Contents

Desafios **1**
- Escala . . . 1
- Productividad . . . 2
- Calidad . . . 2
- Consistencia y Repetitividad . . . 2
- Cambios . . . 3

Caracteristicas SRS **3**
- Correctitud . . . 3
- Completitud . . . 3
- No ambiguedad . . . 3
- Verificabilidad . . . 3
- Consistencia . . . 3
- Rastreabilidad . . . 3
- Modificabilidad . . . 4
- Ordenada en terminos de importancia y estabilidad . . . 4

Validacion SRS **4**

Estilo Arquitectonico **4**

Vista de C&C **4**
- Que es? . . . 4
- Tipos . . . 5
  - Tubos y filtros . . . 5
  - Repositorio de datos . . . 5
  - Cliente servidor . . . 5
  - Otros estilos . . . 6

Cohesion funciones **6**

Metodo ATAM **6**

Desafios

Escala

(IS debe considerar escala) Los metodos utilizados para resolver problemas pequeños no siempre escalan a problemas mas grandes. Por ejemplo, el caso de contar personas. No es lo mismo contar la cantidad de alumnos en un aula que realizar un censo nacional.

(metodos IS deben ser escalables) Cuando hablamos de escala en la IS tenemos dos dimensiones a considerar: los metodos de ingenieria y la administracion del proyecto. Para sistemas pequeños estas pueden ser informales, sin

embargo, para sistemas mas grandes es indispensable que las formalicemos.

Productividad

(cronograma) Unos de los objetivos principales de la IS es reducir el costo y el "time to market", el cual es muy importante en el contexto de negocios. Como analogia podemos tomar el caso de una maquina de cafe. Si una vez que iniciamos la maquina nos da el cafe 2hs despues, no nos va a servir, por mas que sea el mejor cafe del mundo. Algo similar pasa en lo relacionado al cronograma de entrega de un sistema de software.

(se mide en KLOC/PM, P/M) (enfoque IS general alta productividad reducir costo y tiempo) Ahora, como entra en juego la productividad en esto? Uno de los mayores costos en el desarrollo es la mano de obra, por ende, si aumentamos la productividad reducimos el costo al mismo tiempo que reducimos el tiempo requerido, bajando de estea manera el costo total del proyecto y el time to market.

En resumen, la IS busca lograr alta productividad.

Calidad

(desarrollar software calidad objetivo funamental) (enfoque producir software de calidad) La calidad a diferencia del costo y el tiempo es algo complicado de medir. **(definir, no medir)** Para juzgar la calidad generalmente se usa el estandar ISO, que define la calidad en base a 6 atributos: funcionalidad, confiabilidad, usabilidad, eficiencia, mantenibilidad y portabilidad.

De igual forma, el concepto de calidad varia dependiendo el proyecto, y debe definirse de antemano antes de la realizacion del mismo.

Puede que haya proyectos que valoren mucho mas la confiabilidad o la usabilidad sobre otros atributos.

El objetivo del proceso de desarrollo es lograr cumplir con este concepto de calidad.

Finalmente, el objetivo de la IS es lograr alta calidad. **(resumen calidad densidad de defectos, que se busca)**

Consistencia y Repetitividad

(desafio clave, asegurar exito repetible mantener consistencia calidad y productividad) El objetivo de la IS es lograr la sucesiva ejecucion **(produccion)** de proyectos de alta calidad y con alta productividad.

Es importante mantener cierto grado de consistencia, ya que esta es indispensable para estimar costos y determinar si un proyecto puede ser realizado o no. **(predecir resultado proyecto certeza razonable)**

El enfoque de la IS se centra en el proceso. En encontrar un proceso repetible que permita lograr el objetivo (alta calidad y productividad)

Cambios

Las empresas e instituciones sufren cambios constantemente, por lo cual, el software debe poder adaptarse a estos cambios.

(objetivo preparar software facil modificacion)

Un metodo que produzca alta calidad y productividad pero que no se adapte a cambios es poco util.

Caracteristicas SRS

Correctitud

Cada requerimiento plasmado en la SRS describe **(precisamente)** una característica deseada por el cliente **(en el sistema final)**

Completitud

Todas las caracteristicas deseadas por el cliente se encuentran descriptas **(es dificl de lograr)**

No ambiguedad

Para cada requerimiento, existe un único significado.

La no ambiguedad influye mucho en la verificabilidad

Verificabilidad

Para cada requerimiento existe un proceso efectivo para determinar si el sistema final cumple con dicho requerimiento

Consistencia

Los requerimientos no deben contradecirse entre si

Rastreabilidad

Se debe poder rastrear el origen de cada requerimiento y **(como se relaciona)** sobre que elementos de software este tiene impacto.

Puede ser: - Hacia adelante: en base a un requerimiento se debe poder determinar sobre que elementos de codigo o diseño tiene impacto - Hacia atras: Partiendo

de un elemento de codigo o diseño se debe poder determinar que requerimientos atiende

Modificabilidad

La estructura y estilos de la SRS debe permitir **(incorporar cambios facilmente)** que se puedan hacer modificaciones de manera sencilla preservando la completitud y la consistencia **(estorbo redundancia)**

Ordenada en terminos de importancia y estabilidad

Los requerimientos pueden tener distintos grados de importancia y pueden tener mas o menos chances de cambiar en un futuro. **(algunos reqs son esenciales y dificilmente cambien) (otros reqs son propensos a cambiar)** Por ende, para minimizar los riesgos de un posible cambio de requerimientos se debe priorizar el orden de construccion de los requerimientos teniendo en cuenta que requerimientos son mas importantes y tienen menos chances de cambiar en un futuro

Validacion SRS

Es muy comun que se produzcan malentendidos al plantear los requerimientos. Estos errores en los requerimientos son muy costosos de arreglar en etapas mas avanzadas del proceso de desarrollo, por lo cual, nos interesa corregir la mayor cantidad posible de errores en **(esta etapa)** la etapa de analisis y especificacion. Para ello se realizan reuniones entre el autor de la SRS, el cliente, representantes de los usuarios y programadores y se revisa en conjunto la SRS en busca de errores. **(listas de control)**

Este proceso suele ser muy efectivo y termina encontrando entre un 40% y 80% de los errores totales en los requerimientos.

Estilo Arquitectonico

Un estilo arquitectonico es la estructura y restricciones relacionadas observadas en muchos sistemas, que representa una estructura general util para la arquitectura de cierto tipo de problemas

Vista de C&C

Que es?

(componentes almacenan datos o son elementos computacionales) (nombre componenten representa rol) (distintos tipos tienen distintos simbolos) Es un tipo de vista arquitectonica que se centra en representar al

sistema como un grafo en donde los componentes del sistema son nodos y los conectores son aristas que conectan estos nodos. **(conectores mecanismo interaccion componentes) (pueden ser provistos entorno o mecanismos complejos) (tienen nombre y tipo)** Es muy util para determinar como interactuan las diferentes partes del sistema en tiempo de ejecucion. **(describe estructura ejecucion sistema) (que componentes existen como interactuan tiempo ejecucion)**

Tipos

Tubos y filtros

Es muy util para representar sistemas que esencialmente son una serie de transformaciones sobre datos.

Sus componentes son los filtros, los cuales realizan computos sobre los datos. Y sus conectores son los tubos, los cuales conectan unidereccionalmente a los filtros

Los filtros tienen las siguientes restricciones: - No conocen la identidad del emisor ni el consumidor de los datos que procesan/producen. - Son asincronos - Deben ralizar buffering y sincronizacion para poder **(asegurar)** permitir un correcto funcionamiento del sistema **(independiente)**

Los tubos tienen las siguientes restricciones: - Son unidireccionales - Conectan el puerto de salida de un filtro con el puerto de entrada de otro filtro

Repositorio de datos

Los componentes en la vista de repositorio de datos son los repositorios, los cuales proveen un almacenamiento confiable y persistente **(permanente)** de datos. Y los usuarios de datos, los cuales realizan computos sobre los datos. **(acceden datos) (cargan datos) (se comunican con otros usuarios por repo)**

Los conectores de esta vista son conectores de lectura/escritura

Existen dos variantes: - Pizarra: en esta variante siempre que se actualizan los datos se avisa a todos los usuarios pertinentes. Justamente como su nombre lo dice es como si se tratase de una pizarra en un aula. Si alguien escribe o borra algo en la pizarra, todos los que estan en el aula pueden verlo. - Repositorio: en esta variante el repositorio es pasivo y no avisa a los usuarios de cambios.

Cliente servidor

En esta vista tenemos dos tipos de componentes: - Cliente: - Es el que inicia la conversacion - No se comunica con otros clientes - Pide servicios al servidor - Servidor: - Provee servicios al cliente **(responde solicitudes)** - No puede iniciar la conversacion

Los conectores de esta vista son las request/response **(solicitud/respuesta)**, las cuales son asimetricas. La comunicacion en esta vista es asincrona

Cuenta con una estructura multinivel que consta de los siguientes niveles: - Nivel de usuario: aqui se encuentran los usuarios - Nivel de servidor: aqui se encuentran los servidores **(contiene reglas de servicio)** - Nivel de BBDD: aqui se encuentra la informacion

Otros estilos

P2P, publicar-subscribir, procesos que se comunican

Cohesion funciones

La cohesion es un concepto intra-modular que describe la fortaleza del vinculo entre los componentes de un mismo modulo. Nuestro objetivo es incrementar la cohesion, ya que esto provoca que se reduzca el acomplamiento, simplificando el diseño y facilitando el mantenimiento.

Tenemos los siguientes tipos de cohesion: - Casual: **(sin significado)** el programa fue partido arbitrariamente, quizas para lograr un posible reuso de codigo. - Logica: los componentes del modulo perteneces a una misma clase logica - Temporal: es como la clase logica solo que los componentes del modulo se ejecutan al mismo tiempo **(se ejecutan juntos) (relacionados en el tiempo)** - Procedural: los componentes se ejecutan dentro de una misma unidad procedural - Comunicacional: los componentes de un modulo trabajan sobre el mismo tipo de dato - Secuencial: La salida de un componente es la entrada de otro. Tiene el beneficio de que es un nivel bastante bueno de cohesion, es fácil de mantener, sin embargo es complicado de reutilizar. - Funcional: todos los elementos del modulo contribuyen a lograr una unica funcion

Metodo ATAM

El metodo ATAM (Architecture Tradeoff Analysis Method) busca analizar las propiedades y conseciones entre arquitecturas

1. Recolectar escenarios: en este paso se recolectan los escenarios que se desean evaluar para cada arquitectura **(escenarios describen interaccion sistema) (se eligen escenarios de interes para analisis)**
2. Recolectar especificaciones y/o restricciones: en este paso se plantea el comportamiento esperado del sistema y ciertas restricciones que los atributos de calidad deben cumplir **(especificar niveles desaados atributos de interes)**
3. Describir vistas arquitectonicas: en este paso se recolectan las diferentes vistas arquitectonicas **(distintas vistas pueden ser necesarias distintos analisis)**

4. Analisis especificos a cada atributo: se analiza el impacto de cada arquitectura sobre cada atributo de calidad y se comparan entre si **(para cada atributo se analizan las vistas en distintos escenarios separadamente) (se compara con niveles requeridos) (forma base eleccion entre arquitecturas)**
5. Identificar puntos sensitivos y de compromiso: se determinan que factores afectan en gran medida a algun atributo de calidad, estos son los puntos sensitivos. Los factores que son puntos sensitivos para varios atributos se denominan puntos de compromiso **(se realiza analisis sensiblidad y compromiso) (impacto elemento atributo calidad) (elementos mayor impacto punto sensiblidad) (analisis compromiso elementos punto sensibilidad varios atributos)**