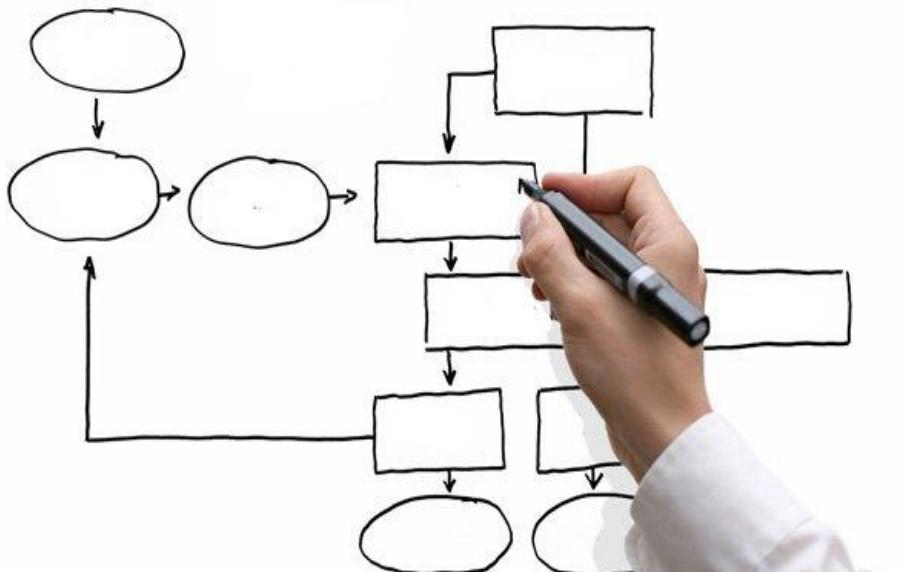




ugr

Universidad
de Granada



APLICACIÓN DE PATRONES DE DISEÑO AL DESARROLLO DE SOFTWARE ÁGIL

Autor: Christian Andradas Molina
Director: Manuel Ignacio Capel Tuñón

ÍNDICE

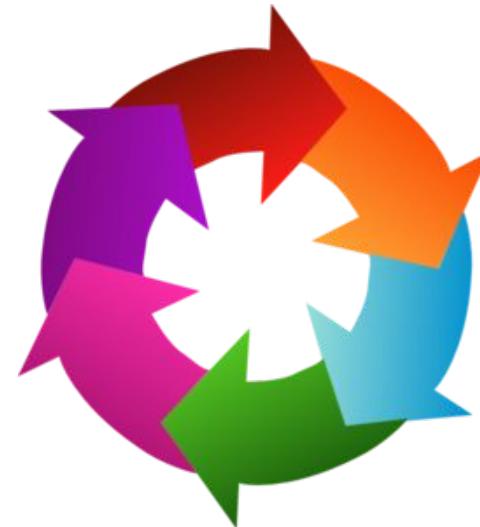
- 1. INTRODUCCIÓN**
- 2. METODOLOGÍAS ÁGILES Y ARQUITECTURA DE SOFTWARE**
- 3. PATRONES DE DISEÑO**
- 4. DISEÑO Y METODOLOGÍAS ÁGILES**
- 5. PROPUESTA DE DISEÑO ÁGIL**
- 6. CONCLUSIONES**

1. INTRODUCCIÓN

¿Qué es un desarrollo ágil?

Métodos de ingeniería del software basados en:

- Desarrollo iterativo e incremental.
- Cara a cara > documentación.
- Priorizar software funcional.
- Iteraciones descompuestas en:
 - Planificación.
 - Análisis de requisitos.
 - Diseño.
 - Codificación.
 - Pruebas.
 - Documentación.



Actualmente, el desarrollo de sistemas software complejos se realiza mediante la aplicación de un conjunto de metodologías ágiles.



Sin embargo, existe un recelo en el uso de patrones de diseño en el desarrollo de software ágil.



Se propone una metodología/fase de diseño:

- Compatible con las características ágiles.
- Uso de patrones arquitectónicos.
- Capacidad de integrarse en una metodología ágil existente.



2. METODOLOGÍAS ÁGILES Y ARQUITECTURA DEL SOFTWARE

Manifiesto ágil - 12 principios

- Entregas continuas funcionales.
- Adaptación a la variación de los requisitos iniciales.
- Motivación y confianza para alcanzar la excelencia.
- Cooperación y comunicación.
- Tareas sencillas.



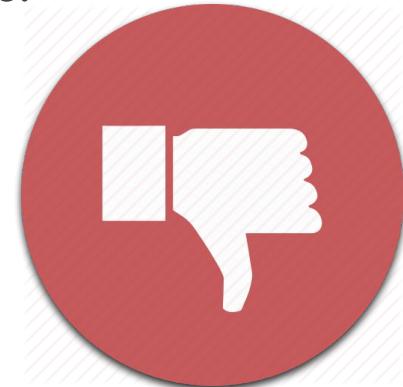
Ventajas

- Mayor implicación del cliente.
- Mayor seguimiento del proceso.
- Respuesta rápida a cambios.
- Reducción de costes y tiempo.
- Temporización realista.



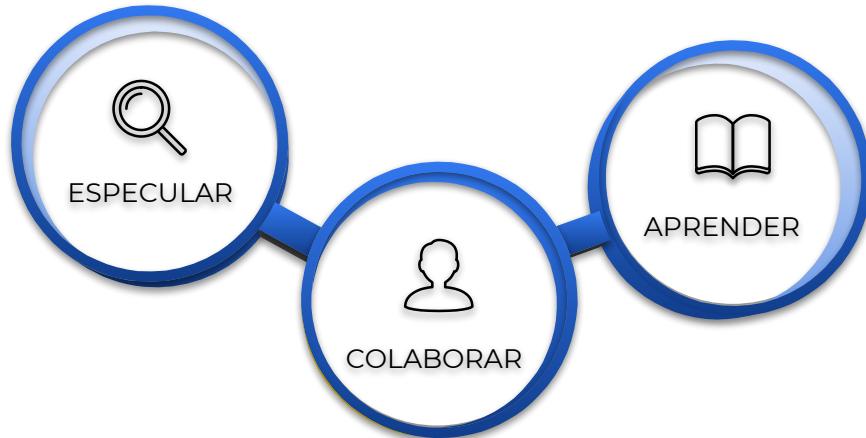
Desventajas

- Problemas de inexperiencia, comunicación o actitud.
- Incapacidad de seguir el ritmo.
- Exige participación continua de cliente.
- Síndrome del Burn Out.
- Falta de documentación.



Agile Unified Process (AUP)

ASD (Adaptive Software Development) es una técnica que fue desarrollada por Jim Highsmith y Sam Bayer a comienzos de 1990. El ciclo utilizado por ASD es conocido como:



Extreme Programming (XP)



La programación extrema es un enfoque de la ingeniería de software propuesto por Kent Beck. Es uno de los procesos ágiles más importantes del desarrollo de software.

SIMPLICIDAD

COMUNICACIÓN

REALIMENTACIÓN

VALENTÍA

Kanban

Kanban nació en 1948 a manos del ingeniero Taiichi Ohno en Japón, dentro de la compañía Toyota.

Kanban!

SEGUIMIENTO DEL TRABAJO

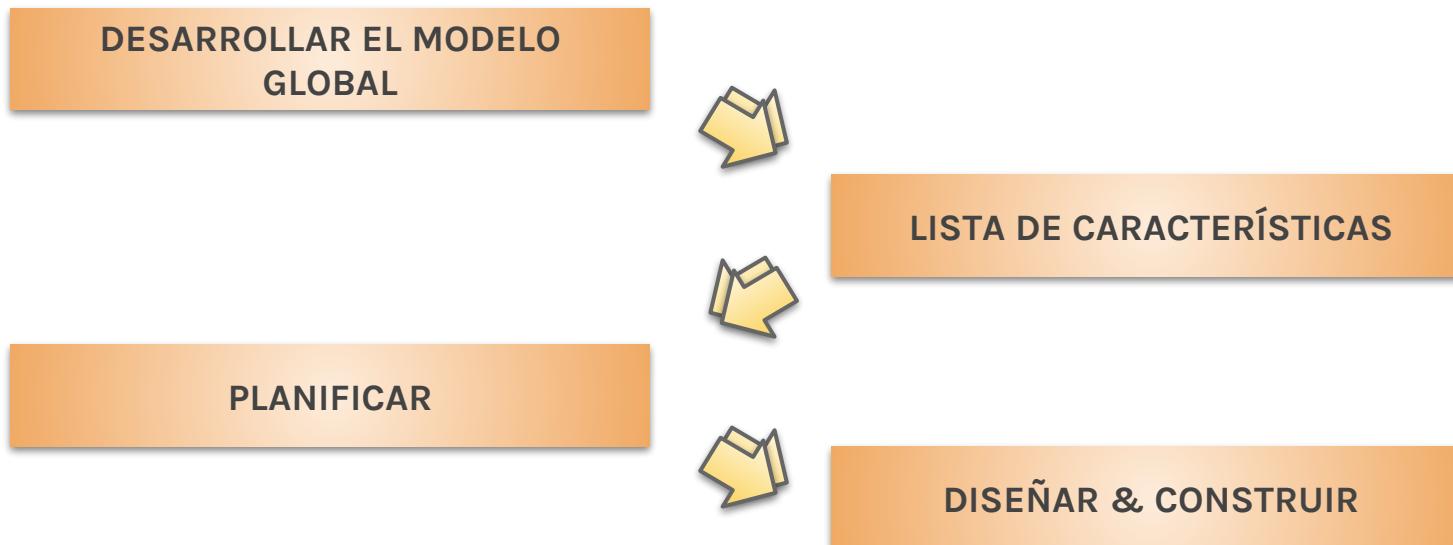
IDENTIFICAR CUELLOS DE BOTELLA

OBTENER INDICADORES VISUALES

Feature Driven Development (FDD)

FDD

FDD fue desarrollado por Jeff De Luca y Peter Coad y se enfoca en iteraciones cortas que van produciendo cada vez más funcionalidad tangible.



Lean Software Development (LSD)

LSD

El término Lean Software Development se acuñó por primera vez como título de una conferencia organizada por la iniciativa ESPRIT de la Unión Europea, en Stuttgart, Alemania, en octubre de 1992.

- Elimina código inútil.
- Potencia la participación.
- Aprendizaje continuo.

Open Unified Process (OpenUP)

OpenUP (Open Unified Process) es un método de desarrollo de software propuesto por un conjunto de empresas. Se encuentra bajo una licencia de software libre.



- Colaboración entre miembros.
- Equilibrio entre prioridades.
- Desarrollo evolutivo.

Dynamic Systems Development Methods (DSDM)

DSDM

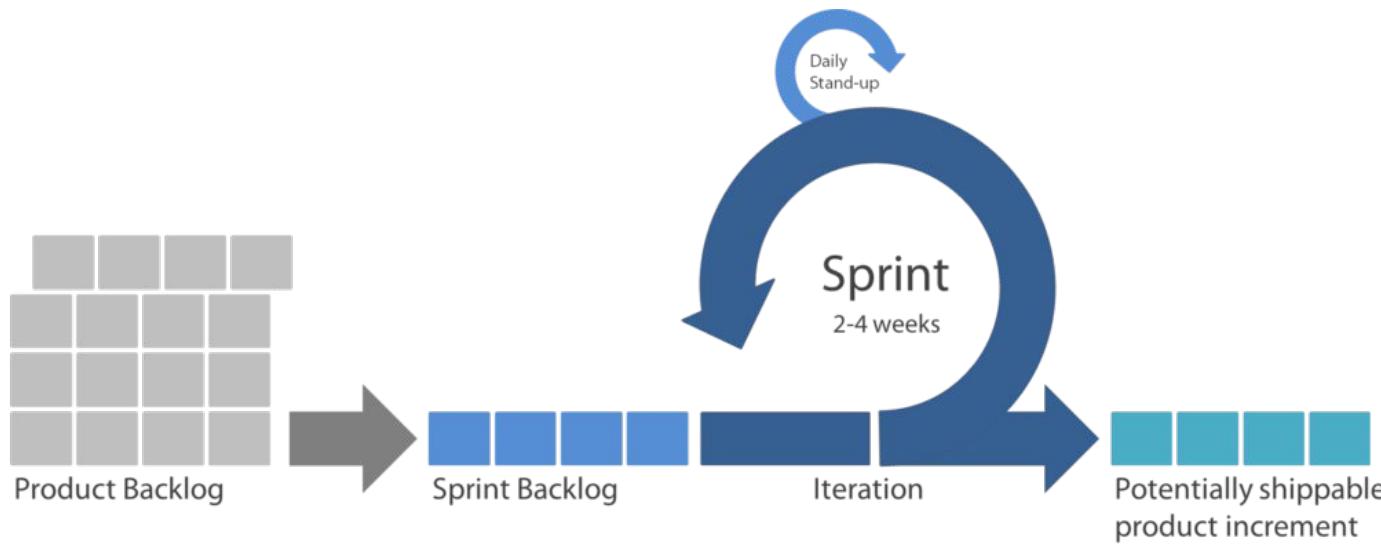
DSDM (Dynamic Systems Development Methods) es un método que provee un framework para el desarrollo ágil. Fue desarrollado en Reino Unido durante los años 90.

- El cliente como prioridad.
- Entregas a tiempo.
- Comunicación continua y fluida.
- Control sobre el desarrollo

SCRUM



El origen de Scrum se remonta a principios de los años 80, a partir de un estudio realizado en 1986 por Hirotaka Takeuchi y Ikujiro Nonaka en la obra 'The New New Product Development Game'.



Arquitectura de software

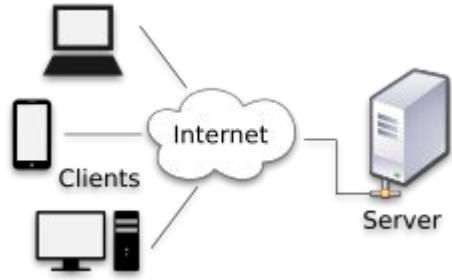
“La arquitectura de software es el conjunto de decisiones de diseño que, si se realizan incorrectamente, pueden provocar la cancelación de su proyecto.”



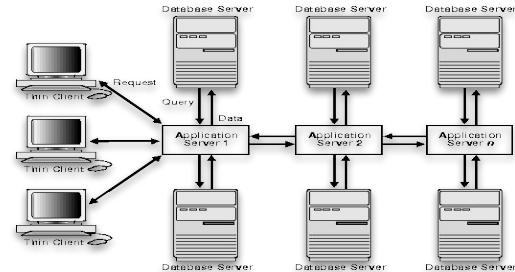
¿Por qué es importante?

- Ofrece la base de las decisiones.
- Constituye la mejor documentación.
- Predice los atributos de calidad.
- Define restricciones de implementación.
- Permite medir el tiempo y coste.
- Reduce la complejidad.

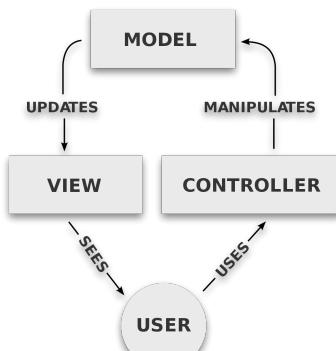
CLIENTE-SERVIDOR



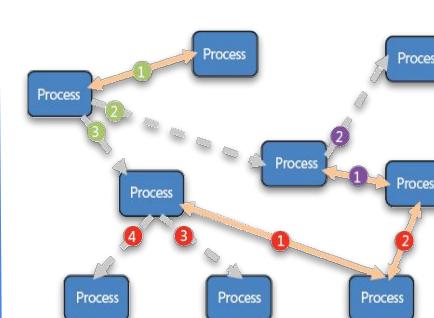
PROGRAMACIÓN POR CAPAS



MODELO VISTA-CONTROLADOR



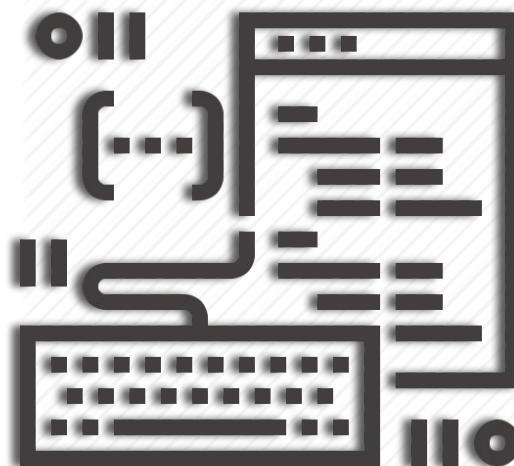
ORIENTADO A SERVICIOS - EVENTOS



3. PATRONES DE DISEÑO

Patrones de diseño

“Los patrones de diseño son unas técnicas para resolver problemas comunes en el desarrollo de software y otros ámbitos referentes al diseño de interacción o interfaces.”



Es el primer paso para abordar el diseño.



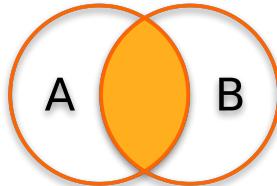
Se basan en los requisitos no funcionales.



Imponen ciertas reglas.



Tienen la capacidad de combinarse.

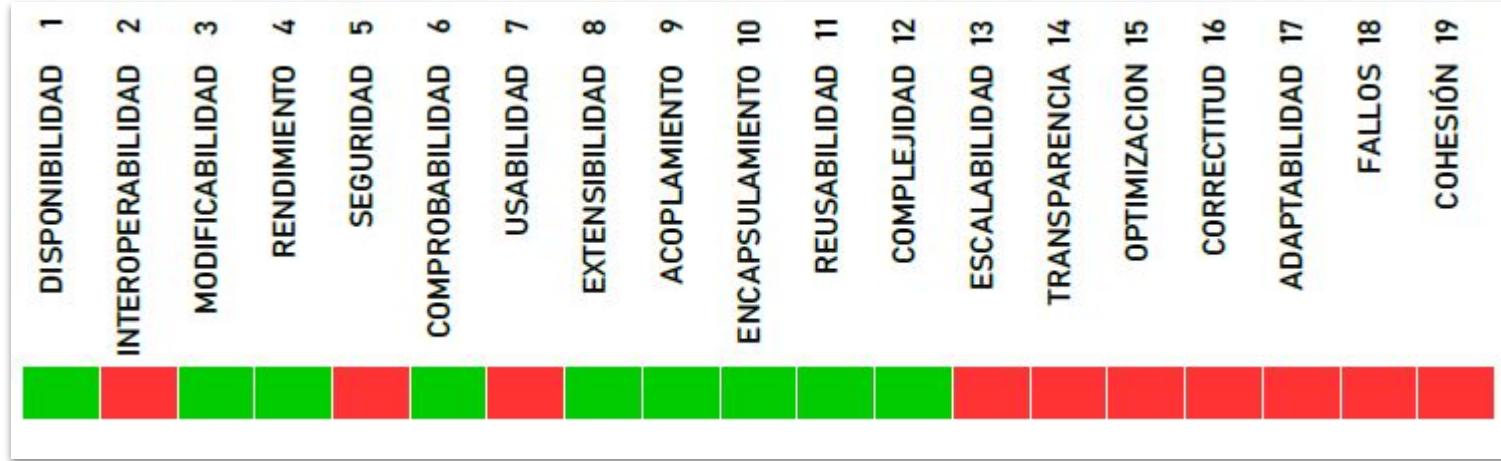


Proporcionan atributos de calidad.



Patrón Intercepto

Permite incluir diferentes servicios o procesos de forma transparente para que sean activados en el momento en el que ocurra un determinado evento. Se basa en la idea de incluir dinámicamente servicios por parte de las aplicaciones.



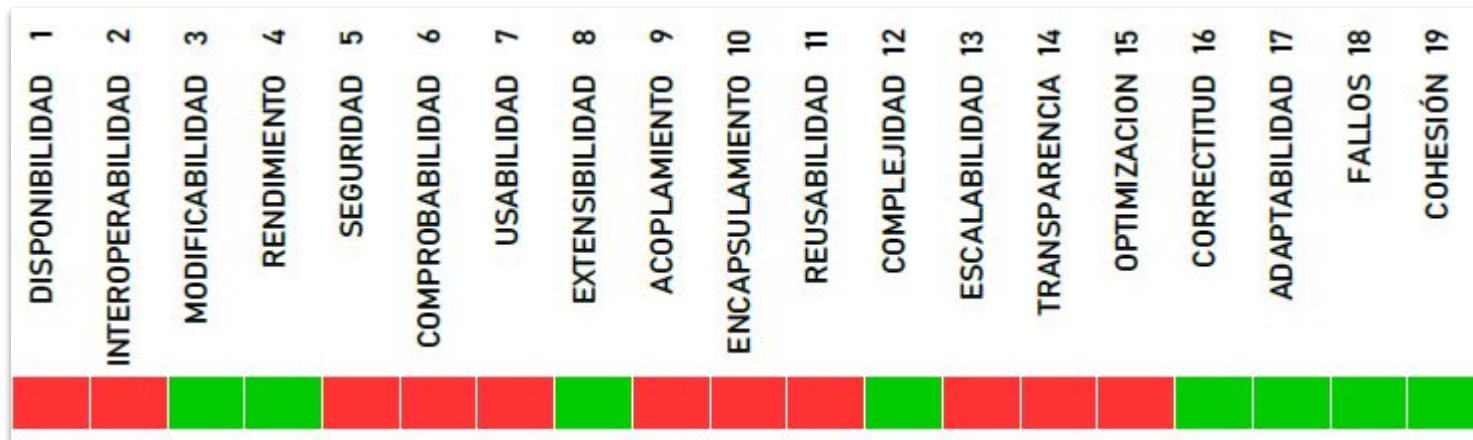
Patrón Broker

Ofrece la posibilidad de modelar sistemas distribuidos compuestos por componentes software desacoplados que interactúan mediante invocaciones a servicios remotos. Para ello el patrón coordina la comunicación, redireccionando peticiones y transmitiendo resultados y excepciones.



Patrón Reflection

Nos permite modificar la estructura y comportamiento de sistemas software de forma dinámica. En este patrón tenemos un nivel que proporciona información sobre propiedades del sistema y otro cuya responsabilidad principal consiste en incluir la lógica de la aplicación.



4. METODOLOGÍAS ÁGILES Y DISEÑO

¿Es necesario el diseño trabajando con metodologías ágiles?



Para unos ...

- No deben existir decisiones inamovibles y prematuras.
- Rompen con un desarrollo evolutivo.
- Implica mantener una documentación.
- Puede llevar a caminos erróneos.

¿Es necesario el diseño trabajando con metodologías ágiles?

Para otros ...

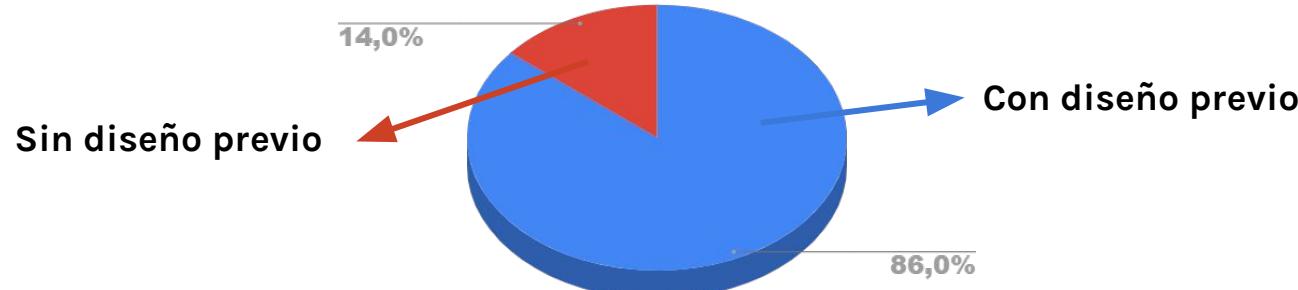
- Proporciona abstracción.
- La documentación mínima es necesaria.
- Permite alcanzar unos atributos de calidad.
- Implica una buena ingeniería.



5. PROPUESTA DE DISEÑO ÁGIL

Experiencias con el diseño ágil ...

- '**Agile Practices and Principles**' (2008) : valor positivo a un diseño inicial y detalles posteriores.
- '**Modeling and Documentation Practices on IT Projects Survey**' (2008) : el enfoque más popular era crear diagramas de alto nivel.
- '**Agile Project Initiation Survey**' (2009) :



Propuesta #1

Scaled Agile propone el framework 'SAFE' (Scaled Agile Framework) :

- El diseño emerge como una colaboración.
- Más grande es el sistema, más largo es el camino.
- Arquitectura más simple que lleve al sistema objetivo.
- En caso de duda, codificarlo o protegerlo.
- Ellos lo construyen, ellos lo prueban.
- No hay monopolio en la innovación.
- Implementar el flujo arquitectónico.

SCALED AGILETM

Propuesta #2

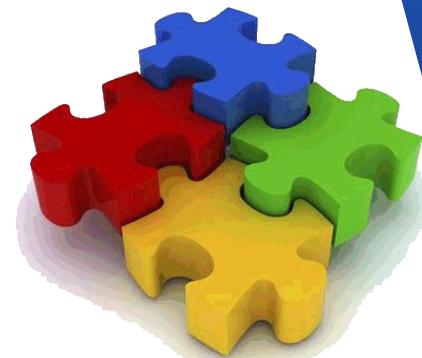
La empresa 'Scott Ambler + Associates' propone:

- Rol del arquitecto a más de una persona.
- Encargado del diseño = resto del equipo.
- Integrar el trabajo del diseño dentro del desarrollo.
- Modelar.
- Tener en cuenta las limitaciones.

**SCOTT AMBLER
+ Associates**

Principios básicos

- Un solo arquitecto para definir la arquitectura o todos.
- El progreso lo ofrece el software.
- El arquitecto asume sus responsabilidades.
- Usar SAAM.
- Dedicarse a las decisiones arquitectónicas importantes.



Principios en el ámbito organizativo

- Promover equipos autoorganizados.
- Compartir las decisiones conflictivas.
- Grupos de arquitectura verticales.
- Usar un lenguaje de alto nivel.
- La arquitectura como un servicio.



Principios en el ámbito personal

- Motivación.
- Mejora personal.
- Confianza en tus compañeros.
- Uso de patterns.



Principios de calidad

- Definición formal de los requisitos de calidad.
- Simplificar.
- Buscar la excelencia.
- Testear los requisitos de calidad.

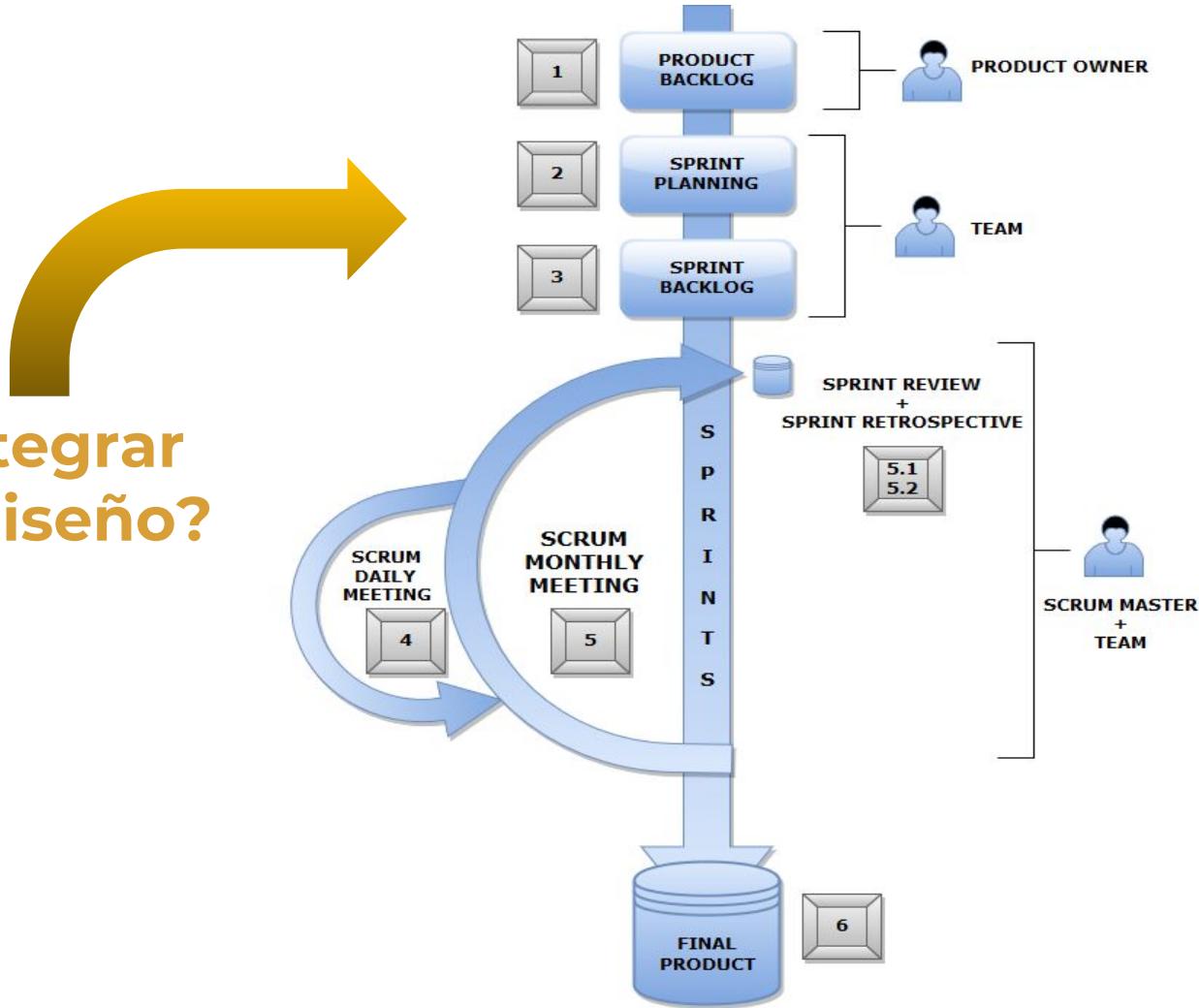


Principios de validación y control

- Utilizar la taxonomía de los riesgos operativos del SEI.
- Uso de prototipos y tracer bullets.



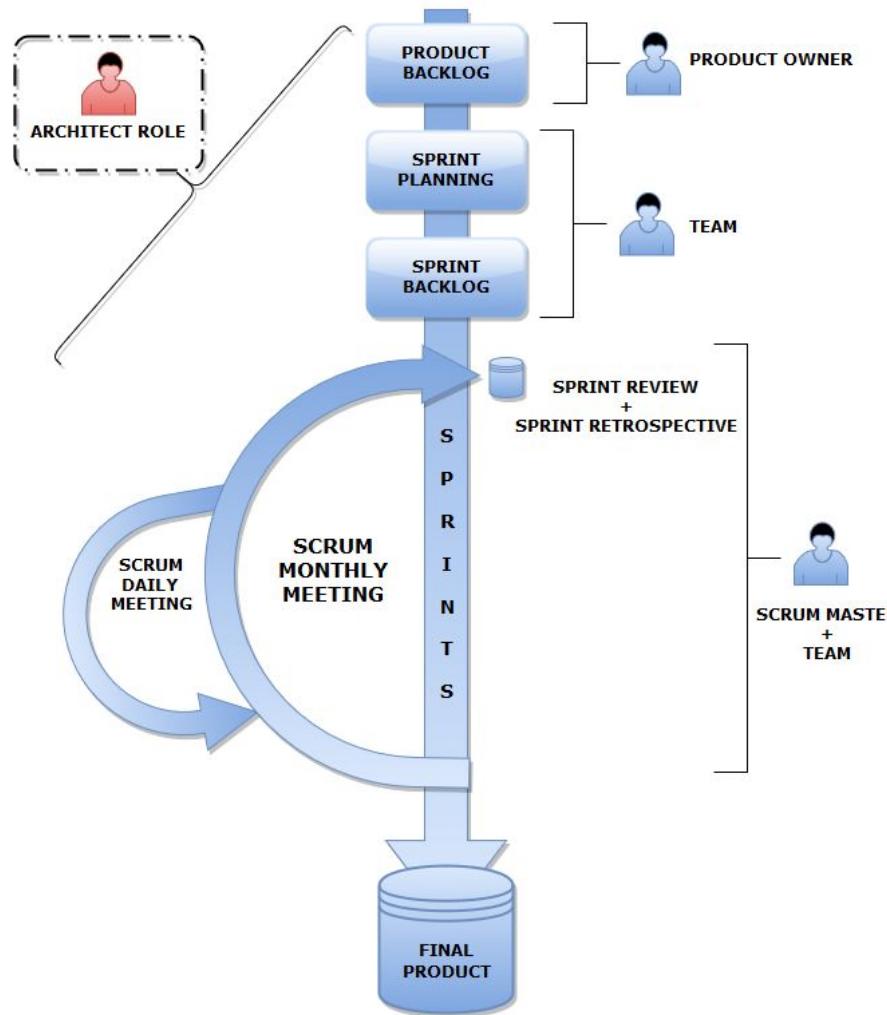
¿Integrar
el diseño?



Solución #1

SCRUM

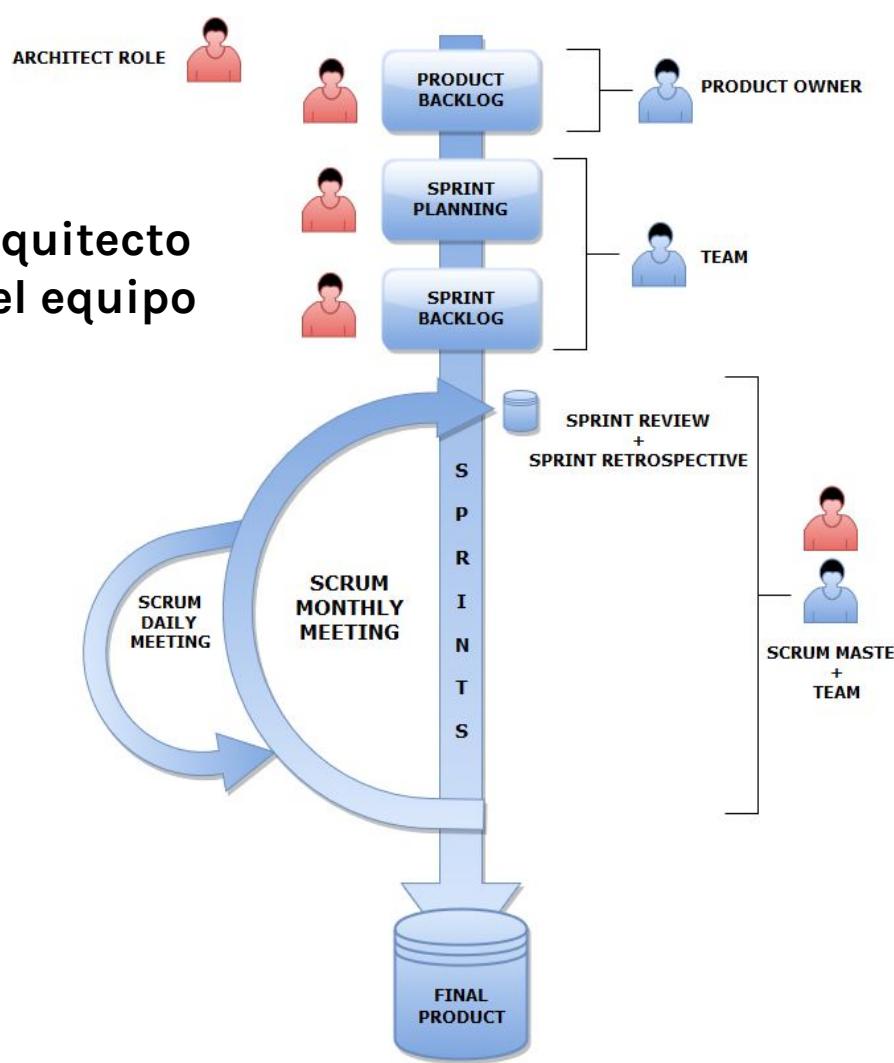
Rol de
arquitecto
externo al
equipo Scrum



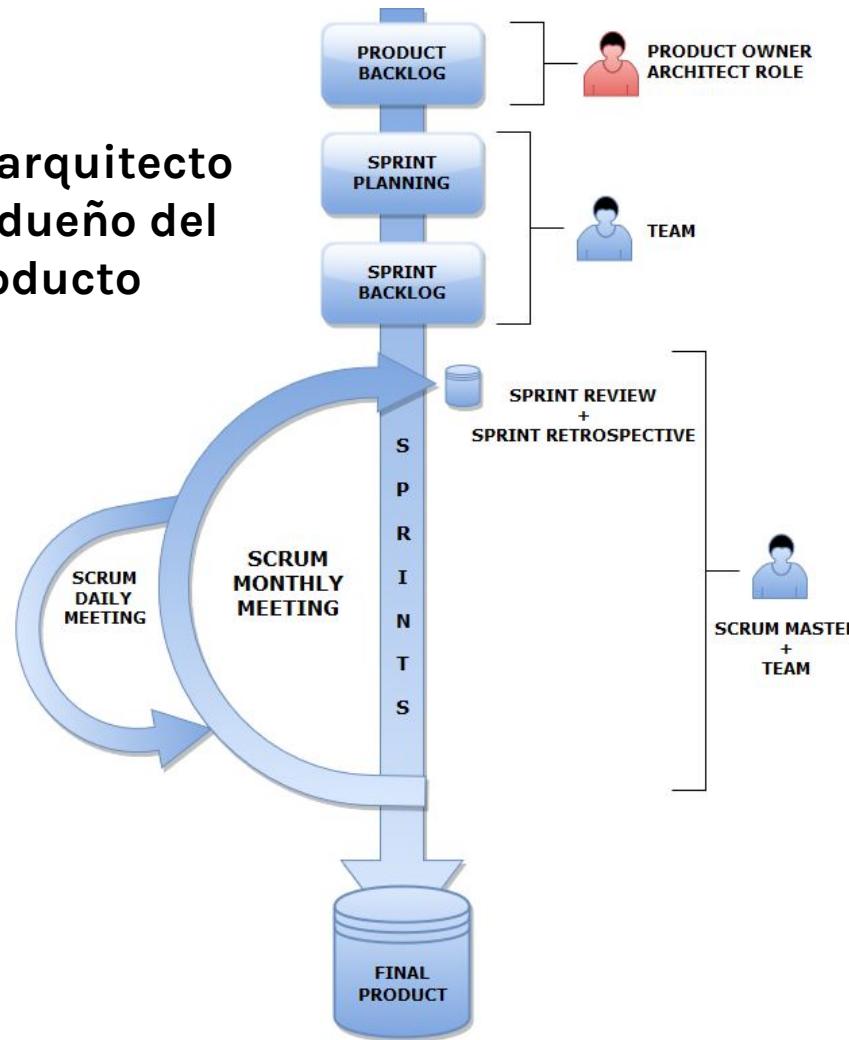
Solución #2

SCRUM

Rol de arquitecto dentro del equipo



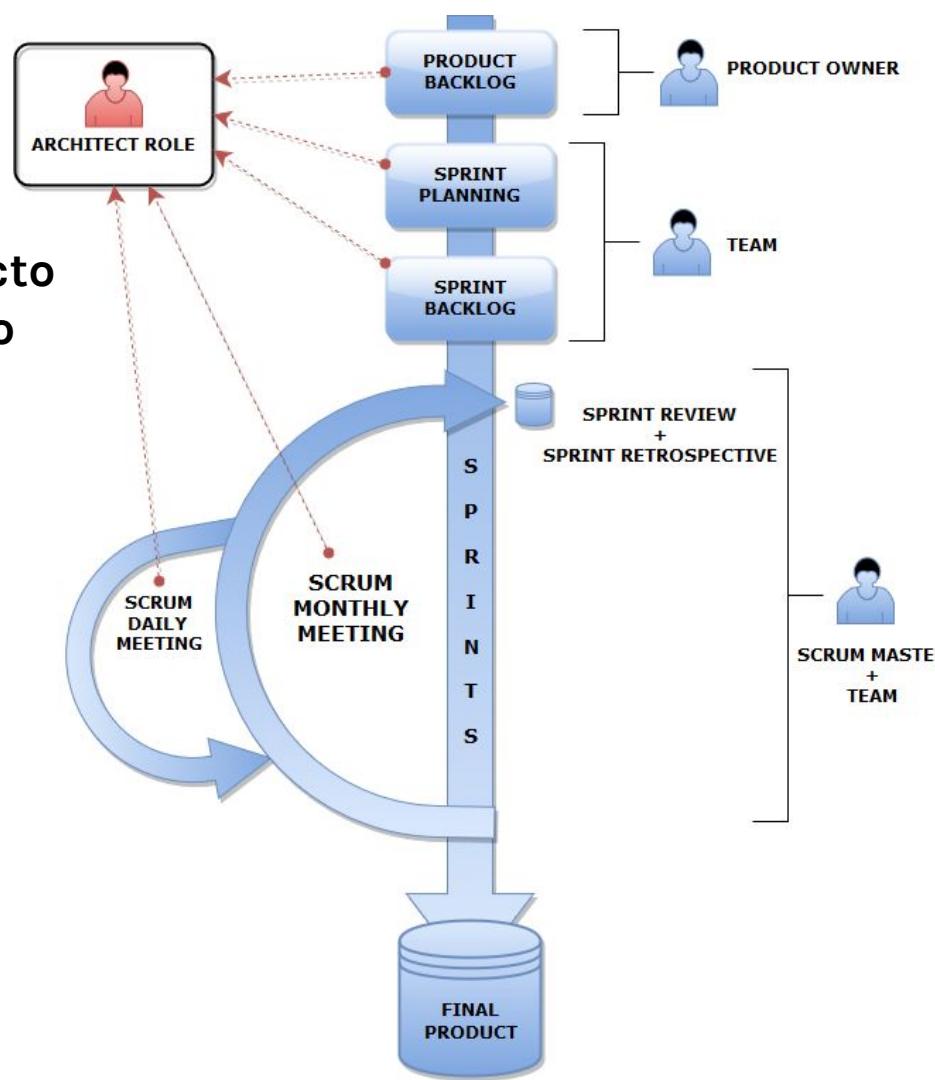
Rol de arquitecto como dueño del producto



Solución #4

SCRUM

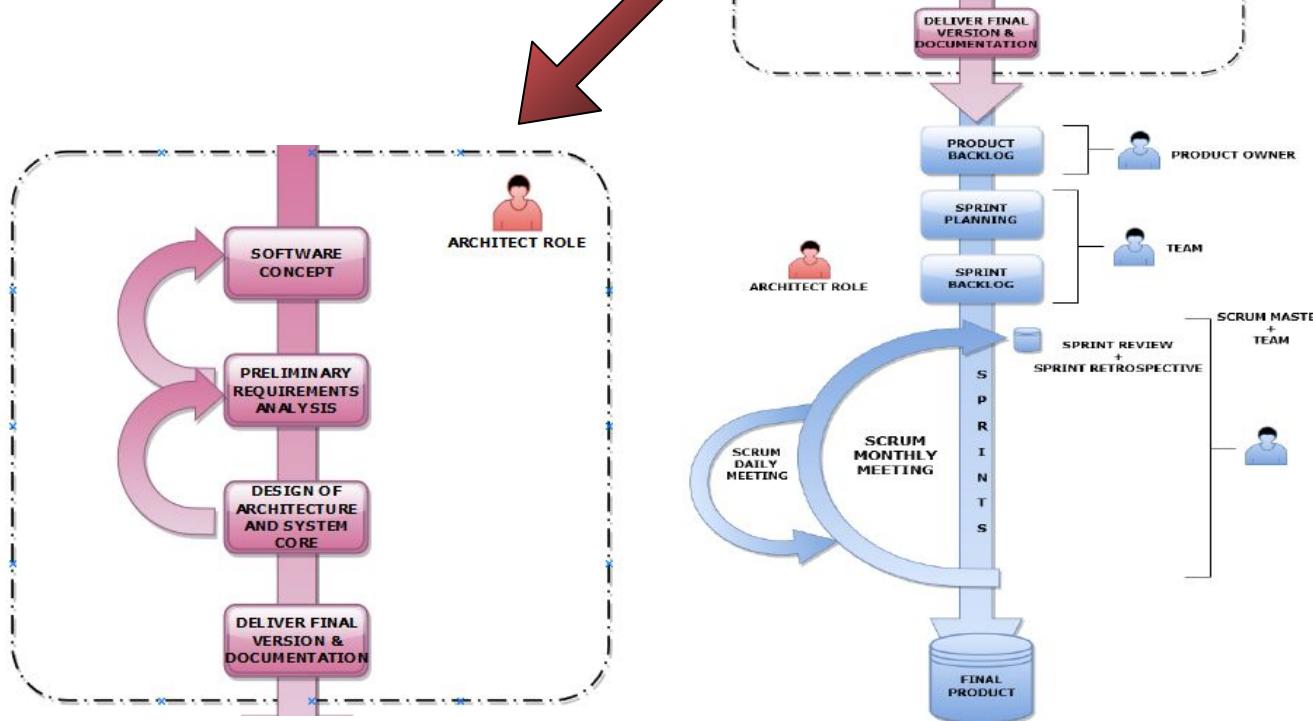
Rol de arquitecto
como recurso
externo



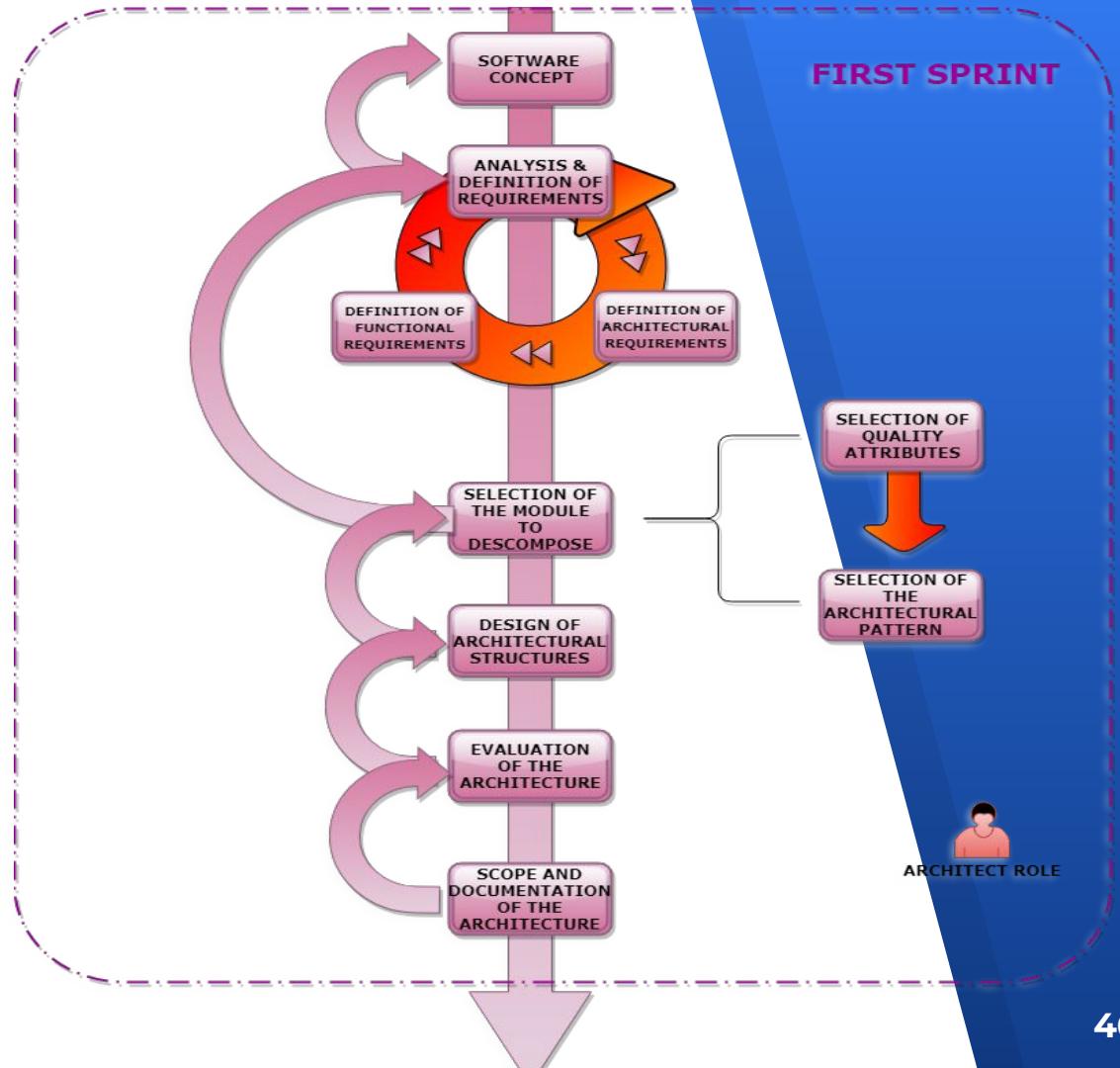
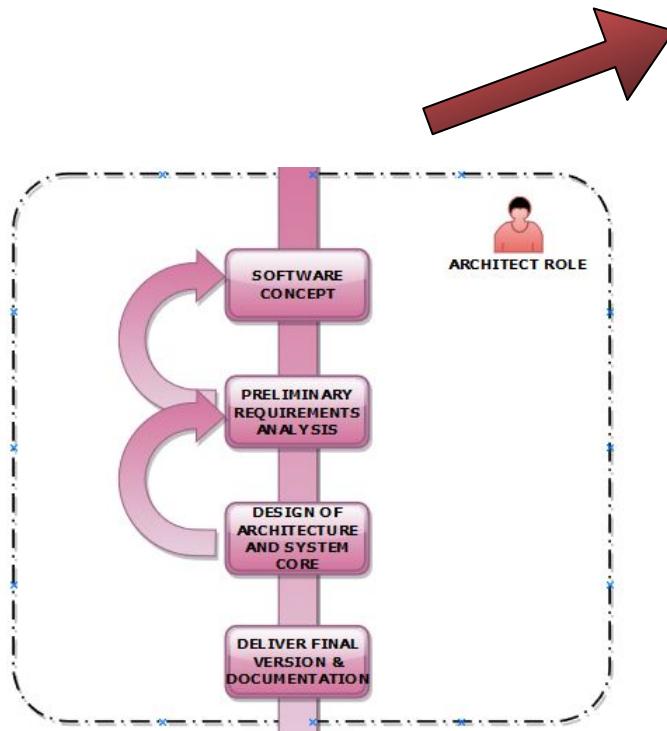
Solución #5

SCRUM

Sprint 0

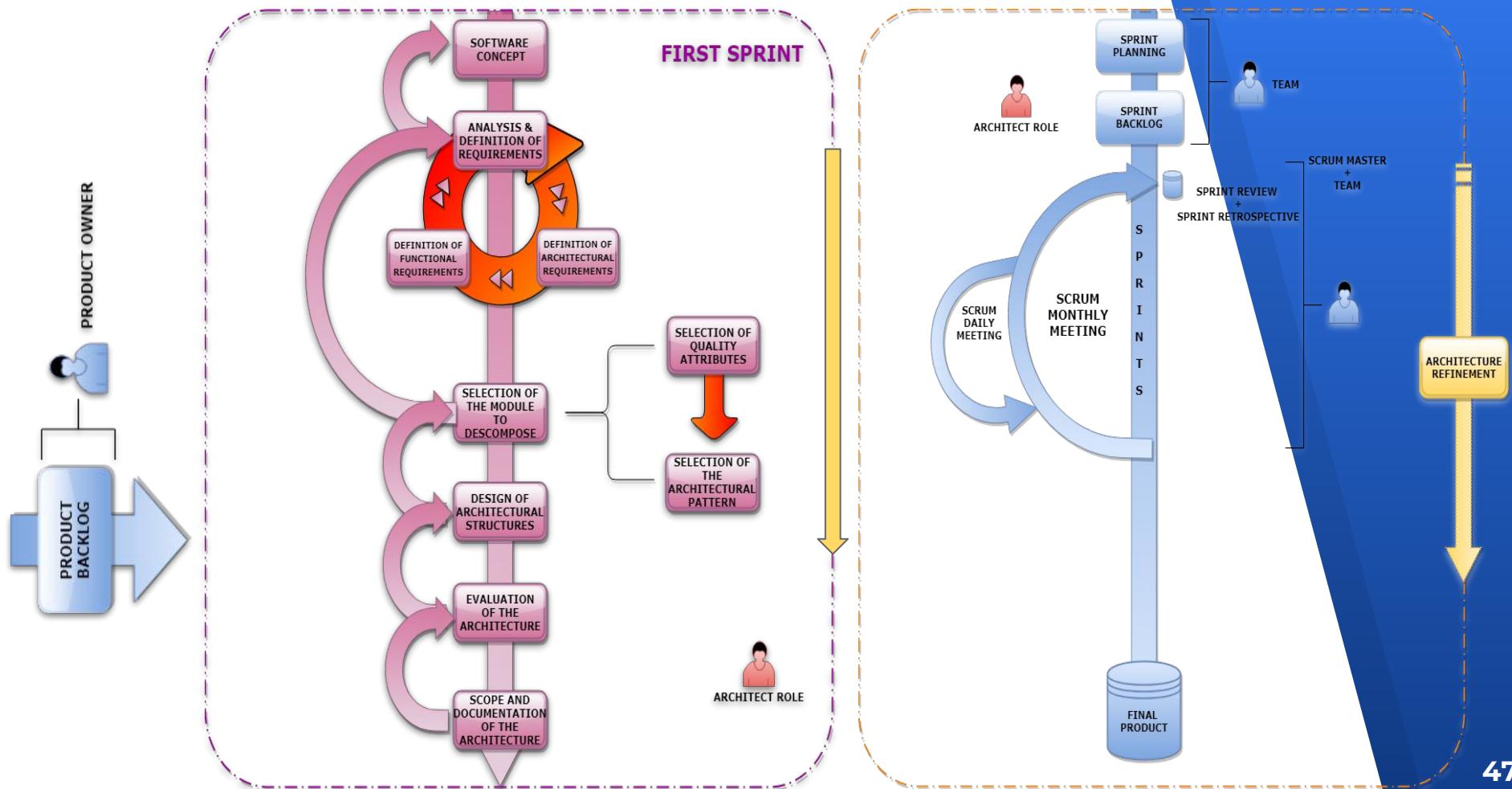


FIRST SPRINT



FIRST SPRINT

Attribute-Driven Design - Bass

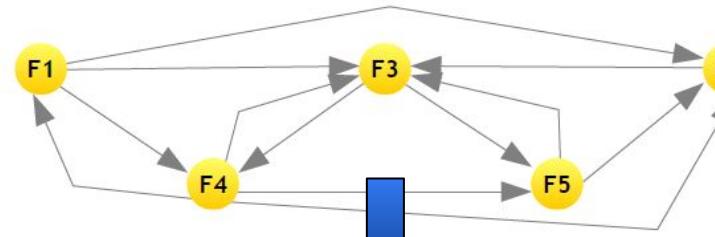
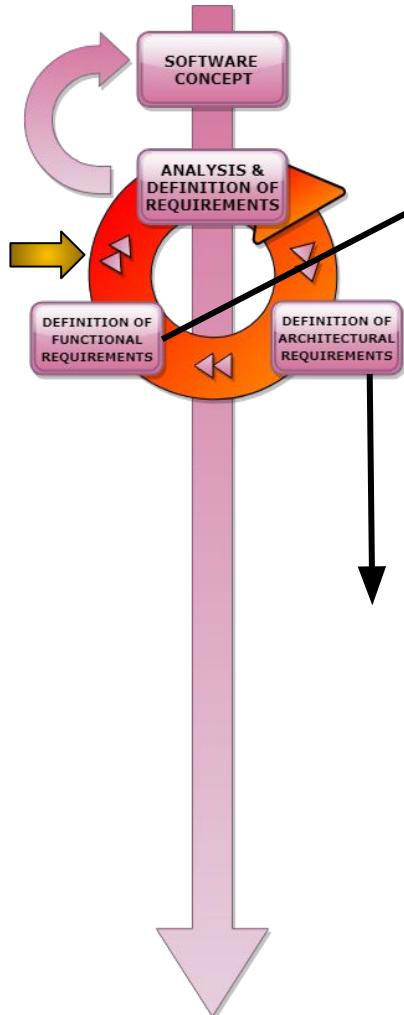


FIRST SPRINT



- Tamaño de los equipos.
- Retroalimentación.
- La participación de todos los miembros.
- Establecer unas fechas límite pero con la posibilidad de ser flexibles.
- Analizar los tipos de arquitecturas posibles y métodos de calidad de software.
- Precisar la documentación.

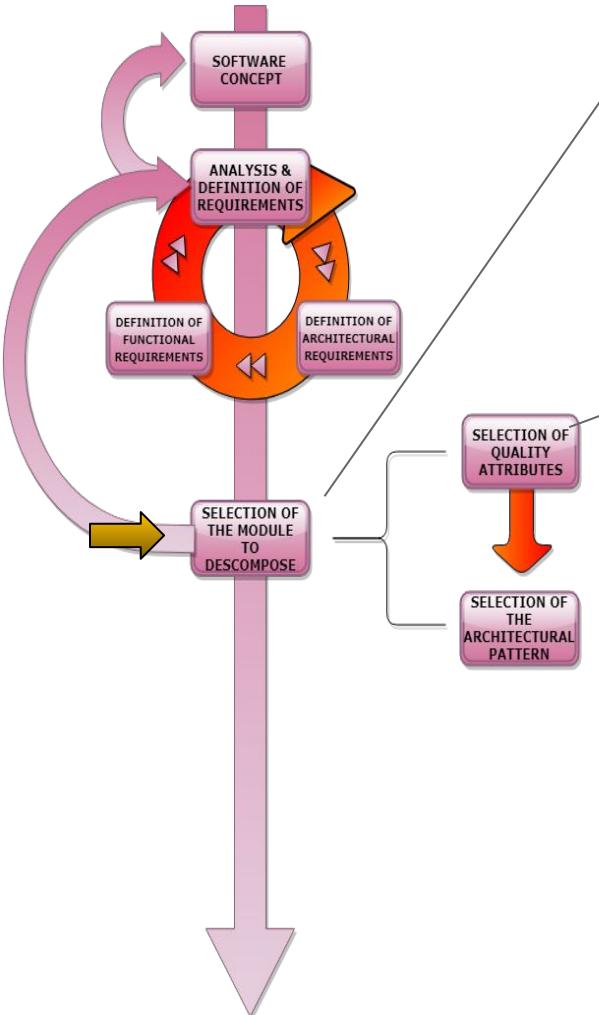
FIRST SPRINT



PROCESOS	DEPENDE DE	DEPENDE PARA	PRIORIDAD
F1	3	1	4
F2	1	3	3
F3	2		5
F4	4	2	2
F5	2	2	1

- Ayuda a la toma de decisiones a partir de los beneficios y limitaciones de cada alternativa.
- Crea nuevas oportunidades de diseño.
- Describe posibles cambios en la arquitectura seleccionada.

FIRST SPRINT



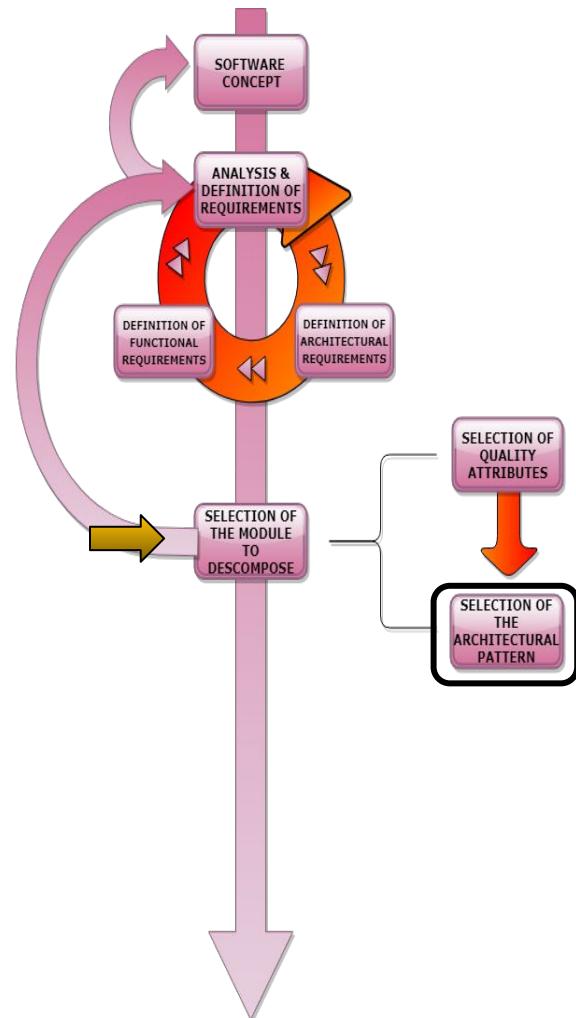
1. Organización
2. Dificultades y riesgos
3. Conocimiento actual
4. Negocio

- Elaborar una lista de atributos.

Disponibilidad	Interoperabilidad	Modificabilidad
Rendimiento	Seguridad	Comprobabilidad
Usabilidad	Extensibilidad	Acoplamiento
Encapsulamiento	Reusabilidad	Complejidad
Escalabilidad	Transparencia	Optimización
Correctitud	Adaptabilidad	Fallos
	Cohesión	

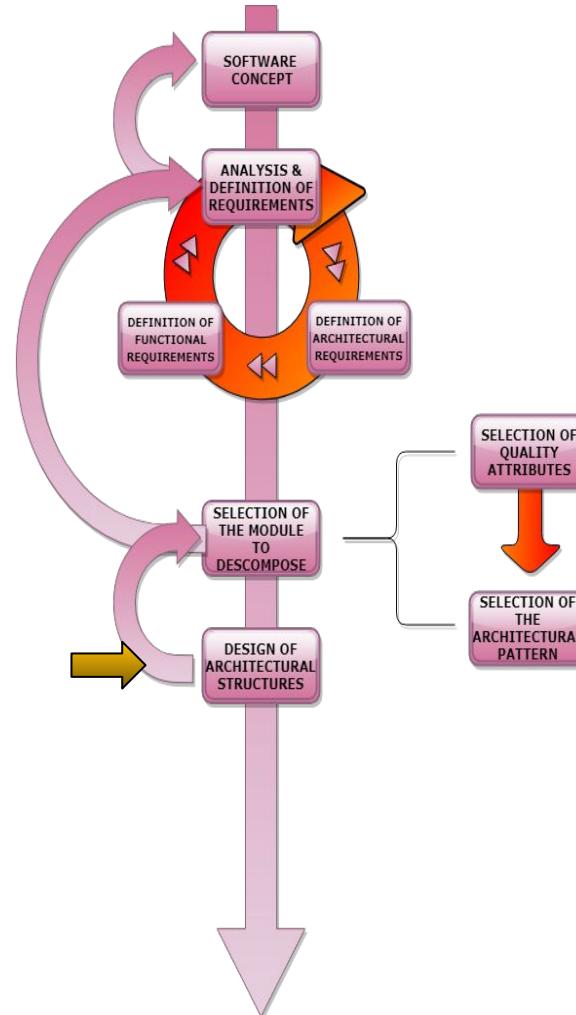
- Clasificar por prioridad (A, M, B).

FIRST SPRINT



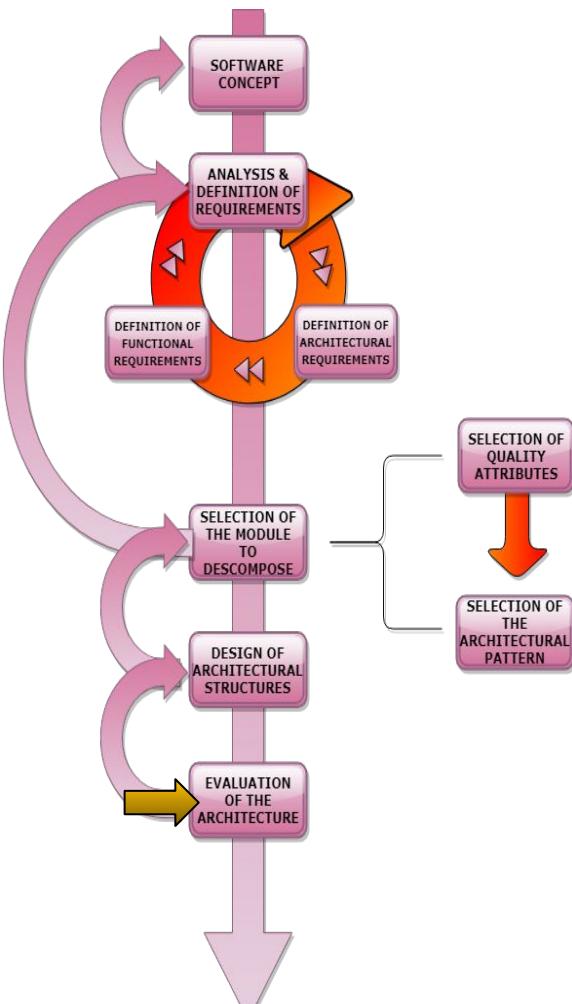
1. Elaborar una lista de directrices consensuada entre los miembros.
2. Seleccionar los patrones más apropiados para cada directriz.
3. Elaborar un nuevo patrón a partir de los patrones elegidos.
4. Crear instancias de los módulos y asignar la funcionalidad.
5. Definir las interfaces de los elementos instanciados.
6. Análisis.

FIRST SPRINT



1. Preparar los equipos necesarios para el diseño con las herramientas previstas.
2. Automatizar todos los procesos posibles.
3. Cumplir con los requisitos conseguidos del paso anterior en la implementación.

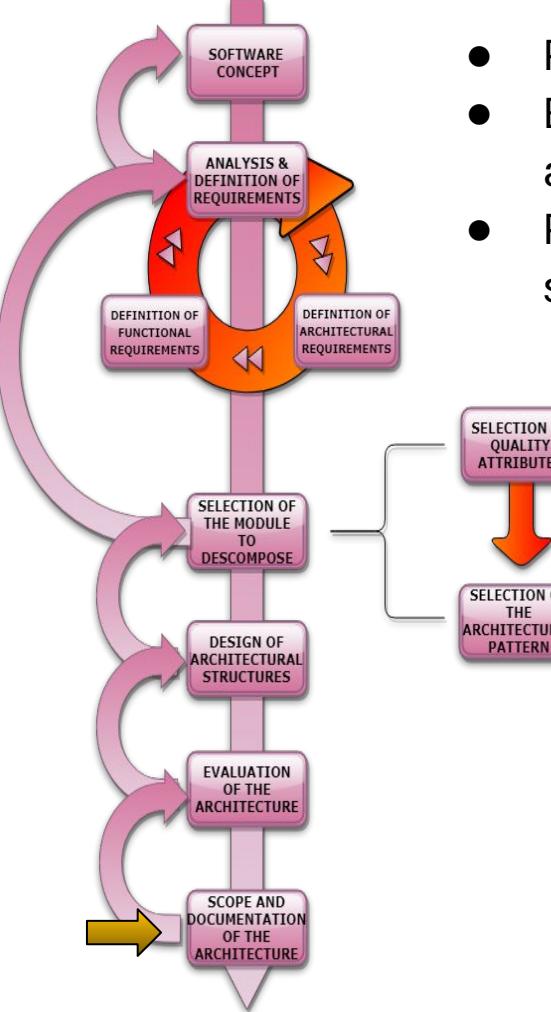
FIRST SPRINT



'Software architecture analysis method' (SAAM)

1. Desarrollar escenarios de calidad, compuestos por cualidades, usos y usuarios esperados.
2. Clasificación de los escenarios en directos e indirectos.
3. Evaluación de los escenarios que presenten problemas.
4. Comprobar la interacción de escenarios a modificar.
5. Evaluación del módulo central

FIRST SPRINT



- Partir de un documento base.
- Especificar las vistas arquitectónicas a producir.
- Priorizar la finalización de todas las secciones restantes.



- Requisitos funcionales y arquitectónicos.
- Resumen de las etapas en ADD.
- Detalles de la evaluación en SAAM.
- Actualizaciones posteriores.



FIRST SPRINT

REFINAMIENTO

- Refinamiento en first sprint y metodología ágil.
- Arquitecto encargado de organizarla.

CONSIDERACIONES

- Implicación del arquitecto en el desarrollo.
- El rol de arquitecto llevará a un consenso las decisiones tomadas en cada entrega de un sprint y la arquitectura
- Refinamiento continuo del product backlog.



FIN