UNIVERSIDAD TECNICA
FEDERICO SANTA MARIA

Departamento de Informática
Universidad Técnica Federico Santa María

Ingeniería de Software

Ingeniería de Requisitos

Hernán Astudillo & Gastón Márquez
Departamento de Informática
Universidad Técnica Federico Santa María

Contexto

Contexto [1]

- ADSW
  - Interacción con el cliente
  - El cliente dijo sus necesidades (Sabía que tenía un problema, pero no sabía lo que quería, ¿no?)
  - Luego, listado de *posibles* requisitos. ¿Eran todos factibles?
  - Los Requisitos se obtuvieron por la visita del cliente (la mayoría)
  - ¿Bastaba con solamente “anotar” lo que el cliente decía?
  - En general, hubo muchas cosas que no se pudo hacer por tiempo, conocimiento, otros.
  - El cliente creyó haber dicho todo, pero bastó un par de preguntas para saber que recién había dicho solo el 20% de sus necesidades

Contexto [2]

- ADSW
  - Entonces, ¿Qué podemos decir de nuestra experiencia?
    - Tuvimos problemas para ***entender*** los Requisitos que obtuvimos desde el cliente
    - Organizamos los Requisitos de forma ***desorganizada***
    - Gastamos muy poco ***tiempo*** en verificar lo que ibamos a realizar realmente
    - Nunca consideramos el ***control*** de los cambios de Requisitos
    - Quizás, fallamos en ***establecer*** las bases del sistema antes de pensar

Contexto [3]

- ADSW
  - Recopilando la experiencia de los alumnos de Santiago y Valparaíso, se puede decir que
    - "*Profe, nosotros queríamos construir software lo más rápido posible y es por eso que empezamos en el momento a desarrollarlo*" (sin haber entendido bien las necesidades del cliente)
    - "*¿Sabe profesor?, nosotros al momento de ir desarrollando el proyecto íbamos aprendiendo sobre algunos requisitos que no habíamos entendido bien*"
    - *"Profe, claramente después de haber presentado el proyecto a la cliente, nosotros pudimos entender lo que ella realmente quería"*
    - *"Profesor, como grupo tuvimos un problema. Cada vez que le enviábamos un correo a la cliente, algunos requisitos iban cambiando rápidamente, por lo que llegamos a considerar que documentar los requisitos era una pérdida de tiempo"*

Contexto [4]

- Los argumentos descritos por los alumnos contienen mucha verdad, especialmente para pequeños proyectos que tardan un par de meses en ser desarrollados.
- Pero….
  - ¿Qué ocurre con un proyecto de gran tamaño y complejo?
    - Claramente con estos argumentos, el proyecto falla.
    - Lleva al fracaso.

¿Alguna solución? [1]

- Solución → Ingeniería de Requisitos (IR)
- IR comienza durante la comunicación de las actividades del proyecto hasta el modelamiento
- IR construye un puente entre los Requisitos del sistema, diseño y construcción del software

¿Alguna solución? [2]

- ¿Qué permite examinar IR?
  - El contexto del software que será realizado
  - Las necesidades específicas de diseño y construcción
  - Las prioridades que guían el orden en que cada trabajo debe ser realizado
  - La información, función y comportamiento que tendrá impacto en el resultado final

¿Revolucionario u obvio?

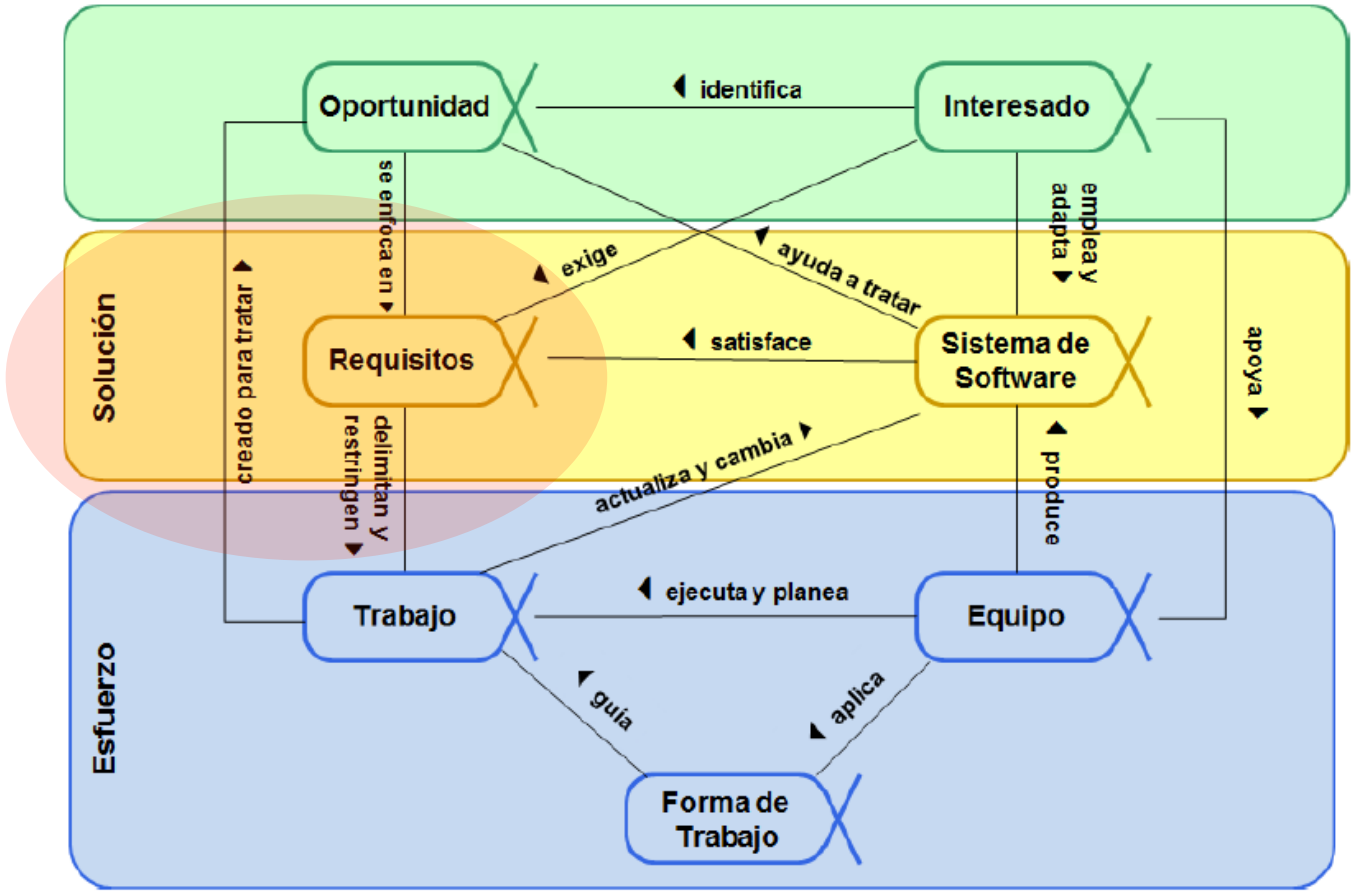
- 30-50% de funcionalidad es innecesaria (overhead)
  - [C. Ebert, R. Dumke - 2007]
- Estudio de 8000 proyectos en EEUU (2000)
  - En plazo y costo: 9%-16%
  - Retrasados/excedidos: 53%-60%
  - Cancelados (abortados): 31%
- ¿Porqué fueron cancelados?
  - Requisitos incompletos: 13%
  - Falta de participación del usuario: 12%
  - Falta de recursos: 11%
  - Expectativas descontroladas: 10%
  - Falta de apoyo gerencial: 9%
  - Requisitos inestables: 9%
  - Falta de planificación: 8%
  - Obsolescencia pre-parto: 7%

Más argumentos

- Cuantificación de Impacto
  - Costo de reparar errores en los requisitos…
    - Detectado durante análisis: $X
    - Detectado durante diseño: $5X
    - Detectado durante construcción: $10X
    - Detectado durante prueba: $20X
    - Detectado después de entregado: $100X-$200X

Requisitos

Modelo SEMAT



Modelo SEMAT

- Los *Interesados (stakeholders)* exigen requisitos
- La *Oportunidad* se enfoca en requisitos
- El *Sistema de Software* satisface los requisitos
- Los requisitos delimitan y restringen el *Trabajo* (proyecto)

Requisitos

- Requisitos: descripciones de lo que el sistema debe/debería hacer
  - Capacidades y condiciones a las que el sistema debe adherir
- Clasificación típica:
  - Requisitos funcionales: servicios/funciones a proveer (ej: sistema debe calcular promedio de ingreso mensual)
  - Requisitos extra-funcionales (NFRs, non-functional requirements): restricciones sobre servicios/funciones del sistema (ej: uptime de 99%)
- FURPS+: otro modelo de clasificación
  - Functional, Usability, Reliability, Performance, Supportability, (+: Implementation, Interface, Operations, Packaging, Legal)

Requisitos y riesgo

- ¿Para qué estudiamos requisitos?
  - Para reducir el riesgo de construcción
    - (R1) Efectividad: riesgo de hacer algo inútil
    - (R2) Eficiencia: riesgo de sub-/sobre-estimar recursos
- Clasifiquemos las causas de cancelación
  - Requisitos incompletos: 13% (R1)
  - Falta de participación del usuario: 12% (R1)
  - Falta de recursos: 11% (R2)
  - Expectativas descontroladas: 10% (R1)
  - Falta de apoyo gerencial: 9%
  - Requisitos inestables: 9% (R1)
  - Falta de planificación: 8% (R2)
  - Obsolescencia pre-parto: 7%

Requisitos funcionales [1]

- Funcionalidad, características, servicios que el sistema:
  - Debe proveer (si son obligatorios)
  - Debería proveer (si son deseables)
- Analogía:
  - Requisitos funcionales: los “verbos”

Requisitos funcionales [2]

- Ejemplo: Proyecto Programa Institucional de Inglés Comunicacional (http://eq42.fdo-may.ubiobio.cl/sgic/)
- Algunos Requisitos funcionales
  - *El sistema debe permitir la inscripción de una asignatura de inglés comunicacional, mostrando él o los posibles cursos a inscribir.*
  - *El sistema debe permitir la autenticación del usuario a través del RUT y la contraseña institucional.*
  - *El sistema debe reconocer la sede respectiva del estudiante, Chillán o Concepción. Para así mostrar las asignaturas y secciones que corresponden.*

Requisitos extra-funcionales (NFRs) [1]

- Condiciones y restricciones sobre el sistema
- Potenciales orígenes:
  - Producto (ej: *uptime de 99%; respuestas en menos de 5[s]*)
  - Organización (ej: *desarrollo sólo con .NET*)
  - Externos (ej: *cumplimiento de normas bancarias, cumplimiento DO-178C en software aeronáutico*)
- No minimizar impacto: NFR no satisfechos pueden dejar un sistema sin uso
  - *Ej: sistema en avión no cumple con condiciones de estabilidad no es certificado y queda fuera del mercado (pero hacía lo que debía hacer)*
- Analogía:
  - Requisitos extra-funcionales: los “adverbios”

Requisitos extra-funcionales (NFRs) [2]

- Ejemplo: Proyecto Programa Institucional de Inglés Comunicacional (http://eq42.fdo-may.ubiobio.cl/sgic/)
- Algunos Requisitos extra-funcionales
  - *La mayoría de los usuarios de la aplicación no son personas experimientadas. Por lo anterior, el sistema deberá ser fácil de entender y utilizar.* (Usabilidad)
  - *El representante de la empresa debe tener el 100% de confiaza en el sistema, por lo que en cada entrega del sistema se probará todo.* (Fiabilidad)
  - *El representante de la Institución dió a entender que es de suma importancia mantener el control de acceso al sistema.* (Seguridad)
  - *El sistema debe funcionar en un servidor HTTP Apache 2.2.3, usando MySQL versión 5.0, Sublime Text 2.0 y Netbeans 7.4.*

Ejercicio

- Formar los grupos ya establecidos para el proyecto de ISW y definan sus requisitos funcionales y extra-funcionales del proyecto semestral.
- ¿Serán los mismos en todos los grupos?

Ejercicio

- Grupo 1
- Grupo 2
- Grupo 3
- Grupo 4
- Grupo 5
- Grupo 6
- Grupo 7

Ingeniería de Requisitos

Ingeniería de Requisitos [1]

- Defínción de IR [Zave, 1997]

  *"Requirements engineering is the branch of software engineering concerned with the **real world goals** for, functions of, and constraints on software systems. It is also concerned with the relationship of these factors to **precise specifications** of software behavior, and to **their evolution over time and across software families**."*

Ingeniería de Requisitos [2]

- Esta defincion es atractiva pues menciona los siguientes puntos:
  - “Real world goals”
    - Esto representa el ¿porqué? y ¿qué? de un sistema
  - “Precise specifications”
    - Lo anterior provee las bases de
      - **Analizar** los Requisitos
      - **Validar** lo que los stakeholders quieren
      - **Definir** lo que los diseñadores deben contruir
      - **Verificar** que lo que se está haciendo está correcto
  - “evolution over time and across software families”
    - Enfatiza la realidad del contante cambio de las tecnología

Ingeniería de Requisitos [3]

- Pero, ¿por qué Ingeniería?
  - Una definición típica de Ingeniería es *creación de soluciones costo-efectivas a problemas prácticos aplicando conocimiento científico* [Shaw, 1990]
- Interesante, ¿no?
  - El término Ingeniería nos recuerda que IR es una parte importante en la ingeniería de procesos: toma problema del mundo real (*real world goals*), y se idea una solución óptima (*precise specifications*) a ser desarrollada en una plataforma (*evolution over time and across software families*)

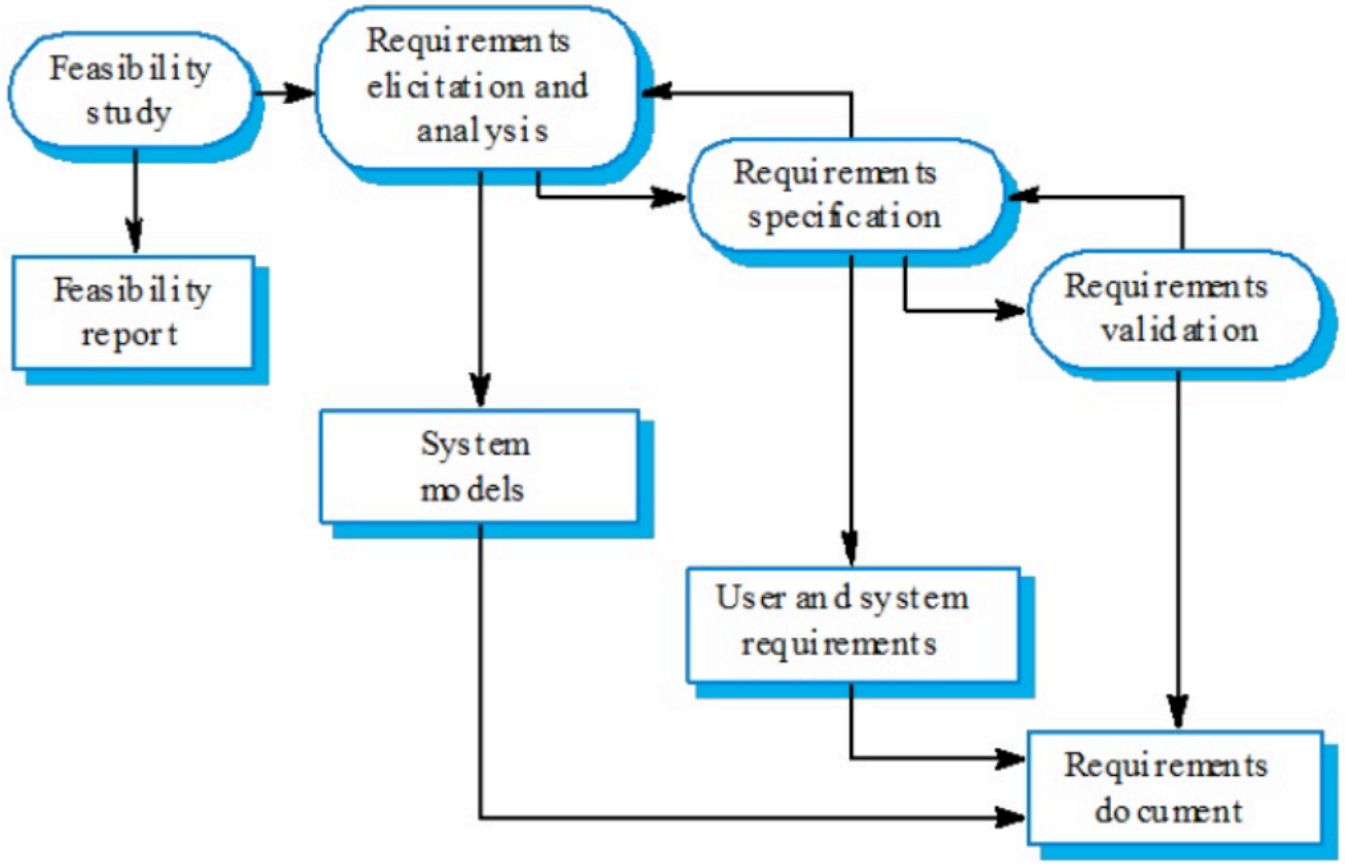
Ingeniería de Requisitos [6]

- Lo anterior nos hace pensar
  - Lo que hicimos en ADSW ¿Fue suficiente?
  - ¿Basta solamente escuchar al cliente y tomar nota?
- Por lo que se ve, se puede hacer muchas cosas más que sólo anotar frases del cliente.
- Entonces, entendamos qué son los Requisitos Funcionales y Extra-Funcionales

Proceso de Ingeniería de Requisitos [1]

- El proceso usado para IR varía dependiendo del dominio de la aplicación, las personas involucradas y los requisitos obtenidos.
- Hay varias actividades genéricas comunes a todos los procesos:
  - Elicitación de Requisitos
  - Análisis de Requisitos
  - Validación de Requisitos
  - Administración de Requisitos

Proceso de Ingeniería de Requisitos [2]



Proceso de Ingeniería de Requisitos [3]

- Estudio de Factibilidad
  - Se decide si el proyecto se realizará o no
    - ¿El sistema contribuye a los objetivos organizacionales?
    - ¿Se puede realizar con la tecnología y presupuesto actual?
    - ¿El sistema se puede integrar a otros?
- Elicitación y Análisis
  - ¿Saben los stakeholders lo que quieren?
  - ¿Los stakeholders expresaron correctamente sus necesidades?
  - ¿Existen conflictos de intereses entre los stakeholders?

Proceso de Ingeniería de Requisitos [4]

- Validación
  - ¿Se demuestra que los requisitos definen el sistema?
  - ¿Costos de los requisitos?
  - ¿Costo de arreglar problemas?

Perspectivas para Describir Requisitos

- Funcional (clásico)
  - Especificar en lenguaje natural lo que el sistema debe hacer
    - “El sistema debe permitir inscribir, modificar y desinscribir cursos”
- Operacional
  - Escenarios concretos de acción
    - Casos de uso
    - Otros tipos de escenarios (casos límite, casos de abuso, etc.)
- Objetivos
  - Especificar no qué hará el sistema, sino para qué
  - Permiten negociación entre stakeholders & priorizar escenarios

Elicitación de Requisitos

Elicitación de Requisitos

- La elicitación de Requisitos es quizás la actividad que es considerada en primer lugar en IR.
- Elicitación→Captura [Goguen et al., 1994]
- La información obtenida de la elicitación son interpretadas, analizadas, modeladas y validadas antes de que los ingenieros de Requisitos puedan sentirse satisfechos que está completa la toma de requisitos.

Requisitos a elicitar [1]

- Uno de los objetivos más importantes de la elicitación es encontrar ***qué problemas deben ser resueltos*** e identificar los ***límites del sistema.*** [Nuseibeh et al., 2000]
- Estos límites definen, a alto nivel, dónde está la entrega final del sistema que mejorará el actual entorno.
- Posteriormente, se deben identificar a los **stakeholders** [Sharp et al., 1999]
  - Clientes (Customers)→los que pagan el sistema
  - Desarrolladores →los que diseñan, construyen y mantienen el sistema
  - Usuarios →los que interactúan con el sistema para realizar su trabajo.

Requisitos a elicitar [2]

- Finalmente, los **objetivos** es lo que el sistema debe tener [Dardenne et al., 1993].
  - Elicitar los objetivos enfoca a los ingenieros de requisitos al dominio del problema y las necesidades de los stakeholders, encontrando posibles soluciones a estos problemas.
- Pero puede pasar que los usuarios encuentren difícil articular toda la información que se describió anteriormente.
  - Si tuviesemos que haber pasado la elicitación de los requisitos a nuestros usuarios en ADSW, ¿obtendríamos algo?
  - Lo más probable es que hubiera leído la primera página solamente

Requisitos a elicitar [3]

- Para no tener problemas con los usuarios, los ingenieros de requisitos pueden describir las ***tareas*** de los usuarios.
- Dichas tareas puede ser representadas por los *Casos de Uso* [Schneider et al., 1998] en un *escenario* determinado [Jarke et al., 1998]

Técnicas de elicitación [1]

- Técnica tradicional
  - Cuestionarios
  - Encuestas
  - Entrevistas
  - Análisis de la documentación existente
  - Análisis de modelos de procesos y estándares
- Técnicas grupales
  - *Brainstorming*
  - *Focus* group
  - RAD/JAD workshop [Maiden et al. 1996] (https://bobsleanlearning.wordpress.com/reference/what-is-rad-jad/)

Técnicas de elicitación [2]

- Prototipos
  - Ayudan al rápido feedback del cliente
  - Provoca discusión y mejoras
- Model-Driven
  - Métodos Goal-Based
    - KAOS [van Lamsweerde et al., 1998] (http://www.objectiver.com/fileadmin/download/documents/KaosTutorial.pdf)
    - I*[Chung et al., 2000] (http://www.cs.toronto.edu/km/istar/)
  - Métodos Scenario-Based
    - CREWS [Maiden et al., 1996] (http://sunsite.informatik.rwth-aachen.de/CREWS/)

Técnicas de elicitación [3]

- Cognitivas
  - Análsis de protocolo →los expertos piensan mientras desarrollan una tarea siendo observado por especialistas cognitivos
  - “Laddering” → uso de sondas para elicitar estructuras y contenidos del conocimientos del stakeholder
  - “Card sorting” → se le pide a los stakeholders ordenar cartas según dominio de entidades
  - “Repertory grids” → matriz de entidades a partir de preguntas a los stakeholders
- Contextuales
  - Nacieron como alternativas a las cognitivas [Goguen, 1993]. Agregan técnicas etnográficas, análisis de conversaciones y etnometodología para obtener patrones de conversaciones e interacción

Modelamiento, Análisis y Comunicación

Modelamiento y Análisis de Requisitos [1]

- Ejemplo
  - *En ADSW se tuvo que hacer modelos (Diagrama de Casos de Uso, Diagramas de Clases, otros); ¿qué se buscaba?*
- Modelamiento
  - Descripciones abstractas para ser interpretadas
  - Los modelos pueden ser utilizados para presentar un gran rago de IR
  - Los modelos, ¿Para qué son buenos?
- En las siguientes slides, se verá algunas utilidades de los modelos bajo distintos contextos [Nuseibeh et al., 2000]

Modelamiento y Análisis de Requisitos [2]

- Modelado Empresarial
  - La mayoría de las tareas de IR se relacionan con la organización que necesita el software.
  - El modelado empresarial es útil para capturar el propósito del software en una organización [Loucopoulos et al., 1995].
  - Los modelos de objetivos son muy útiles en IR.
- Modelado de Datos
  - Los proyectos de software generan grandes volúmenes de información.
  - El modelamiento de datos ayuda a encontrar oportunidades de decisiones en las organizaciones
  - Entity-Relationship-Attribute (ERA) [http://www.tac.mta.ca/tac/volumes/10/3/10-03.pdf]

Modelamiento y Análisis de Requisitos [3]

- Modelamiento conductual
  - Modelar Requisitos involucra modelar el comportamiento de los stakeholders.
  - ¿Qué se obtiene con esto?
    - Saber cómo se maneja el trabajo
    - Determinar funcionalidades clave
    - Finalmente, determinar un modelo operativo del sistema
- Modelo del dominio
  - Una parte importante de IR es sobre el desarrollo de las descripciones del dominio [Jackson et al., 1993]
  - El modelo del dominio provee una descripción abstracta del mundo en donde el software operará.

Modelamiento y Análisis de Requisitos [4]

- Modelado de requisitos extra-funcionales
  - Estos requisitos son generalmente los más difíciles de expresar como medición.
  - Los modelos que representan los NFRs se hacen para poder medir y testear el software.
  - Hay mucho conocimiento sobre algunos NFRs clave: seguridad, confiabilidad, escalabilidad, usabilidad…
- Ejemplo
  - *¿Cómo se podría haber modelado la confiabilidad del proyecto en ADSW?*
  - *¿Y en su proyecto de ISW?*

Modelamiento y Análisis de Requisitos [5]

- Análisis de Modelos de Requisitos
  - Un beneficio de modelar requisitos, es que se pueden analizar.
- Algunas técnicas de análisis de requisitos
  - Animación de requisitos [Gravell et al., 1996]
  - Razonamiento automático (case-based reasoning) [Maiden et al., 1992]
  - Knowledge-based critiquing [Fickas et al., 1988]
  - Consistency checking [Holzmann, 1997]
  - Validation & Verification (V&V)

Comunicación de los Requisitos [1]

- IR no es solamente un proceso de descubrimiento y especificaciones, sino también debe proveer la fácil comunicación de estos.
- Es importante saber ***cómo*** se documentará los requisitos.
- El foco de la documentación radica en la especificación de lenguajes y anotaciones, donde existen variadas propuestas:
  - Lenguajes formales y semi-informales [Wieringa, 1996]
  - Lenguaje lógico [Antoniou, 1998]
  - Lenguaje natural [Ambriola et al., 1997]
- Pero, ¿cuál es la idea de los puntos anteriores?
  - Los *stakeholders* deben poder entender & validar fácilmente las necesidades

Comunicación de los Requisitos [2]

- Volere [http://www.volere.co.uk/index.htm] es una de las formas de especificar y documentar Requisitos más usadas en la industria.
- Ranking de documentación de requisitos [Makar, LiquidPlanner, 2015] más utilizados (industria, academica, investigación, otros)
  - Context Diagram
  - Functional Decomposition
  - Use Case Diagram
  - Sequence Diagram
  - AS-IS and TO-BE process model
  - User Stories
  - Mind maps

¿Qué hacen los analistas de requisitos?

¿Qué hacen los analistas de requisitos? [1]

- El analista de requisitos debe hacerse 7 preguntas: ¿Quién?, ¿Qué?, ¿Dónde?, ¿Cuándo?, ¿Porqué?, ¿Cuál? y ¿Cómo?
  - ¿**Cuál** es el problema qué necesita ser resuelto?
    - Límites del problema
  - ¿**Dónde** está el problema?
    - Contexto organizacional
    - Dominio del problema
  - ¿**Cuándo** necesita ser resuelto?
    - Restricciones de desarrollo
- [Easterbrook, 2004]

¿Qué hacen los analistas de requisitos? [2]

- El analista de requisitos debe hacerse 7 preguntas: ¿Quién?, ¿Qué?, ¿Dónde?, ¿Cuándo?, ¿Porqué?, ¿Cuál? y ¿Cómo?
  - ¿**Qué** problema prevenimos si lo resolvemos?
    - Identificar Factibilidad & Riesgo
  - ¿**Quién** (Quiénes) es (son) el problema?
    - Identificar a los *stakeholders*
  - ¿**Porqué** necesita ser resuelto?
    - Identificar objetivos
  - ¿**Cómo** un software ayudará?
    - Identificar escenarios

¿Qué hacen los analistas de requisitos? [3]

- Ejercicio:
  - Trabajar en los grupos del proyecto de ISW y responder las 7 preguntas para el proyecto.
- Debatir:
  - ¿Tenemos todos respuestas parecidas? ¿O no? ¿Por qué?

Requisitos- Conclusiones

Evaluación de Especificaciones

- Nuestro propósito: determinar si ciertos requisitos propuestos son…
  - correctos (describen algo bien)
  - útiles (sirven para guiar construcción)
  - necesarios y suficientes
- Preguntas
  - ¿Son correctos?
  - ¿Son consistentes?
  - ¿Están completos?
  - ¿Son realistas?
  - ¿Son verificables?
  - ¿Describen una sola cosa (cada uno)?
  - ¿Son rastreables?

Requisitos y SQA

- Idea importante: trazabilidad de Requisitos
  - Reqs. alto nivel → Reqs. bajo nivel (detalles) → Modelos → Código → Ejecución
- Permite tener certeza de que lo que deseaba el usuario es lo que está en ejecución
- Herramientas actuales apoyan la trazabilidad
  - Requisitos se plasman en casos de pruebas que son automatizados y ejecutados contra el código de manera automática (o semiautomática)
- Métricas importantes:
  - Complejidad/Tamaño
  - Volatilidad (cambios/tiempo)
  - Tasa Reqs. Implementados (implementados/total)

UNIVERSIDAD TECNICA
FEDERICO SANTA MARIA

Departamento de Informática
Universidad Técnica Federico Santa María

Ingeniería de Software

Ingeniería de Requisitos

Hernán Astudillo & Gastón Márquez
Departamento de Informática
Universidad Técnica Federico Santa María