**INTRODUCCION A LA INGENIERÏA DE REQUERIMIENTOS**

En la clase de hoy verá una aproximación a la ingeniería de requerimientos, sus dificultades, stakeholders, el problema de la comunicación, entre otros conceptos.

Comenzaremos por ver algunas dificultades esenciales del software.

* Problemática de la representación / modelo / descripción.
* Tipos de modelos.
* Dificultades para comprender los requerimientos.
* Stakeholders
* El punto de vista de los diferentes stakeholders.
* El gap semántico.

**LAS DIFICULTADES ESENCIALES DEL SOFTWARE**

Hay que distinguir dos tipos de dificultades en el software:

Esenciales: inherentes a su naturaleza,

Accidentales: las ofrece (actualmente) su producción pero no son inherentes al software (son las del desarrollo)

La esencia de una entidad de software es que es una construcción de conceptos de interconexión: conjuntos de datos, relaciones entre ítem de datos, algoritmos e invocaciones de funciones.

Esta esencia es abstracta y no varía con las distintas representaciones.

Propiedades inherentes a la esencia del software:

* complejidad
* conformidad (conformity)
* modificabilidad
* Invisibilidad

**COMPLEJIDADES.**

Las construcciones de software probablemente son más complejas que cualquier otra construcción humana, incluyendo las computadoras.

* “Escalar” el software no consiste en repetir los mismos elementos en una escala mayor.
* De la complejidad esencial y de su incremento no lineal con el tamaño se derivan importantes consecuencias:
* problemas clásicos de la producción de software:
* la dificultad de comunicación
* de la complejidad de la estructura deriva la dificultad de extender los programas a nuevas funciones y los estados no visualizados
* problemas de administración de los proyectos:
* hace difícil la visión global, impidiendo la integración conceptual
* crea la tremenda necesidad de aprendizaje y comprensión que convierte a la rotación de personal en un desastre

**CONFORMIDAD.--> Dejar conforme al usuario sobre lo que nos da para construir.**

Parte importante de la complejidad del software es de origen arbitrario, derivada de las instituciones y sistemas que su interface debe conformar.

El software debe conformar porque:

* muchas veces es el último en llegar a la escena
* otras veces se lo percibe como el que tiene mayor capacidad de conformar interfaces

**MODIFICABILIDAD. (siempre se va a modificar, en todas las etapas, aunque el cliente así no lo quiera)**

El software está constantemente bajo la presión del cambio, en los productos manufacturados los cambios sustanciales se hacen en series posteriores.

Razones de cambios en el software:

* el software corporiza funciones, y estas son las que mayor presión tienen por el cambio. **Si hay un cambio en la empresa, en el software también. (por ejemplo un cambio legal en el pais)**
* el software puede cambiarse muy fácilmente, es muy maleable

Todo el software exitoso es objeto de cambios, porque:

* si funciona bien se le añaden funciones a las originales
* sobrevive a la máquina para la que se diseño originalmente

En resumen:

* el software está embebido en una matriz cultural de: aplicaciones, usuarios, leyes y máquinas
* estas cambian continuamente y sus cambios fuerzan cambios sobre el producto de software

**INVISIBILIDAD.**

La realidad del software no está sumergida en el espacio, luego no tiene representaciones geométricas **(la tierra tiene mapas, pero el software no tiene un estilo de guía visible, entonces tendremos que abstraernos para construirlo)** en el sentido que:

* la tierra tiene mapas
* los chips tienen diagramas
* las computadoras tienen diagramas de interconexión

El software tiene varios diagramas que pueden representar:

* flujo de control
* flujo de datos
* patrones de dependencia
* secuencia de tiempo
* relaciones nombre-espacio

Se progresó en restringir y simplificar las estructuras del software, pero permanece no visualizable por lo que:

* no permite que la mente use algunas de sus herramientas más poderosas
* se dificulta la comunicación entre varias mentes

**DIFICULTADES ESCENCIALES.**

Los siguientes conceptos:

* complejidad
* conformidad (conformity)
* modificabilidad
* invisibilidad

**Impactan** en :

* Cómo entendemos lo que el cliente pide que hagamos (y determinar bien que es lo que quiere)
* Qué tenemos qué hacer
* La volatilidad de lo que nos piden
* La imposibilidad de visualizar el producto. **🡪 el prototipo intentará solucionar esto.**

**------------------------ o --------------------------------**

Existen dificultades que son esenciales a la hora de desarrollar software. También existe la problemática asociada a La representación/modelo/descripción.

La dificultad para comprender los requerimientos. Y la comunicación y relación con los Stakeholders.

Existen una amplia variedad de conceptos de modelo.

Un modelo puede ser:

* Una copia pequeña y exacta de algo (del objeto que querramos).
* Un ejemplo: el aeromodelismo

Una copia no es un modelo: no tienen las ventajas de un modelo (ej reducción de costos) ni sus desventajas (ej: imprecisión respecto al original)

Una descripción o analogía utilizada para ayudar a visualizar algo que no puede ser observado directamente

* Captura la motivación de la “simulación” o “descripción”

Un patrón o figura de algo a construir

* Apunta al plan de “construcción”

Una proyección teórica de un sistema imaginario o posible

* Esto se vincula con el interés de SE de utilizar modelos con un propósito de construcción
* El “original” no tiene porque existir o ser imaginario
* Esta alternativa es la pertinente en nuestro campo

**¿PARA QUE, DE QUE, Y COMO HACER MODELOS?**

El modelo y la realidad son diferentes. Construimos modelos con motivos de:

* Representación (algo que se quiere pareer a lo que es)
* Construcción

Se construyen modelos de objetos que existe o no (abstractos),

A partir de la realidad construimos un modelo que la representa con el fin de entenderla [representación, existe]

A partir de ese modelo construimos otro modelo que representa el artefacto que queremos construir [construcción, no existe.

**SUSTANCIA DEL PROCESO DE DESARROLLO.**

Blum describe la esencia del proceso software en tres pasos:

formular el modelo conceptual, que describe una solución al problema, el MC carece del rigor y precisión del MF

* su traducción al modelo formal que prescribe la solución implementada, en el MF se pueden verificar las afirmaciones que contenga
* la construcción de una implementación que satisface el modelo formal

No hace falta el grafico

Nótese que hablamos de desarrollo, no hemos considerado el proceso de obtención del conocimiento necesario para esa desarrollo

**EL DESARROLLO COMO SECUENCIA DE TRANSFORMACIONES**

“[...] el proceso software siempre involucra transformaciones desde modelos mentales informales (que describen qué queremos hacer) hasta una implementación formal (que detalla cómo operará la solución prevista)” (Blum)

Necesitamos la abstracción para tratar con la complejidad (Rumbaugh):

* Examen selectivo de ciertos aspectos
* Objetivo: aislar los aspectos importantes y suprimir los no importantes
* La abstracción es guiada por un propósito (pues este define lo que no es importante)
* Es posible múltiples abstracciones de la misma cosa, dependiendo del propósito para el que se hacen

**EL PUNTO DE VISTA DEL OBSERVADOR**



*(tenes que guardar un producto, y como ing en sist. Pensas en la base de datos, mientras que el usuario piensa en un carrito de compras)*

Le sugerimos ver el siguiente video:

(video)

**LA DIFICULTAD PARA OBTENER REQUERIMIENTOS.**

Brooks siempre presente...

**Lo mas difícil de construir un sistema es saber bien lo que se quiere construir. A veces ni el usuario sabe bien que quiere construir.**

•"The hardest single part of building a software system is deciding precisely what to build.

•No other part of the conceptual work is as difficult as establishing the detailed technical requirements, including all the interfaces to people, to machines, and other software systems.

•No part of the work so cripples the resulting system if done wrong.

•No other part is more difficult to rectify later."

•[Brooks, No Silver Bullets]

•Lo más difícil del trabajo conceptual

•El impacto perjudicial del error

•La dificultad de la rectificar

El problema principal es adquirir el conocimiento de los usuarios y otras fuentes**.(el usuario va a dominar el problemas mucho mas que nosotros. Entonces tenemos que ponernos a su nivel a través de él)**

De allí se deriva: ¿Dónde encontrar el conocimiento?

Problemas en la captura del conocimiento:

* formas no utilizables del conocimiento
* dificultades cuando se trata de un experto humano (no quiere decir que tenga la solución, esta la vamos a tener nosotros)

Los requerimientos:

* sólo tienen sentido en el contexto organizacional
* son producto de la interacción usuario-técnico **(pasarlo de lenguaje natural a técnico)**

Usuarios:

* falta de consenso entre usuarios **(los usuarios no se ponen de acuerdo que es lo que quieren)**
* tiempo limitado del usuario
* dificultades comunicacionales ¡!
* problemas del poder político (político a nivel empresa, hay usuarios que van a intentar ocultar información!)

Usuarios y desarrolladores

* problema de la comunicación **(a veces el conocimiento del usuario puede sobrepasar nuestro conocimiento o al revés )**
* diferencias de lenguaje **(nosotros tendremos lenguaje técnico, tendremos que saber traducírselo al usuario)**
* diferencias de formación profesional **(profesiones distintas entre usuario desarrollador)**

**Desarrolladores**

* **orientación a la solución (el desarrollador solo va a pensar en la solución, tendríamos que analizar bien cuál es el problemas que debemos atacar)**

Vea el siguiente video que lo ayudará a comprender estos conceptos.

**STAKEHOLDERS**

•Son los individuos u organismos que ganan o pierden con algún cambio, son impactados positiva o negativamente independientemente o no de su voluntad.

Loucopoulos (cap 3):

•“the parties who have interest in the software system under development”

(Wieringa&Glinz, 2007)

•“A stakeholder is a person or organization who influences a system’s requirements or who is impacted by that system”

Clasificación según su interés (Macaulay, L., 1996)

•Interesados en su construcción **(programadores, etc, todas las personas que vn a estar en la construcción del software)**

•Interés financiero **(el que pone la plata)**

•Interesados en su introducción y operación

•Interesados en su uso

**El universo de los stakeholders**

1.Interesados en su construcción

•Responsabilidad por el diseño y desarrollo

•Incluye: project managers, diseñadores de software, expertos en comunicaciones

2.Interés financiero

•Responsabilidad por la compra o la venta

•Incluye: analista de negocios, gerente de ventas, comprador

3.Interesados en su introducción y operación

•Responsabilidad por la implementación y el mantenimiento

•Incluye: equipo de entrenamiento y soporte del usuario, ingenieros de instalación y mantenimiento, gerentes usuarios

4.Interesados en su uso

•Interés en su uso

•Incluye: gerentes usuarios, toda clase de usuarios (directos e indirectos)

¿Cuáles son los objetivos de cada uno de estos grupos?

**¿COMO NOS VEMOS EL UNO AL OTRO?**

**COMO VEMOS AL** USUARIO**? (nosotros como rol de desarrollador)**

•Ellos...

•**No saben qué quiere**

**•No puede expresar qué quieren**

•Muchas “necesidades” motivadas políticamente

•**Quieren todo ahora (*y gratis (?* )**

•**No puede priorizar necesidades (como hay prioridades tiene que entenderlo, buscaremos entregar el MVP)**

•Quieren “primero yo” no “primero la empresa”

**•Rechazan responsabilidad por el sistema (el usuario siempre va a considerar que la responsabilidad es nuestra y no de él, cuando en realidad es de ambos porque nos puede dar algo erroneo)**

•**No están disponible para proveer una definición de un sistema que funcione**

**•No se comprometen con los proyectos de desarrollo**

**•No desean el compromiso**

**¿COMO NOS VE EL USUARIO?**

•Nosotros...

•No entienden “el negocio”

•Manejan la política de la empresa torpemente

•**Tratan de explicarles cómo hacer su trabajo**

•No pueden traducir una definición de un sistema en un sistema exitoso

•Dicen “no” todo el tiempo

•Ponen mucho énfasis en los tecnicismos **(háblame en castellano nos va a decir el usuario)**

•Siempre están por encima del presupuesto

•Siempre están demorados (nuestro mayor desafio son las fechas de entrga, hay que ver el motivo)

•Reclaman de los usuarios por tiempo y esfuerzo en detrimento de sus objetivos primarios

•Establecen estándares no realistas para la definición de requerimientos

•No están disponibles para responder a cambios legítimos de necesidades

•No puede mantenerse en el cronograma

**LA COMUNICACION**

El mensaje es completo: verbal y no verbal. Existen problemas en el emisor, receptor y en el medio.

**•Emisor: 🡪 Todo parte del emisor**

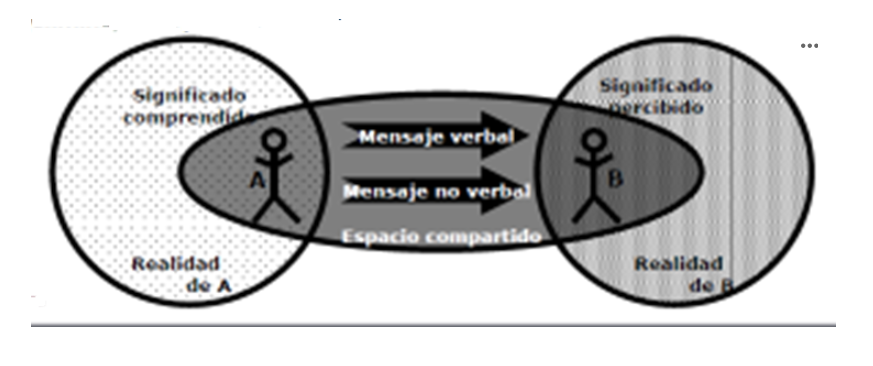
* no emite lo que debe (omisión)
* emite incorrectamente (error)

**•Receptor: 🡪 Y repercute en el receptor**

* no recibe lo que debe (incompletitud)
* recibe incorrectamente (error)

**•Medio**

* perturba (elimina, incorpora, deforma)



**LA IMPORTANCIA DE LA COMUNICACION**

Los stakeholders se comunican entre sí

La necesidad de comunicación entre los stakeholders es fuente de problemas de comunicación

El esfuerzo de desarrollo requiere comunicaciones con otras fuentes/receptores **(todo el conjunto se va a ver afectado)**

Se amplifica el impacto de los problemas comunicacionales.

**GAP SEMANTICO🡪 que tan aproximado estamos del entendimiento del sistema**

**Lo que nos va a dificultar el entendimiento:**

•Las aspiraciones, formación y experiencia de los usuarios y analistas difieren

•Los primeros manejan su terminología orientada al dominio en tanto que los analistas usan un vocabulario orientado a la computación

•Hay una sería dificultad en establecer un esquema de comprensión entre el analista y el usuario, esto es: **hacer que hablan el mismo lenguaje**

•Esto requiere reglas que ordenen las relación: el “contrato social” de los requerimientos

**El contrato social de los requerimientos Derechos del usuario**

1.Esperar del analista que hable su lenguaje **(hablar en el mismo lenguaje que el cliente)**

2**.Que el analista aprenda sobre su negocio y sus objetivos 🡪 Esto lo vamos a ir viendo en el tP**

3.Que el analista escriba una Especificación de Requerimientos de Software

4.Recibir explicaciones de los productos creados en el proceso de requerimientos. (producto = c0njunto de Req.)

5.Esperar de los desarrolladores un trato respetuoso

6.Obtener de los analistas y desarrolladores ideas y alternativas para los requerimientos y su implementación. **Hacerle sugerencia a los usaurios**

7.Describir características del producto que lo hagan simple y agradable de usar. **Usabilidad y amigabilidad**

8.Manejar opciones de ajustar sus requerimientos para permitir la reutilización de componentes de software(o modularizar) preexistentes

9.Recibir estimaciones de buena fe de costos de los cambios. Ser honestos en estimar

10.Recibir un sistema que satisfaga sus necesidades funcionales y de calidad 🡪 El cliente tiene que quedar conforme.

**El contrato social de los requerimientos Deberes del usuario**

1.Educar a los analistas y desarrolladores en su negocio. **(hacernos entender el mismo)**

2.Destinar tiempo a proveer y clarificar los negocios

3.Ser específico y preciso acerca de los requerimientos. **Evitar ambigüedades**

4.Tomar decisiones a tiempo

5.Respetar las evaluaciones de los desarrolladores de costo y factibilidad

6.Establecer prioridades a los requerimientos

7.Revisar los documentos de requerimientos y evaluar prototipos

8.**Comunicar los cambios** a los requerimientos. **(¿Por qué los quiere cambiar?)**

9.Seguir el proceso de cambios de la organización de desarrollo

10.Respetar el proceso de Ingeniería de Requerimientos que el analista utiliza. **Respetar nuestra labor.**

**¡¡¡Derechos y Deberes busca encontrar un equilibrio!!!**

RESUMIENDO....

**Temas clave**

•Las dificultades esenciales del software

•La complejidad, conformidad, cambiabilidad e invisibilidad impactan en nuestro entendimiento de lo que nos piden

•Problemática de la representación/modelo/descripción

•diferenciación modelo-realidad

•motivos para construir modelos: entender y describir lo que queremos construir

•los modelos se insertan en la visión del desarrollo como secuencia de transformaciones

•en el proceso de transformación necesitamos la abstracción

•Dificultades para entender los requerimientos

•en lo que se ve y en lo que se lee

•es la parte más difícil de la construcción de un sistema software

•en la obtención del conocimiento

•en los usuarios, desarrolladores y su relación

•Los stakeholders

•visión amplia de los que impactan o son impactados por un sistema

•heterogeneidad de los stakeholders

•diferencias usuarios-analistas, gap semántico

•el sesgo del punto de vista con el que "vemos" la realidad

•los problemas que introduce la comunicación

•el contrato social

Con este último concepto damos por terminada la clase 2.

Le sugerimos ver el siguiente video para ampliar sus conocimientos.

(en rojo los mas críticos o importantes)

- De modelos saber la definición