)

Seleccionar un conjunto de puntos de vista

De acuerdo a las necesidades de los stakeholders

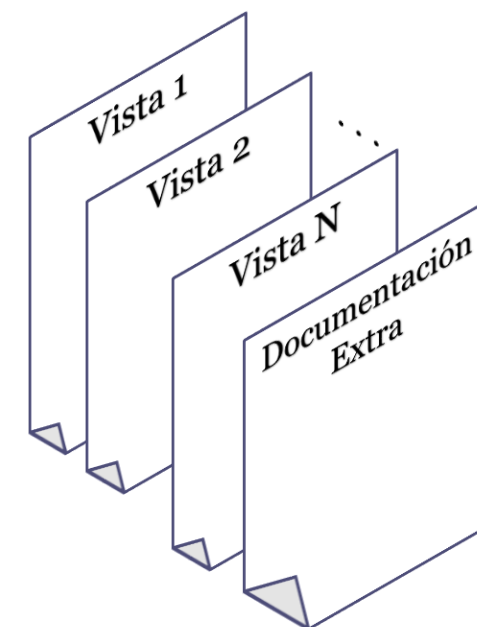
Definir vistas de acuerdo a los puntos de vista

Añadir un documento "Extra"

Arquitectura general

Información sobre cómo se relacionan las vistas

...



# Modelo C4 (<https://c4model.com/>)

## Describir

**Contexto:** Diagrama de contexto o empresarial

**Contenedores:** Estructura de alto nivel

**Componentes:** zoom y descomponer

**Código:** Diagramas de clases UML, diagramas ER, ...

## Guía de documentación

Contexto

Resumen funcional

Atributos de calidad

Restricciones

Principios

Código

Datos

Infraestructura de la arquitectura

Despliegue

Entorno de desarrollo

Operaciones y soporte

Log de decisiones

# Arc42 <https://arc42.org/>

Estructura para documentar sistemas de software

Objetivo: Claro, simple y efectivo

Plantillas disponibles en varios sistemas

**AsciiDoc**

Word (docx)

Markdown

LaTeX

ReStructuredText

Confluence

. . .



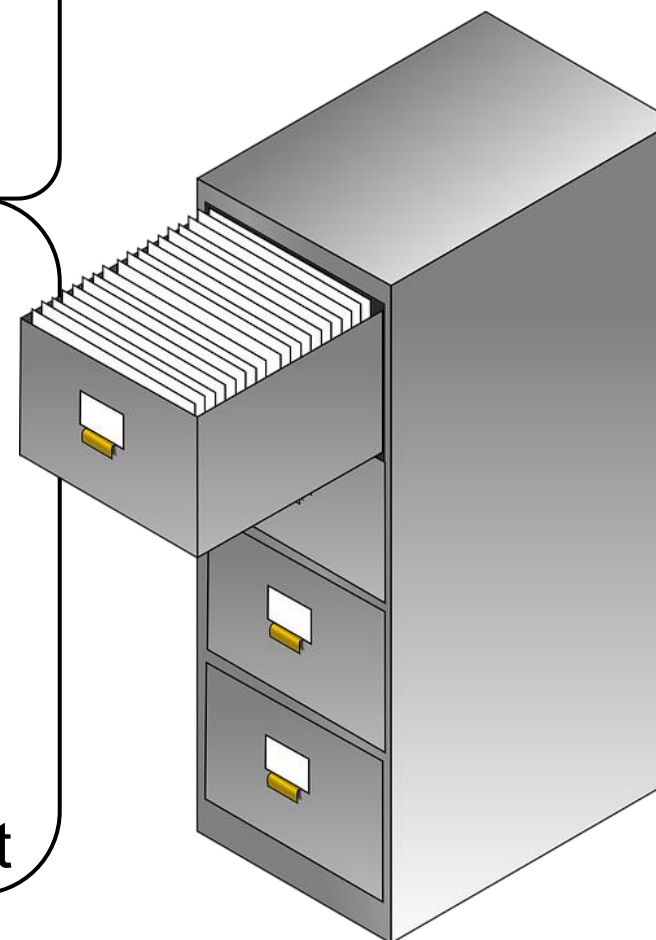
# Resumen Arc42

Problema

- 1.- Introduction and goals
- 2.- Constraints
- 3.- Context & scope

Solución

- 4.- Solution strategy
- 5.- Building block view
- 6.- Runtime view
- 7.- Deployment view
- 8.- Crosscutting concepts
- 9.- Architectural decisions
- 10.- Quality requirements
- 11.- Risks and technical debt
- 12.-Glossary



# 1 - Introduction and goals

Descripción breve de:

- Requisitos
- Principales objetivos de calidad
- *Stakeholders*



# 1 Introduction and goals

## 1.1 Requirements overview

Breve descripción de requisitos funcionales

Formato:

- Tablas con casos de uso

- Pueden enlazarse a documentos de requisitos ya existentes

  - Los documentos de requisitos suelen ser más largos

  - Elegir requisitos funcionales **significativos** para la arquitectura

# 1 Introduction and goals

## 1.2 Main quality goals

Enumerar objetivos de calidad principales

Objetivos de calidad:

Principales atributos de calidad que se deben alcanzar

Formato: Una tabla simple puede servir

Ejemplo:

[https://biking.michael-simons.eu/docs/index.html#\\_quality\\_goals](https://biking.michael-simons.eu/docs/index.html#_quality_goals)

# ¿Cómo elegir atributos de calidad?

## Talleres de atributos de calidad

Involucrar *stakeholders* para dar prioridad a atributos de calidad

Puede ser de ayuda distinguir entre:

Atributos de calidad en tiempo de ejecución

Rendimiento, seguridad, disponibilidad, usabilidad,...

Atributos de calidad en tiempo de no-ejecución

Modificabilidad, portabilidad, reusabilidad, testabilidad

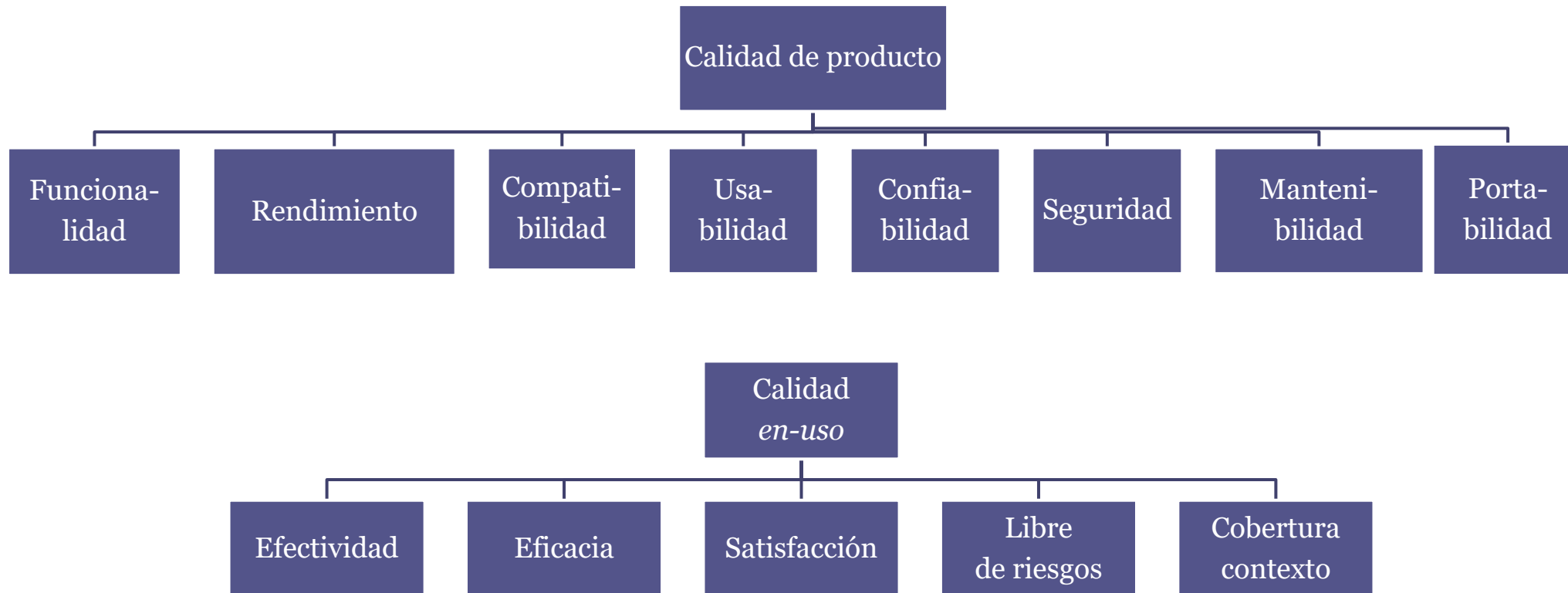
Atributos de calidad orientados al negocio

Coste, planificación, *time-to-market*, ...

# ¿Cómo elegir atributos de calidad?

ISO-25010 Modelo de calidad de Software

2 partes: Calidad de Producto, Calidad en-uso



# 1 Introduction and goals

## 1.3 Stakeholders

*Stakeholder*: persona que afecta, es afectada o puede contrubuir al sistema y su arquitectura

*Persona interesada*

Tratar de identificar y hacer explícitas las expectativas y motivaciones

Formato: tabla o mapa

Stakeholder	Descripción	Expectativas, motivaciones
...	...	...

## 2 - Constraints

Cualquier cosa que restringe las decisiones de diseño e implementación

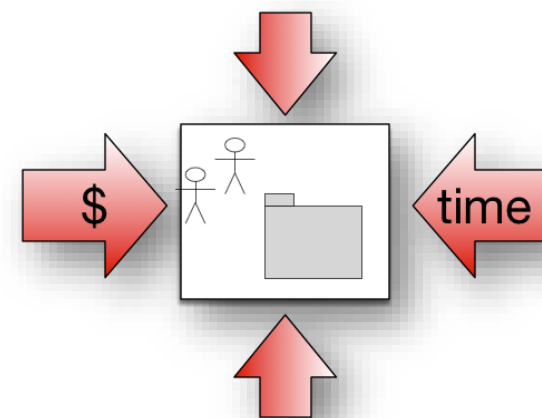
Algunas veces es a nivel de la organización

Decisiones ya tomadas

Formato: una tabla con explicaciones

Pueden dividirse en organizativas, técnicas, etc.

Constraint	Explanation
...	...





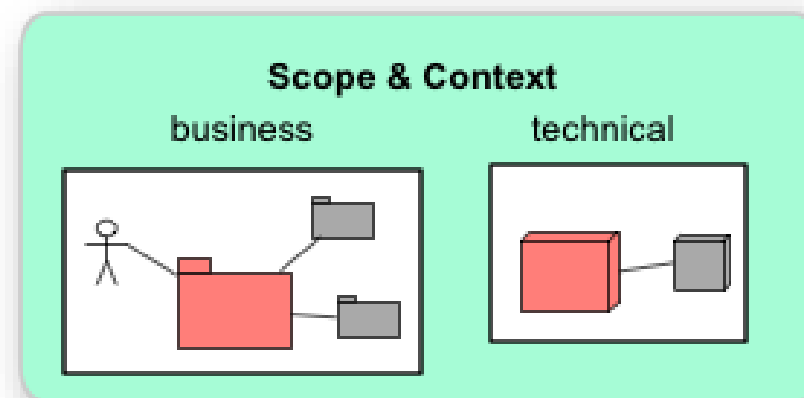
# 3 - Context and scope

Delimita el sistema de elementos externos

Usuarios y sistemas vecinos

Especifica los interfaces externos

Perspectiva de negocio y técnica



## 3. Context and Scope

### 3.1 Business context

Especificar los agentes involucrados en el entorno del sistema

Formato: Diagrama o tabla

Diagramas que muestran el sistema como caja negra

Opcional: Explicación de interfaces externos

**NOTA**  
El contexto de negocio es obligatorio

## 3 Context and scope

### 3.2 Technical context

Especificar interfaces técnicos que enlazan el sistema con su entorno

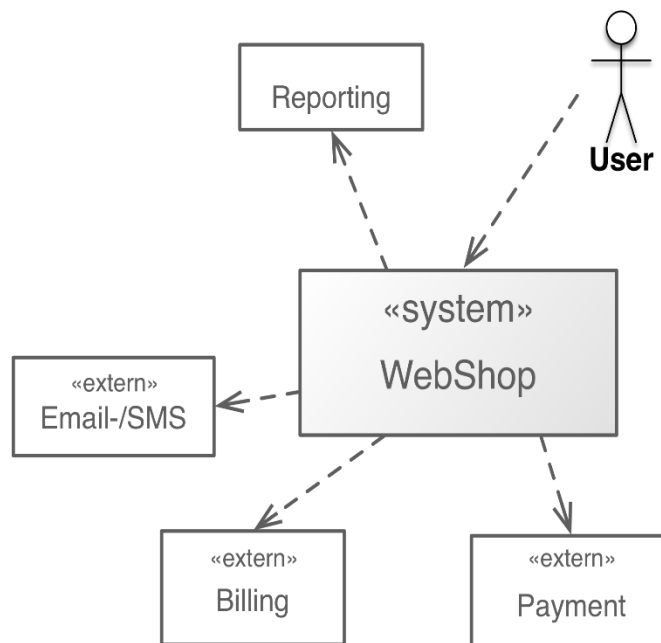
Formato: Diagrama o tabla

Normalmente: Diagramas despliegue UML

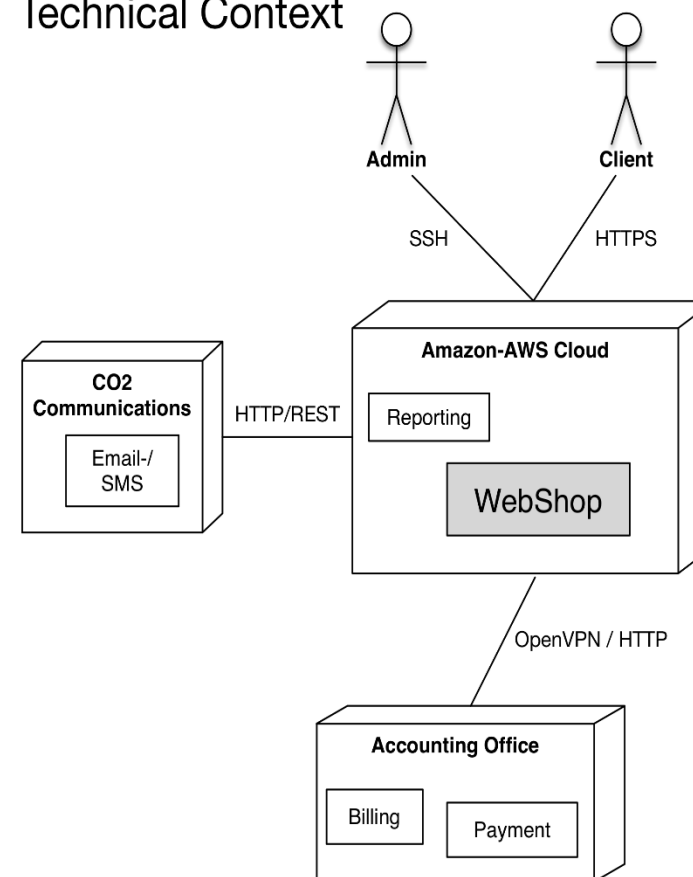
**NOTA**  
El diagrama de contexto técnico es opcional  
Muchas veces puede bastar con la vista de despliegue

# Contexto de negocio vs Contexto técnico

## Business Context



## Technical Context



# 4 - Solution strategy

Resumen de estrategias y decisiones fundamentales

Puede incluir:

- Tecnología
- Descomposición de alto nivel
- Enfoques para alcanzar objetivos de calidad
- Decisiones organizativas relevantes

Formato: Descripción textual breve

Mantener explicaciones breves de las decisiones



Picture source:

[https://commons.wikimedia.org/wiki/File:Light\\_Bulb\\_or\\_Idea\\_Flat\\_Icon\\_Vector.svg](https://commons.wikimedia.org/wiki/File:Light_Bulb_or_Idea_Flat_Icon_Vector.svg)

# 5 - Bulding block view

Descomposición estática del sistema

Módulos del sistema

Jerarquía de cajas blancas que contienen cajas negras

Formato:

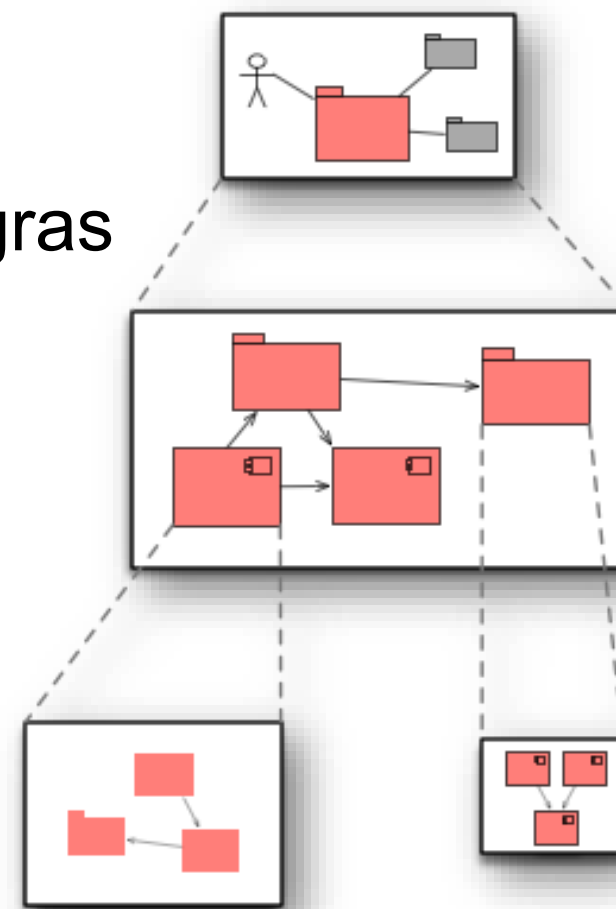
Comenzar con diagrama general

Caja blanca

Descomponer en otros diagramas

Normalmente se utilizan

Diagramas de componentes UML



# 6 - Runtime view

Comportamiento de bloques de construcción mediante escenarios

Casos de uso o características importantes

Interacciones con interfaces externos críticos

Comportamiento de Error y excepciones

Formato:

Varias notaciones

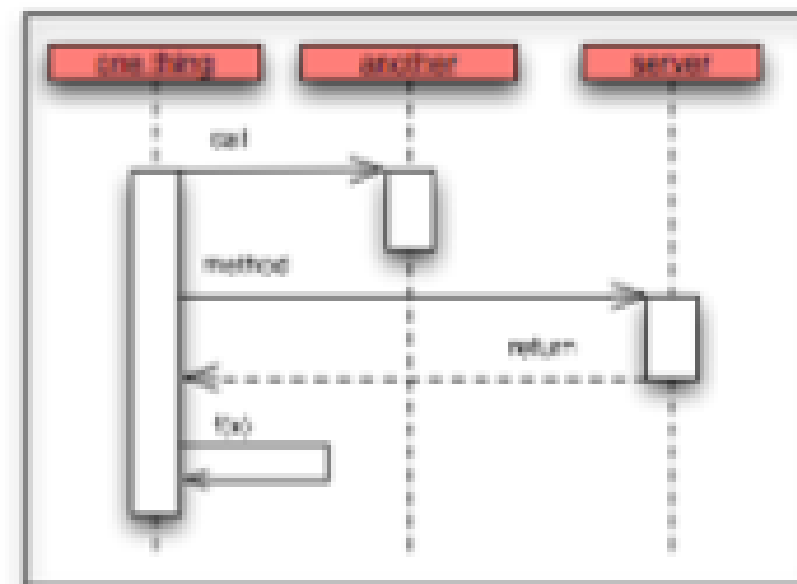
Lenguaje natural (lista de pasos)

Diagramas de secuencia UML

Diagramas de flujo (*flowcharts*)

BPMN

...



# 7 - Deployment view

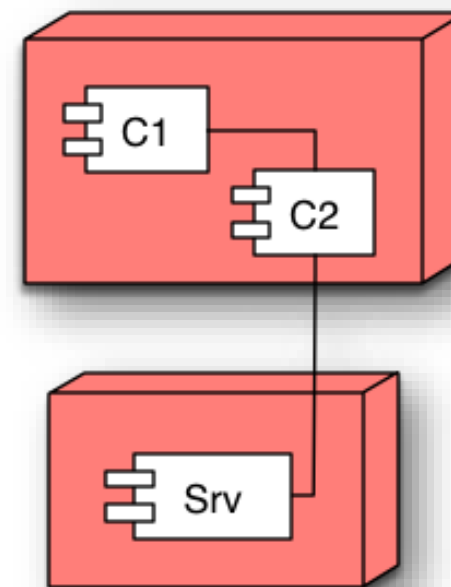
Infraestructura técnica con el entorno, ordenadores, procesadores, topologías, etc...

Asocia bloques de construcción (software) a la infraestructura

Formato:

Normalmente diagramas despliegue UML

Añadir tablas de asociación





## 8 - Crosscutting concepts

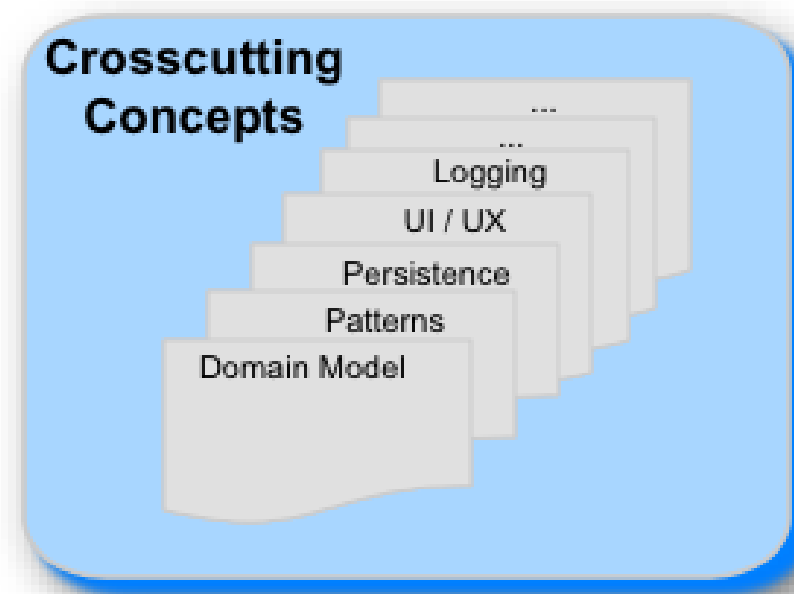
Enfoques relevantes en múltiples partes del sistema

Se pueden incluir tópicos como:

Modelo de dominio

Estilos y patrones

Reglas específicas



## 9 Architectural decisions

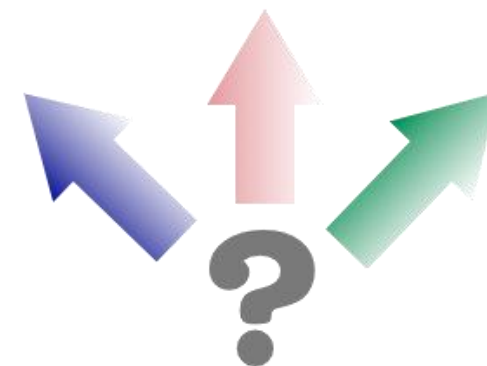
Decisiones importantes, de alto coste, críticas, de gran escala o arriesgadas

Incluir justificación de las decisiones

Formato:

- Lista o tabla ordenada por importancia

- Registro de decisión arquitectónica para decisiones importantes

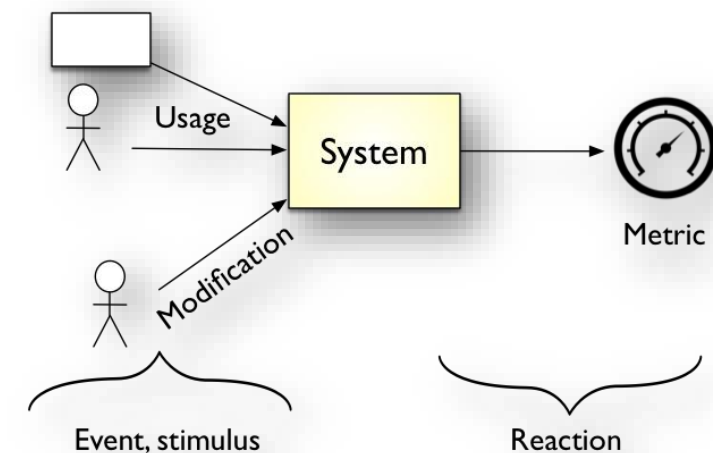


# 10 - Quality requirements

## Requisitos de calidad como escenarios

Árbol de calidad para proporcionar visión de alto nivel

Los objetivos de calidad más importantes deberían ser descritos en la sección 1 (*quality goals*)



# 10. Quality requirements

## 10.1 Quality tree

Un árbol de calidad con escenarios de calidad como hojas

Incluir prioridades para una visión general

En ocasiones el número de requisitos de calidad puede ser elevado

Formato:

Un *mind-map* con categorías de calidad en las ramas

Incluir enlaces a escenarios de la siguiente sección

# 10. Quality requirements

## 10.2 Quality scenarios

Los escenarios describen qué debería ocurrir cuando llega un estímulo al sistema

2 tipos:

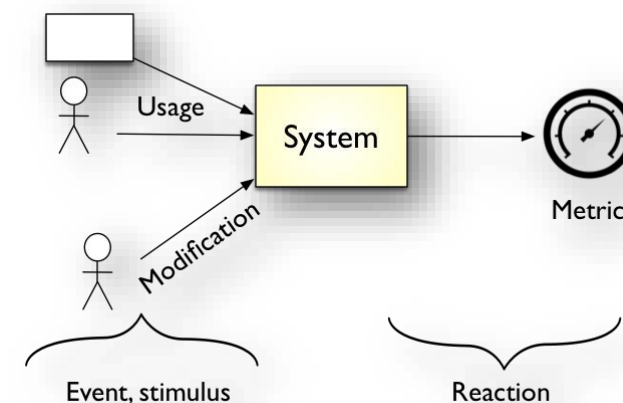
Uso: reacción en tiempo de ejecución ante cierto estímulo

*"El Sistema debe reaccionar a la petición X en 1 segundo"*

Cambio: Modificación del Sistema o su entorno

*"Un formato nuevo puede añadirse en 4h"*

Formato: Tabla o texto libre.



# 11 - Risks and technical debt

Riesgos conocidos o deuda técnica

¿Qué problemas potenciales existen?

¿Qué hace que el equipo de desarrollo se sienta mal?

Formato:

Lista de riesgos o deuda técnica

Incluir medidas sugeridas para minimizar, mitigar o evitar riesgos,  
o reducir la deuda técnica.



# 12 - Glossary

Términos importantes del dominio o técnicos

Términos utilizados por los *stakeholders* cuando discuten sobre el sistema

Vocabulario común

Traducción de referencia en entornos multi-idioma

Formato: tabla

Term	Definition
...	...

# Estilo codificación arquitectónicamente evidente

Dejar pistas sobre la arquitectura en el código

Facilitar al lector inferir el diseño a partir del código

El código debe reflejar la arquitectura

Ejemplos:

Componentes como paquetes

Módulos en repositorios o carpetas diferentes

Herramientas para chequear/controlar restricciones arquitectónicas

<https://www.lattix.com/>

<https://structurizr.com/>