Requisitos funcionales:
Relatos de usuario y casos de uso

Ingeniería de Software – IIC2143

Yadran Eterovic (yadran@ing.puc.cl)

En el mundo del software, todos queremos clientes contentos

El primer paso para producir software de gran calidad es asegurarnos de que hace lo que el cliente quiere que haga

Pero … ¿cómo averiguamos lo que el cliente realmente quiere?

¿Y cómo nos aseguramos de que el cliente sabe lo que realmente quiere?

Hay que hablar con el cliente

El próximo eclipse total de sol

… en el hemisferio sur tendrá lugar el 2 de julio de 2019

En La Serena, va a comenzar alrededor de las 3.30pm, va a ser total a eso de las 4.30pm y por unos 2 minutos, y va a terminar poco después de las 5.30pm

A una amiga tuya, que tiene una agencia de turismo, se le ha ocurrido el negocio del año: un servicio de astro-turismo para ir a La Serena a observar el eclipse (y más adelante a otros lugares en que se pueda observar eclipses totales o anulares de sol)

Tu amiga ya ha logrado una serie de acuerdos con hoteles y otros tipos de hospedaje, líneas aéreas nacionales e internacionales, empresas de buses interurbanos, empresas de arriendos de autos, retaurantes, etc.

Ahora “solo” le falta desarrollar el sistema de reservas de su servicio de astro-turismo:

“… un sitio Web que muestre las ofertas, que permita a tus clientes reservar viajes y paquetes especiales, y pagar sus reservas online; incluso, que te permita ofrecer un servicio de lujo que incluya los viajes desde y hacia la ciudad de origen del cliente, con hospedaje en un hotel de La Serena (o alrededores) y con disponibilidad de auto …”

… para lo que te ha pedido ayuda a ti

Ahora tu tarea es analizar la frase de tu amiga e identificar algunos requisitos iniciales

Un requisito es **una** cosa que el software tiene que hacer

Para cada requisito, dale un título y escribe una descripción breve

Ahora tu tarea es analizar la frase de tu amiga e identificar algunos requisitos iniciales:

"… un sitio Web que muestre las ofertas, que permita a tus clientes reservar viajes y paquetes especiales, y pagar sus reservas online; incluso, que te permita ofrecer un servicio de lujo que incluya los viajes desde y hacia la ciudad de origen del cliente, con hospedaje en un hotel de La Serena (o alrededores) y con disponibilidad de auto …"

Un requisito es una cosa que el software tiene que hacer

Para cada requisito, dale un título y escribe una descripción breve

Los requisitos …

Los **requisitos** son capacidades y condiciones que el sistema debe cumplir

… por supuesto, afectan fuertemente el desarrollo de un proyecto de software

… y su análisis

El **análisis de requisitos** es la tarea de lograr un buen ajuste entre lo que los clientes y usuarios quieren y lo que el sistema hace

Es una de las tareas centrales del desarrollo de software

El mal manejo de requisitos es una de las causas más comunes de fracaso de los proyectos:

- requisitos ambiguos, desactualizados, omitidos, contradictorios, o simplemente erróneos

Un pequeño error de requisitos llega a ser un defecto mayúsculo durante la instalación del producto:

- corregir estos defectos es muy caro —se ha gastado mucho esfuerzo yendo en una dirección equivocada

Habla con el cliente para obtener más información

Siempre va a haber brechas en lo que uno ha entendido sobre lo que el software tiene que hacer, especialmente al principio del proyecto

Cada vez que tengas preguntas, o cuando empieces a hacer suposiciones,

… debes ir y **hablar nuevamente** con el cliente —tu amiga de la agencia de turismo— para que te responda tus preguntas

¿Qué preguntas se les ocurren?

… y aprovecha de identificar requisitos adicionales

… en los que el cliente no había pensado antes

No hay nada peor que terminar el proyecto y que entonces el cliente te diga que se le habían olvidado algunos detalles importantes

Pero encontrar lo que el cliente y los usuarios realmente necesitan es solo el primer paso

Es necesario tener un **proceso de requisitos**

1. Encontrar lo que el cliente y los usuarios realmente necesitan

2. **Comunicar y recordar (es decir, escribir) esas necesidades:**
   - escribirlas de manera clara, tanto para el cliente y los usuarios como para el equipo de desarrollo
   - evitar incluir decisiones de diseño o sobre la interfaz de usuario

3. …

4. …

Los requisitos deben estar orientados al cliente

Un buen requisito es escrito desde la perspectiva del cliente:

- debe describir lo que el software va a hacer para el cliente

Cualquier requisito que el cliente no entienda es una advertencia de peligro:

- no es algo que el cliente hubiera podido pedir

Un requisito debe estar escrito en el “lenguaje” del cliente

Un requisito debería leerse como un “relato del usuario”

Un relato de cómo los usuarios interactúan con el software:

Los relatos de usuario **deben**:

- describir una cosa que el software tiene que hacer para el cliente
- estar escritos usando lenguaje que el cliente entiende
- ser escritos por el cliente
- ser breves (p.ej., no más de tres frases)

Los relatos de usuario **no deben**:

- ser un largo ensayo
- usar términos técnicos que no son familiares para el cliente
- mencionar tecnologías específicas

El proceso para obtener los requisitos de un sistema varía mucho:

- muy informal —unas pocas personas mencionando lo esencial del sistema y procediendo rápidamente al desarrollo mismo
- enfoque "ágil" —constante interacción con los clientes, en lugar de un completo proceso de requisitos al inicio
- "*requirements right first*" —los proyectos industriales grandes destinan un enorme esfuerzo a identificar todos los requisitos correctamente desde un comienzo,

  … aunque a veces dejan espacio para revisiones posteriores a medida que la construcción del software produce nueva información

Se podría pensar en un proceso de requisitos ideal

… en que el sistema es derivado completamente a partir de las necesidades de los usuarios:

- el equipo de analistas va pacientemente preguntando a los clientes qué quieren

  … anotan todas las respuestas, las ordenan, las organizan en un documento de requisitos,

  … y entregan el documento al equipo de desarrollo que entonces implementa lo que los clientes quieren

Pero **casi nunca es así**

… por varias razones

Los "*stakeholders*" a menudo tienen visiones que están en conflicto unas con otras —alguien debe resolver los conflictos

Las demandas de los usuarios a menudo incluyen una mezcla de propiedades simples (o fáciles), factibles y difíciles (o imposibles):

- los usuarios a menudo no entienden claramente lo que es fácil y lo que es difícil
- sólo el equipo de desarrollo puede evaluar el costo técnico de cada funcionalidad

Los usuarios tienden a pensar en términos de sistemas existentes —o, en el otro extremo, sistemas imposibles de construir:

- los desarrolladores están en una buena posición para proponer funcionalidades que los usuarios no se imaginan

Finalmente, factores externos afectan la elección final de funcionalidades —presu-puesto, sistemas existentes

Empleemos técnicas simples para capturar los *objetivos de los usuarios*

Como usuarios, tenemos objetivos

… y queremos que los computadores nos ayuden a lograrlos:

- como *pasajero*, consultar por vuelos
- como *profesor*, poner notas a los alumnos

Las mejores formas para capturar objetivos son simples y familiares:

- facilitan —especialmente a los clientes— contribuir a su definición y revisión, reduciendo el riesgo de equivocarse
- la falta de involucramiento de usuarios es la primera razón por la que los proyectos de software fracasan

Dos técnicas simples:

- **relatos de usuario (ahora)**
- casos de uso (más adelante)

Los *relatos de usuario* fueron desarrollados originalmente como parte de XP

Los relatos de usuario nos ayudan a entender qué necesitan los usuarios

Un relato de usuario es una descripción muy corta de una cosa específica que los usuarios necesitan

Se escribe en una tarjeta, siguiendo el formato de *completar los espacios en blanco*:

> **Como** ⟨ *tipo de usuario* ⟩,
>
> **yo quiero** ⟨ *acción específica que voy a realizar* ⟩
>
> **de modo que** ⟨ *qué quiero que pase como resultado* ⟩

… y se le da un título breve que resume el relato

Pueden ser —en una versión preliminar— simplemente una lista de **cosas que el sistema necesita hacer para el usuario**

Como pasajero,

… quiero poder indicar adónde y cuándo quiero viajar.

… quiero poder ver una lista de vuelos que me sirven.

… quiero poder elegir el asiento en el vuelo que reservé.

Los **detalles** del comportamiento del sistema **no aparecen** en estas declaraciones breves:

- los dejamos para ser desarrollados después mediante conversaciones y criterios de aceptación entre el equipo y el dueño del producto

Ejemplo 1: Un relato de usuario de una aplicación para manejar películas

Como un entusiasta de las películas (<rol>)*,*

… yo quiero poder agregar películas a la base de datos (<lo que yo hago con el sistema>)

… de modo que yo pueda compartir una película con otros entusiastas (<valor de negocio que yo recibo>)

Ejemplo 2: Un relato de usuario de una aplicación de *e-commerce* para productos electrónicos

Como cliente (<rol>)*,*

… yo quiero poder comparar dos productos (<lo que yo hago con el sistema>)

… de modo que yo pueda entender fácilmente las diferencias en cuanto a atributos y precio (<valor de negocio que yo recibo>)

Usamos los relatos de usuario para definir dos cosas:

El comportamiento del sistema

El valor de ese comportamiento para el usuario

Los relatos de usuario son inseparables del foco en la entrega de valor

Así lo explican Beck y Fowler en “Planning Extreme Programming” [2005]

*“El relato es una **unidad de funcionalidad** en un proyecto.*

Demostramos progreso entregando código validado e integrado que implementa un relato.

Un relato debiera ser inteligible y valioso para los clientes, validable (testable) por el desarrollador, y suficientemente pequeño de modo que el programador pueda construir media docena en una iteración.”

Relatos de usuario (títulos) para el servicio de reservas de la empresa de astro-turismo

Mostrar las ofertas

Reservar un viaje

Reservar un paquete

Pagar online

Reservar traslado desde y hacia el aeropuerto

Reservar un hotel

…

Otros ejemplos

Como ayudante del doctor, necesito crear registros de pacientes para los nuevos pacientes

Como ayudante del doctor, necesito buscar y obtener los registros de pacientes usando la fecha de nacimiento o el apellido y nombre del paciente

Como ayudante del doctor, necesito saber dónde están los registros de los exámenes médicos de un paciente

Como doctor, necesito documentar todas las conversaciones con un paciente

Como doctor, frecuentemente necesito revisar las conversaciones con un paciente para encontrar cierta información

Como doctor, necesito revisar los registros de los exámenes médicos de un paciente

Como doctor, quiero tener acceso al sistema remotamente desde mi casa o desde donde me encuentre

Como encargado de seguridad, quiero proteger el sistema de accesos no autorizados

Los cuatro pasos del proceso de requisitos

1. Encontrar lo que el cliente y los usuarios realmente necesitan

2. Comunicar y recordar (es decir, escribir) esas necesidades:
   - escribirlas de manera clara, tanto para el cliente y los usuarios como para el equipo de desarrollo
   - evitar incluir decisiones de diseño o sobre la interfaz de usuario

3. Reducir el número y complejidad de los requisitos:
   - resolver requisitos en conflicto, eliminar requisitos redundantes, separar los requisitos funcionales de los no funcionales

4. Asegurarse de que se le pueda “seguir la pista” a los requisitos:
   - saber qué requisitos definen lo que este componente tiene que hacer
   - saber qué parte del código implementa un determinado requisito
   - saber qué pruebas pasó el código para poder asegurar que un determinado requisito está correctamente implementado

Un **proceso de requisitos** debiera producir dos resultados concretos:

- un **documento de requisitos**,

  … que describe las características del software que se va a construir

- un **plan de V&V** (o un plan de pruebas),

  … que describe cómo ese software futuro, una vez construido, va a ser evaluado

El **plan de V&V** es a menudo ignorado … pero

- el momento de producir un buen plan de pruebas es antes de que el software haya sido construido
- esto asegura que las pruebas evalúan si el sistema satisface su verdadero propósito
- mientras más tarde se definen las pruebas, más probable es que estén influenciadas por el diseño e implementación elegidos y menos por las necesidades de los usuarios

Una distinción importante es entre requisitos *funcionales* y requisitos *no funcionales*:

- **requisitos funcionales** especifican las respuestas del sistema

  p.ej., “Si el dato en el campo del RUT es un número de RUT válido, entonces el sistema mostrará el nombre y apellido de la persona correspondiente.”

- **requisitos no funcionales** especifican todas las otras propiedades de un sistema —desempeño, disponibilidad, facilidad de uso, etc.

  p.ej., “Mostrar el nombre y apellido deberá tomar no más de 0.2 segundos en el 99.5% de las consultas.”

Técnicas para recolección de requisitos

Entrevistas:

- preguntamos a los representantes de cada categoría de “stakeholders” qué esperan del nuevo sistema (o de la extensión al sistema existente)
- las entrevistas deben prepararse —preguntas sobre problemas predefinidos y preguntas abiertas que permiten a los “*stakeholders*” describir sus pensamientos libremente
- conviene grabar las entrevistas

Talleres:

- juntar a varios "*stakeholders*" en una misma sala para discutir sobre las propiedades deseables puede ser más eficiente
- promueve la discusión —las diferentes visiones aparecen temprano y pueden ser reconciliadas mediante interacción directa