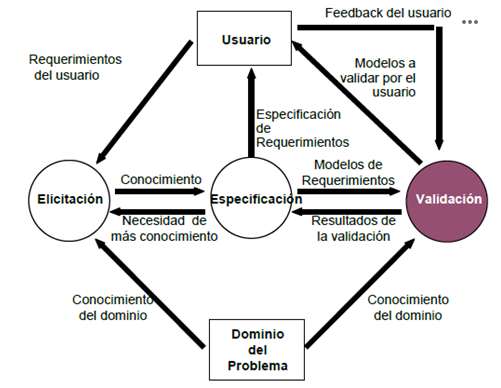
**VALIDACION DE LOS REQUERIMIENTOS**

Bienvenidos a la clase 12. En esa clase comprenderá la importancia del proceso de validación de los requisitos.



La figura anterior muestra un esquema del proceso de ingeniería de requerimientos para la etapa inicial, que está formado por 3 grandes fases. La elicitación nos va a generar el conocimiento necesario del dominio del problema, de los requerimientos de los usuarios y ese conocimiento se va a transformar en un conjunto de requerimientos que se conoce como especificación de los requerimientos y es el artefacto de software que está sujeto a la validación para acordar si ese modelo que construimos está en línea o es concordante con lo que pretende el usuario para resolver su problema.

**El problema y las causas**

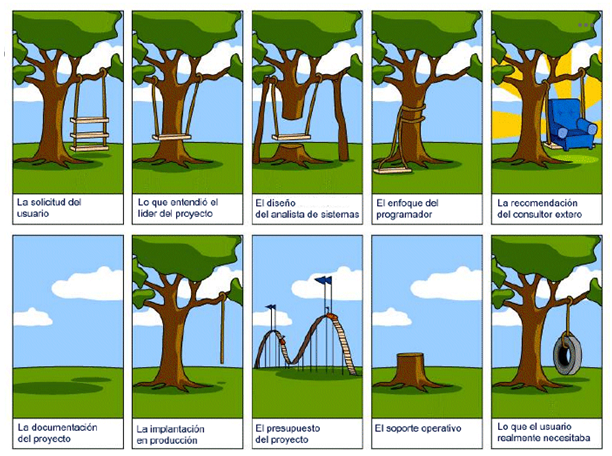
Pueden surgir por ejemplo que el modelo de requerimiento no tenga contradicciones o que no contenga definiciones auto contradictorias, Un modelo correcto de requerimientos no es necesariamente correcto. Tenemos que evitar hacer un esfuerzo en analizar el problema erróneo o evitar que tenga errores. Cuando hablamos de analizar un problema erróneo es porque hubo un problema en la elicitación y va a tener impacto en la validación.

Las causas tienen que ver con las dificultades en la elicitación de requerimientos del usuario porque , como ya vimos, en este proceso de elicitación de requerimientos hay dificultades y termina siendo el cuello de botella para adquirir conocimiento.

Otra de las dificultades es tener un esquema de comprensión común entre el analista y usuario, Cuando hablamos de un esquema de compresión común o tratar de ser compatibles con el usuario, en el sentido de que hablemos el mismo lenguaje que el usuario. Pero ya hemos manifestado que es difícil hablar el mismo lenguaje porque el usuario conoce su dominio y quizás nosotros no lo conocemos tanto.

El usuario habla con términos específicos del dominio que nosotros tenemos que aprender. Entonces tenemos que asegurarnos en etapas o en el proceso inicial de ingeniería de requerimientos, que realmente estamos analizando el problema que tiene el usuario para resolver y que estamos entendiendo, analizando ese problema y que el resultado de ese proceso inicial de ingeniería de requerimientos es una representación de que lo que el usuario demanda como solución a su problema.

Ese modelo de requerimientos sujeto a validación una vez validado, debería ser una representación de lo que pretende el usuario como solución a su problema. Obviamente la representación tiene que ver con un producto de software, por eso es una especificación de requerimientos o un modelo de requerimientos de software que una vez validado, va a constituir la solución para el usuario.



Esta figura es muy conocida, pero en vez de focalizarnos en cada una de las 10 imágenes que contiene esta figura lo importante de ver aquí, es lo que pide el usuario, que es la primer figura y qué es lo que realmente necesita, que está representado en la última figura. Y esto es claramente un problema de ingeniería de requerimientos; entender lo que solicita al usuario y entender el problema del usuario, porque el usuario puede tener un problema y muchas veces nos quiere dar la solución a su problema y realmente la solución no la tiene que dar el usuario, el usuario plantea el problema, no plantea la solución. Para eso estamos los ingenieros de requerimientos o de software.

Fíjense la diferencia entre lo que pide el usuario y lo que realmente necesita suponiendo en este contexto que el usuario necesita hamacarse utilizando el árbol bueno.

Muchas veces todo lo que pide el usuario no necesariamente es parte de la solución del problema. La solución del problema la construye el analista. El usuario explica el problema, debería explicar cuál es el problema y debería transferir los conocimientos sobre el problema, pero la solución la construye el analista, los ingenieros de software y cuando hablamos de proveer la solución de software no sólo estamos hablando de software ejecutable, estamos hablando de construir una especificación de requerimientos una representación de software

Entonces esa primera versión del software generalmente en lenguaje natural la construye el analista, el ingeniero de requerimientos, el ingeniero de software.

Esta figura sintetiza una situación que muchas veces se da respecto a lo que solicita el usuario y lo que realmente necesita y aquí es donde queda en evidencia que la solución la debe construir el ingeniero de software.

**Validación de requerimientos**

Entonces, la validación de requerimientos es el proceso de certificar la corrección del modelo de requerimientos (RM) contra las intenciones de los usuarios.

•RV (Requirements Validation): ayuda a hacer el artefacto correcto

•RM (Requirements Modelization): ayuda haciendo correctamente el artefacto

Una primera definición de validación de requerimientos primero tiene que ver con que es un proceso organizado, que tiene un conjunto de pasos y ese proceso tiene como propósito certificar la corrección de la especificación o del modelo de requerimientos contra lo que pretende el usuario. Por eso la validación de requerimientos ayuda a realizar el artefacto de software correcto .

Vamos a ver más adelante que un conjunto de técnicas disponibles, ( prototipos animación y paráfrasis del lenguaje natural), para utilizarlas como técnica de validación.

Según IEEE la validación tiene que ver con que al final del desarrollo hay que evaluar el software para asegurar que cumple con los requerimientos. IEEE también menciona verificación y a diferencia de la validación la verificación consiste en determinar si una fase del producto de software cumple con los requerimientos de la fase anterior .

**V&V - IEEE**

**Validación:** al final del desarrollo evaluar el software para asegurar que el software cumple los requerimientos

**Verificación:** determinar si un producto de software de una fase cumple los requerimientos de la fase anterior

**Sobre estas definiciones:**

•la validación sólo se puede hacer con la activa participación del usuario

•validación: hacer el software correcto

•verificación: hacer el software correctamente

**Proceso:**

•validación: tiene lugar cuando el software ha sido desarrollado para asegurar el cumplimiento de los requerimientos

•verificación: se ejecuta siempre contra algunos requerimientos

**Dos preguntas:**

•1.¿Es suficiente validar después del desarrollo del software?

•2.¿Contra qué se verifican los requerimientos?

**Los dilemas de la validación**

**¿Validar después del desarrollo?**

* La evidencia estadística dice NO
* Cuanto más tarde se detecta, más cuesta corregir (Boehm)
* Efectos del modelo Mizuno
* Bola de nieve de defectos

Validar en el fase de especificación de requerimientos puede ayudar a evitar costosas correcciones después del desarrollo

¿Contra qué verificar los requerimientos?

* Contra nada
* No existen “los requerimientos de los requerimientos”
* Si los hubiera estarían en la cabeza del usuario.
* No puede probarse formalmente que un RM es correcto. Puede alcanzarse una convicción de que la solución especificada en el modelo de requerimientos es el correcto para el usuario.

Pensemos en estas preguntas. Realmente es útil validar sólo después del desarrollo ?. y la evidencia estadística que dice que no, porque porque si nosotros validamos sólo después del desarrollo , cuando lo entrego o lo estoy por entregar, podemos detectar errores. Cuando más tarde detecto un error de requerimientos más cuesta corregir ese error porque tengo que volver hacia la fase donde se produjo. Aqui vemos que puede generarse una propagación de esos errores, más si ese error se produce en etapas iniciales y queda latente, Cuanto más tarde se detecta ese error, mayores consecuencias trae. La posible solución sería validar en la fase de especificación de requerimientos donde nos puede ayudar a evitar costosas correcciones después del desarrollo.

¿Contra que verificar los requerimientos?

por ejemplo, tomamos el producto de software "especificación de requerimientos" que es un producto de software. Contra que vamos a verificar los requerimientos?, con nada porque no hay una fase anterior con requerimientos. No existen los requerimientos de los requerimientos y si existieran, van a estar en la cabeza del usuario. Entonces no podemos verificar los requerimientos como plantea la IEEE, pero sí podemos alcanzar la convicción de que ese modelo de requerimientos es correcto para el usuario.

**Características**

La validación de los requisitos es más una actividad de debugging que de probar corrección. La meta es identificar y corregir errores, en la fase de requerimientos y no más tarde o cuando el software esté desarrollado. Es una actividad que está siempre presente .

Otra característica es que es un proceso no estructurado, o sea no tiene una solución algorítmica.

**Tareas**

Hay una serie de tareas de validación: aplicables a todas las metodologías y formalismo usados para construir un modelo de requerimientos,

El propósito es identificar una serie de propiedades deseables en la RM.

**Modelo de requerimientos: Propiedades deseables.**

* Consistencia interna
* No-ambigüedad
* Consistencia externa
* Minimalidad
* Completitud
* Redundancia

Que explicaremos a continuación.

**Consistencia interna**

Un RM es consistente internamente cuando se derivan conclusiones no contradictorias.

Los requerimientos contradictorios pueden ser implícitos: uno o ambos pueden ser implicados por otros requerimientos. Esto dificulta la tarea de validación, pues no pueden revisarse todas las implicaciones.

La alternativa que tenemos cuando no podemos manejar esas implicancias, es la lógica matemática. El uso de la lógica va a facilitar el chequeo de esas consistencias porque los requerimientos se ven como sentencias. Por ejemplo los requerimientos complejos se arman con conectores and u or

•Facilita el chequeo de la consistencia

•Los requerimientos se ven como sentencias

•Los requerimientos complejos se arman con los conectores and, or e implica.

Y como ventaja puedo trabajar con un programa que puedan derivar esos requerimo y descubrir ciertas contradicciones

**No-ambigüedad**

CUando hablamos de no ambiguedad, decimos que un requerimiento no puede interpretarse en más de una forma. La ambigüedad se introduce fácilmente, en especial con el uso del lenguaje natural.

¿Como se elimina?:Con sentido común, reglas de lógica, trabajar y clarificar en conjunto con los usuarios

**No-ambigüedad Lenguaje Natural**

El lenguaje natural, como hemos visto en clases anteriores, es ambiguo.

•Oxford English Dictionary: las 500 palabras más usadas en Inglés, tienen un promedio de 23 significados diferentes.

•Ejemplo: "round" tiene 70 significados distintos.

•[Collection of Ambiguous or Inconsistent//Incomplete Statements, (http://www.vuse.vanderbilt.edu/~jgray )]

•Real Academia: saltar tiene 24 significados, salto tiene 17 y además salto con pértiga, de agua, de altura, de caballo, de cama, de campana, de carnero, de la garrocha, de lobo, de longitud, de mal año, de mata, etc. etc.

**Consistencia Externa**

La idea es establecer un acuerdo entre lo que se establece en la RM y lo que es cierto en el dominio del problema. Esto requiere esfuerzo por asegurar la concordancia entre el RM y el dominio del problema.

**Minimalidad**

•Tratar de evitar la sobreespecificación (=incluir en el modelo más de lo necesario; por ej: prescribir soluciones de diseño)

•No incluir algo que puede limitar innecesariamente las opciones de diseño.

**Completitud**

Un RM completo no omite información esencial sobre el dominio del problema que podría resultar en un sistema que no satisface las necesidades del usuario.

•No hay procedimientos formales para demostrar completitud.

Un RM debe contener:

•objetivos a alcanzar

•reglas y hechos

•restricciones

No capturar alguno de estos elementos impacta la corrección del modelo.

¿Cuáles son las consecuencias de omitir objetivos? Puede generar un modelo que no represente las necesidades del usuario

Si omitimos reglas o hechos, puede inhibir al modelo de representar adecuadamente el dominio

si omitimos alguna restricción, puede llevar a un incorrecto comportamiento del sistema

•Resumen: objetivos afectan las necesidades del usuario; reglas y hechos la representación del dominio y restricciones al comportamiento del sistema.

**Completitud Tipos de chequeos**

•Chequeos automatizables: ej. el control de nombres.

Algunas formas de chequear son

•mirar sistemas con requerimientos similares

•asumir una organización jerárquica de los requerimientos

•Una buena forma de chequear completitud del RM es usar prototipos.

**Redundancia**

un requerimiento es redundante cundo se puede obtener de otra parte del RM o es algo no está deseado del sistema. Que no necesita el usuario

•Ideal: no tener múltiples definiciones

Puede haber requerimientos explícitos y además deducidos (lógicamente) de otro explícito. en este caso:

•el explícito puede removerse si no hay peligro de ignorar la implicación

•el explícito puede mantenerse, pero con una referencia al que lo implica

Establecer la redundancia de un requerimiento, depende de:

•la comprensión de las expectativas del usuario y de la factibilidad de los requerimientos propuestos.

Por eso existe un enfoque de asignación de prioridades a los requerimientos.

**Validación y calidad**

La validación: proceso de aseguramiento de la calidad que se aplica a los requerimientos

La calidad tiene que ver con el tiempo y recursos asignados a la validación y criterios de calidad.

**Los mundos de la validación**

tenemos dos mundos. el mundo ideal y el mundo real.

**En el mundo ideal:**

•El proceso de validación lleva tanto tiempo como se quiera.

•El RM es chequeado en profundidad por el analista por los puntos señalados

•El comportamiento del sistema se exhibe con técnicas como prototipos, animación

•El modelo se modifica hasta que satisface las expectativas del usuario.

•En los cambios se chequea su impacto

**En el mundo real:**

La RV toma sólo una pequeña parte del tiempo del ciclo de vida del desarrollo del software (SDLC) y posiblemente no pueda durar todo el tiempo.

•Hay una tendencia de los desarrolladores a moverse hacia áreas más “desafiantes” como el diseño

•En general los usuarios no tienen tiempo suficiente para colaborar en la terminación del RV pero tienen que contribuir en todas las fases. importante. avisar con anticipación al usuario de las actividades que se deben cumplimentar.

**Herramientas disponibles**

•Organización lógica del RM: muestra la interdependencia entre los requerimientos

•Herramientas automáticas que evitan errores administrativos y aumentan la velocidad de la validación

•Comunicación con el usuario, con técnicas como simulación y prototipos

•Usar experiencia de otros sistemas similares, reusar RM

**Prototipos (prototipos rápidos)**

Es un proceso de construcción y evaluación de modelos de trabajo de un sistema para aprender de ciertos aspectos del sistema requerido y/o su solución potencial. es una técnica común de la ingeniería.

En Ingeniería de Software: es el paradigma de producir software tan rápido y económicamente como sea posible en alguna etapa del desarrollo.

Un modelo es considerado un prototipo si es posible obtener información sobre el comportamiento y performance del sistema que se “prototipea”. Construirlo debería ser un proceso rápido. También se pueden aplicar en cualquiera de las tres fases de la ingeniería de requerimientos.

•Nos interesa en la validación: proceso de certificación de la corrección del RM contra las intensiones del usuario. Como el usuario participa o debería participar en el proceso de validación, la información que se provee tiene que tener significado para el usuario.

**Animación 🡪 Se puede incluir dentro de prototipos**

Es otra herramienta que es una visión gráfica múltiple de un proceso en acción donde se representan gráficamente los principales objetos y se interactúa con ellos en tiempo real.

Por ej.: el usuario cambia el estado de un proceso con el mouse y analiza sus consecuencias.

Es util en sistemas en tiempo real.

•Tienen nociones comunes: proceso, sincronización de procesos, mensajes, timers

•Clave en la validaci´´on de requisitos: concurrencia y restricciones de tiempo

* LOS PROTOTIPOS ANIMACION AYUDAN A VALIDAR Y DETECTAR ERRORES

**Características**

•Ventaja sobre prototipos: no impone decisiones de diseño muy tempranas

•Provee una indicación de la dinámica del sistema mediante una recorrida

**Paráfrasis del lenguaje natural**

Paráfrasis significa explicar algo de otra forma.

•Ataca el doble conflicto:

•necesidad del analista de un modelo formal

•necesidad del usuario de comunicarse en su terminología

•Provee una versión amistosa del RM

•No necesita aplicarse a todo el RM

Un ejemplo claro es el Manual del usuario

Es una alternativa de reescritura de los requerimientos y debería incluir:

• Una descripción de la funcionalidad a implementar y cómo acceder desde la interface del usuario

•Deberían dejar claro lo no implementado

•una descripción de cómo recuperarse de las dificultades

•Si los usuarios instalan el prototipo, descripción de cómo instalarlo

•El producto puede ser la base para la versión final

Resumen de la validación de requerimientos.

Por ser La última fase no la tenemos que subestimar porque pueden existir errores que sobreviven a la fase de requerimientos pueden tener consecuencias catastróficas.

•No puede ejecutarse contra las intenciones del usuario, como dijimos anteriormente , la participación de éste es crucial.

.La corrección de los requerimientos no puede probarse formalmente, pueden ser chequeadas por consistencia, minimalidad, no-redundancia, etc.

•Uno de los mayores problemas es que el usuario entienda los modelos formales creados por el analista, técnicas para evitar este problema: prototipos, simulación/animación y parafraseo.