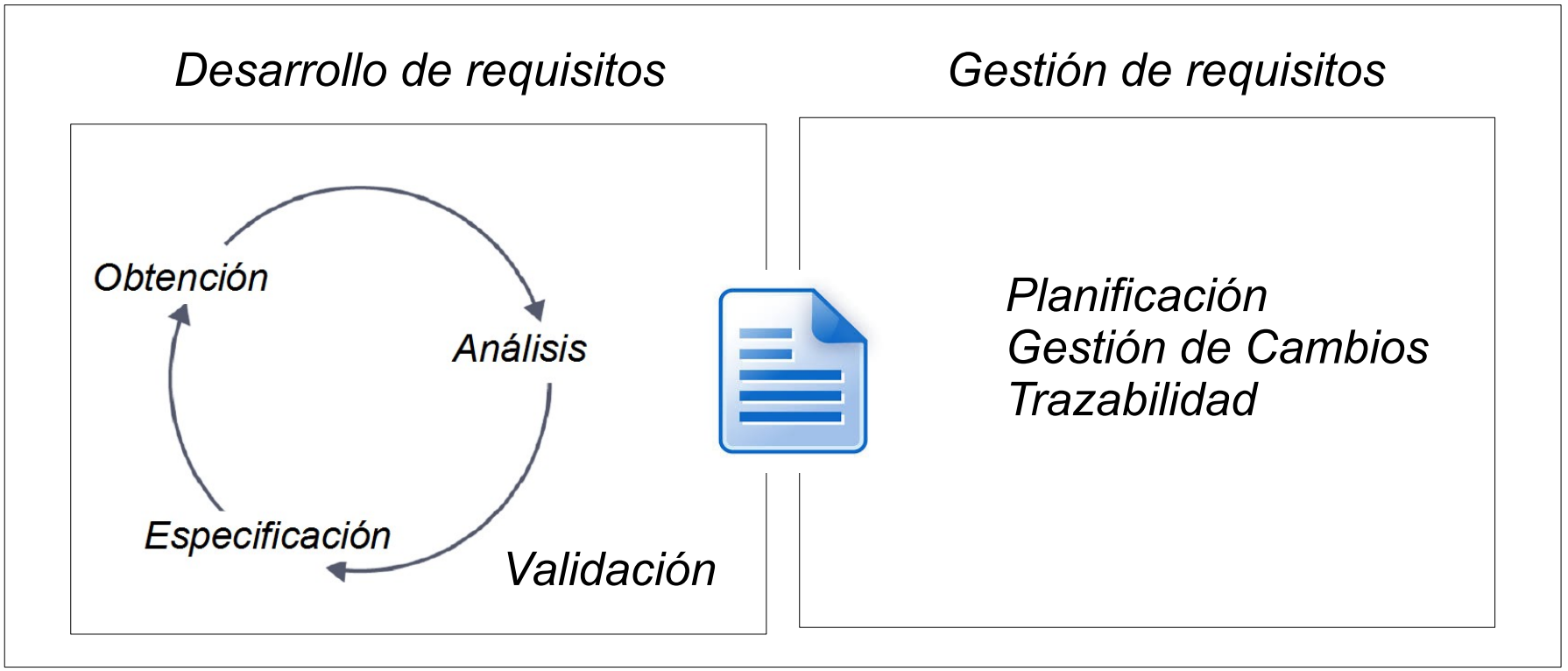
Ingeniería de Software

Ingeniería de Requisitos
Clase 3

Sommerville capítulo 4 – secciones 4.1, 4.3 y 4.4
Wiegers – capítulos 1 y 11

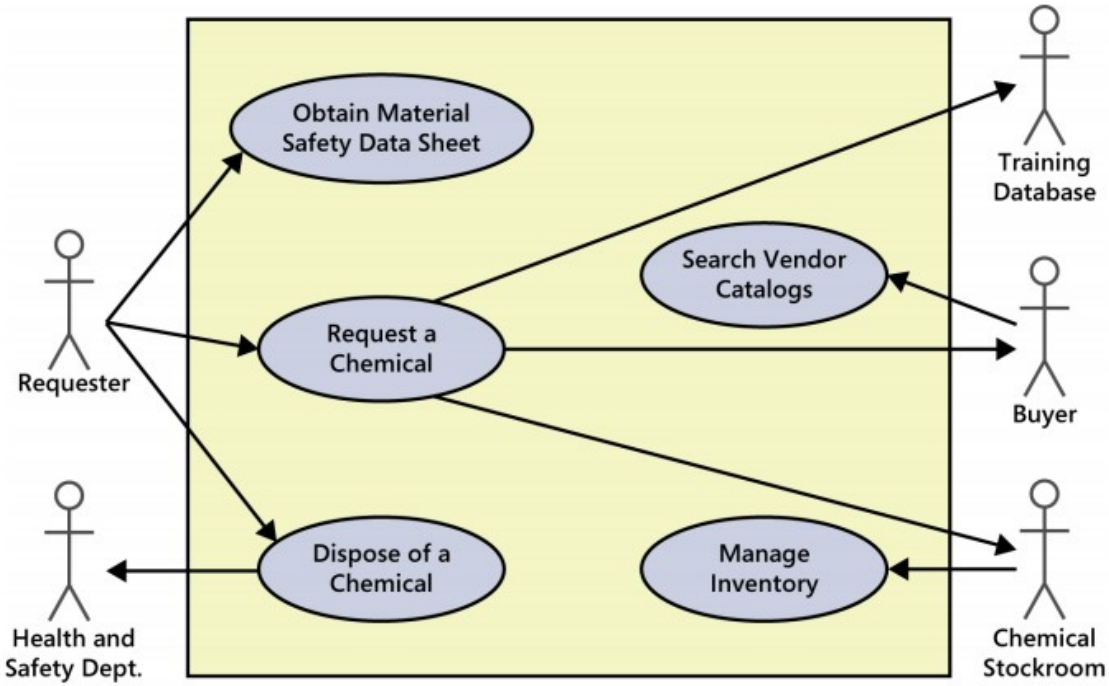
User Stories applied for agile software development – capítulos 1, 2, 3 (User Roles) y 7

Actividades de la ingeniería de requisitos



Análisis de requisitos

- El análisis se da naturalmente junto a las actividades de obtener y especificar los requisitos, conceptualmente implica:
  - Detectar y resolver conflictos entre los requisitos.
  - Descubrir las fronteras del software y cómo debe interactuar con los ambientes organizacional y operacional.
  - Elaborar los requisitos del sistema para derivar los requisitos del software.

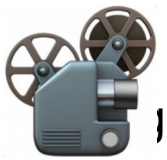


Análisis de requisitos

- Las actividades incluyen:
  - **Clasificación de requisitos**
    - Utilizando, por ejemplo, las categorías que vimos la clase pasada.
    - Prioridad, alcance y estabilidad de los requisitos.
  - **Modelado conceptual**
    - Realizar modelos resulta clave para el análisis de requisitos. Ayudan a entender la situación en la cual ocurren los problemas o necesidades, así como representar soluciones.
  - **Diseño de la arquitectura y asignación de requisitos**
  - **Negociación de requisitos**
    - Resolución de conflictos. Priorización.
  - **Análisis Formal**
    - Muy recomendado para ciertos dominios de aplicación.

Especificación de requisitos

- Es el proceso de escribir los requisitos de usuario y del sistema en un documento de requisitos.
- Los requisitos de usuario deben ser entendidos por los usuarios finales y los clientes que no tienen formación técnica.
- Los requisitos del sistema son más detallados y pueden incluir mas información técnica.
- Los requisitos pueden ser parte de un contrato para el desarrollo del sistema. Por lo tanto, se debe intentar que sean lo más completos posible.



Formas de escribir una especificación de requisitos

Notación	Descripción
Lenguaje natural	Los requisitos son escritos usando frases numeradas escritas en lenguaje natural. Cada frase debe expresar un requisito.
Lenguaje natural estructurado	Los requisitos son escritos en lenguaje natural utilizando un formulario estándar o plantilla. Cada campo proporciona información acerca de un aspecto del requisito.
Lenguajes de descripción de diseño	Este enfoque usa un lenguaje como un lenguaje de programación pero con características más abstractas para especificar los requisitos mediante la definición de un modelo operativo del sistema. Ahora este enfoque es raramente usado aunque puede ser útil para las especificaciones de interfaz.
Notaciones graficas	Modelos gráficos complementados por anotaciones de texto son usados para definir los requisitos funcionales para el sistema. Comúnmente se utilizan diagramas UML de casos de uso y diagramas de secuencia.
Especificaciones matemáticas	Estas notaciones son basadas en conceptos matemáticos tales como maquinas de estados finitos o conjuntos. Aunque estas especificaciones pueden reducir la ambigüedad la mayoría de los clientes no entienden la especificación formal. Ellos no pueden chequear que se representa lo que ellos quieren y se resisten a aceptarlo como un contrato del sistema.

Requisitos y diseño

- En principio, los requisitos deben indicar lo **qué** el sistema debe hacer y el diseño debe describir **cómo** lo debe hacer.
- En la práctica, es prácticamente imposible excluir toda la información de diseño al especificar en un nivel adecuado los requisitos de software.
  - La arquitectura del sistema puede ser diseñada para estructurar los requisitos.
  - El sistema puede interactuar con otros sistemas que generan requisitos de diseño.
  - El uso de una arquitectura especifica para satisfacer los requisitos no funcionales puede ser un requisito.

Especificación en lenguaje natural

- Los requisitos son escritos en frases en lenguaje natural.
- Se utiliza para escribir los requisitos porque es expresivo, intuitivo y universal. Esto significa que los requisitos pueden ser entendidos por los usuarios y los clientes.

Especificación en lenguaje natural - Ejemplo

- 3.2 El sistema debe medir el azúcar en la sangre y liberar insulina, si es necesario cada 10 minutos. *(Los cambios en el azúcar en la sangre son relativamente lentos entonces mediciones más frecuentes son innecesarias; mediciones menos frecuentes podría llevar a niveles innecesariamente altos de azúcar.)*

- 3.6 El sistema debe ejecutar un auto-test de rutina cada minuto con las condiciones que deben verificarse y las acciones asociadas definidas en la Tabla 1. *(Una rutina de auto-test puede descubrir problemas en el hardware y software y alertar al usuario sobre el hecho de que la operación normal puede ser imposible.)*

Guía para escribir requisitos

- Crear un formato estándar y usarlo para todos los requisitos.
- Usar un lenguaje de una manera consistente. El lenguaje usado para requisitos obligatorios debe ser el mismo que para requisitos deseables.
- Utilizar texto subrayado para identificar las partes clave de los requisitos.
- Evitar el uso de jerga informática.
- Incluir una explicación (lógica) de porqué es necesario un requisito.

Problemas con el lenguaje natural

- **Falta de claridad**
  - Es difícil la precisión sin hacer que el documento sea difícil de leer.
- **Confusión de requisitos**
  - Los requisitos funcionales y no funcionales tienden a mezclarse.
- **Requisitos mezclados**
  - Varios requisitos diferentes pueden ser expresados juntos.

Especificaciones estructuradas

- Es una aproximación a los requisitos donde la libertad del escritor de los requisitos es limitada y los requisitos son escritos de una manera estándar.
- Esto funciona bien para algunos tipos de requisitos ej: sistemas de control embebidos pero a veces es demasiado rígido para escribir los requisitos de sistemas de negocio.

Especificaciones estructuradas – posibles secciones

- Definición de la función o entidad.
- Descripción de la entradas y de dónde vienen.
- Descripción de las salidas y a dónde ir.
- Información acerca de lo datos necesario para el calculo y otras entidades usadas.
- Descripción de las acciones a tomar.
- Pre y post condiciones (si es apropiado).
- Los efectos secundarios (si hay alguno) de la función.

Especificación estructurada - ejemplo

Insulin Pump/Control Software/SRS/3.3.2

Function	Compute insulin dose: Safe sugar level.
Description	Computes the dose of insulin to be delivered when the current measured sugar level is in the safe zone between 3 and 7 units.
Inputs	Current sugar reading (r2), the previous two readings (r0 and r1).
Source	Current sugar reading from sensor. Other readings from memory.
Outputs	CompDose—the dose in insulin to be delivered.
Destination	Main control loop.
Action	CompDose is zero if the sugar level is stable or falling or if the level is increasing but the rate of increase is decreasing. If the level is increasing and the rate of increase is increasing, then CompDose is computed by dividing the difference between the current sugar level and the previous level by 4 and rounding the result. If the result, is rounded to zero then CompDose is set to the minimum dose that can be delivered.
Requirements	Two previous readings so that the rate of change of sugar level can be computed.
Pre-condition	The insulin reservoir contains at least the maximum allowed single dose of insulin.
Post-condition	r0 is replaced by r1 then r1 is replaced by r2.
Side effects	None.

Especificación tabular

- Usada para complementar el lenguaje natural.
- Particularmente útil cuando se tienen que definir una serie de posibles flujos alternativos.
- Por ejemplo, el sistema de la bomba de insulina basa sus cálculos en la tasa de cambio de nivel de azúcar en la sangre y la especificación tabular explica como calcular los requisitos de insulina para diferentes escenarios.

Especificación tabular - ejemplo

Condition	Action
Sugar level falling (r2 < r1)	CompDose = 0
Sugar level stable (r2 = r1)	CompDose = 0
Sugar level increasing and rate of increase decreasing ((r2 − r1) < (r1 − r0))	CompDose = 0
Sugar level increasing and rate of increase stable or increasing ((r2 − r1) ≥ (r1 − r0))	CompDose = round ((r2 − r1)/4) If rounded result = 0 then CompDose = MinimumDose

Características deseables de un requisito según Wiegers

- **Completo** — contiene toda la información para entenderlo.
- **Correcto** — describe de forma precisa una necesidad y de forma clara la funcionalidad a desarrollar para cumplirla.
- **Factible** — es posible implementarlo dada las capacidades y limitaciones del sistema y su ambiente operacional.
- **Necesario** — tiene cierto valor para el negocio.
- **Priorizado** — fue priorizado según su importancia utilizando la perspectiva de todos los interesados.
- **No ambiguo** — no admite múltiples interpretaciones.
- **Verificable** — se pueden realizar pruebas para determinar sí se encuentra presente en un producto de software o no.

Características deseables de un conjunto de requisitos según Wiegers

- **Completo** — no hay información necesaria o de requisitos ausente.
- **Consistente** — no hay requisitos en conflictos con otros.
- **Modificable** — es posible reescribir un requisito.
- **Trazable** — un requisito puede ser rastreado hacía atrás a su origen y hacía adelante a elementos de diseño, código implementado y pruebas que verifican su implementación.

Guías para la escritura de requisitos

- **Perspectiva del sistema o del usuario**
  - *[optional precondition] [optional trigger event] the system shall [expected system response].*
  - Si el químico solicitado se encuentra en el almacén químico, el sistema mostrará una lista de todos los contenedores del producto químico que actualmente hay en el almacén.
  - *The [user class or actor name] shall be able to [do something] [to some object] [qualifying conditions, response time, or quality statement].*
  - El químico deberá ser capaz de volver a ordenar cualquier producto químico que ha ordenado en el pasado, para lo cual podrá recuperar y editar una orden antigua.

Guías para la escritura de requisitos

- **Estilo de escritura**
  - Tratar de incluir la frase clave al principio de cada requisito (la declaración de la necesidad o de la funcionalidad) y luego los detalles accesorios (justificación, origen, prioridad, etc.)
  - Evitar utilizar voces activas y pasivas de forma intercalada. Es más, es mejor usar voz activa.
  - No utilizar múltiples términos para el mismo concepto.
  - Escribir frases completas y utilizar correctas gramática, ortografía y puntuación.
  - Mantener las oraciones y los párrafos cortos y directos.
  - Evitar escribir largos párrafos con más de un requisito.

Guías para la escritura de requisitos

- **Nivel de detalle**
  - Se debería incluir más detalle sí:
    - El trabajo será hecho para un cliente externo.
    - El desarrollo o las pruebas serán tercerizados.
    - Los miembros del equipo del proyecto están dispersos.
    - Las pruebas del sistema se basan en los requisitos.
    - Son necesarias estimaciones precisas.
    - La trazabilidad de los requisitos es importante.
  - Se puede incluir menos detalle cuando:
    - Los clientes están muy involucrados.
    - Los desarrolladores tienen mucha experiencia en el dominio.
    - Se dispone de precedentes (software a reemplazar)
    - Se usará una solución empaquetada.

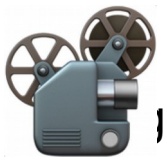
Guías para la escritura de requisitos

- **Nivel de detalle**
  - No todos los requisitos tienen que tener el mismo nivel de detalle. Aunque se recomienda mantener el mismo nivel en requisitos relacionados.
  - Una guía útil es escribir requisitos testeables de forma individual.
- **Evitar la ambigüedad**
  - Utilizar términos de forma consistente y como están definidos en el glosario.
  - Evitar adverbios (generalmente, rápidamente, etc)
  - Tratar de escribir todos los requisitos de forma positiva.
- **Evitar incompletitud**
  - Incluir operaciones simétricas (guardar borrador/recuperar)
  - Revisar en búsqueda de excepciones que falten.

Ingeniería de Requisitos en Procesos Ágiles

Pregunta general

En un desarrollo incremental: ¿Qué diferencias les parece que hay entre la ingeniería de requisitos en procesos ágiles y la ingeniería de requisitos en procesos basados en planes?



Ingeniería de Requisitos en Procesos Ágiles

- Principio del manifiesto #1: Nuestra mayor prioridad es satisfacer al cliente mediante la entrega temprana y continua de software con valor.
- Principio del manifiesto #2: Aceptamos que los requisitos cambien, incluso en etapas tardías del desarrollo. Los procesos Ágiles aprovechan el cambio para proporcionar ventaja competitiva al cliente.
- Generalmente los requisitos o características del producto se especifican como Historias de Usuario.

Historias de Usuario (HU)

- Una HU describe una funcionalidad que, por sí misma, aporta valor al usuario. (eXtremme Programing XP)
- Representación de un requisito escrito de manera simple, sencilla y fácil de entender.
- Son una forma rápida de administrar los requisitos cambiantes de un proyecto.
- Una Historia de Usuario se compone de tres elementos (“las tres Cs”):
  - Card (Ficha): Descripción breve de la historia de usuario, utilizada como recordatorio y para planificar.
  - Conversación: Comunicación cara a cara que intercambia no solo información sino también pensamientos, opiniones y sentimientos.
  - Confirmación: Detalles de la historia de usuario para que el equipo sepa lo que tienen que construir y lo que la contraparte espera. Se conoce como Criterios de Aceptación.

Redacción de Historias de Usuario (Card)

- Mike Cohn sugiere una determinada forma de redactar Historias de Usuario bajo el siguiente formato:

  Como [**rol**] quiero [**funcionalidad**] para [**beneficio**]

- Beneficio de la redacción:
  - Primera persona
  - Priorización
  - Propósito
    - Permite al equipo de desarrollo plantear alternativas que cumplan con el mismo propósito cuando el costo de la funcionalidad sea alto o su construcción sea no viable.

Ejercicio - Redacción de una Historia de Usuario

Como empleado de una empresa quiero contactar estudiantes que quieran trabajar en algo relacionado a su carrera para comunicarme con ellos

Redacción de Historias de Usuario

- Cuando una historia de usuario es demasiado grande se la denomina **épica**
- Generalmente se clasifican en una de las siguientes categorías:
  - Historia de usuario compuesta: contiene múltiples historias de usuario
    - "Contactar estudiantes" es una funcionalidad muy importante para un sistema de reclutamiento, pero hay muchas funcionalidades involucradas.
  - Historia de usuario compleja: no se puede separar fácilmente en un conjunto de historias de usuario
    - Si una historia es compleja debido a que existe una incertidumbre asociada a la misma, se puede dividir en dos historias: una para investigar y otra que desarrolle la nueva funcionalidad.
- La determinación final de si una historia tiene el tamaño adecuado se basa en el equipo, sus capacidades y las tecnologías en uso.

Redacción de Historias de Usuario (Confirmación)

- **Criterios de aceptación**
  - Ayudan a entender mejor cómo se espera que el producto se comporte frente a ciertas situaciones.
  - Ayudan a resolver ciertas expectativas y permiten que el desarrollo sea más fluido, claro y con menor incertidumbre para el equipo
  - ¿Cuáles son los criterios de aceptación?

HU: Visualizar locales de comida con delivery

COMO: cliente habitual

QUIERO: ver un listado de locales de comida con delivery

PARA: evaluar las distintas propuestas

Redacción de Historias de Usuario (Confirmación)

- **Criterios de aceptación**
  - Ayudan a entender mejor cómo se espera que el producto se comporte frente a ciertas situaciones.
  - Ayudan a resolver ciertas expectativas y permiten que el desarrollo sea más fluido, claro y con menor incertidumbre para el equipo
  - Ejemplos de criterios de aceptación
    - Los locales de comida con delivery están ordenados según la valoración (mejor valoradas primero) y la ubicación que se encuentra el usuario.
    - Para cada local de comida con delivery se muestra: nombre, foto, calificación, distancia con respecto a la ubicación del usuario y tiempo estimado de entrega.
    - Si el usuario selecciona un local de comida con delivery, se despliegan las comidas que vende el local: nombre, precio y una breve descripción.

HU: Visualizar locales de comida con delivery

COMO: cliente habitual

QUIERO: ver un listado de locales de comida con delivery

PARA: evaluar las distintas propuestas

Redacción de Historias de Usuario (Confirmación)

- **Criterios de aceptación**
  - Aportan más contexto.
  - Sirven de guía para el desarrollo de los test de aceptación.
  - Deben cubrir tanto los casos comunes como los alternativos.
  - La calidad de un criterio de aceptación eficaz se define bajo el método SMART.
    - Specific (Específico)
    - Measurable (Medible)
    - Achievable (Alcanzable)
    - Relevant (Relevante)
    - Time-bound (Temporalmente limitado)

Redacción de Historias de Usuario (Confirmación)

- **Criterios de aceptación:** generalmente se escriben utilizando una de las siguientes técnicas:
  - Técnicas de comportamiento: Dada una condición, cuando ocurre un evento o acción, entonces sucederá una consecuencia. Así se consigue una estructura consistente que se trasladará fácilmente a tests automáticos.
    - Ejemplo criterio de aceptación de la HU “Visualizar delivery”:
      - Si el cliente tiene el gps desactivado, se muestra la última ubicación detectada por el teléfono celular y despliega un listado con las 10 opciones de delivery mejor valoradas en base a dicha ubicación
  - Técnicas de escenarios: Suele definir el escenario normal o usual y un escenario alternativo de la funcionalidad en cuestión, y debe describir cómo el usuario ejecutaría o intentaría ejecutar los diferentes pasos en dichos trayectos.

Redacción de Historias de Usuario (Confirmación)

- **Criterios de aceptación**
  - Técnicas de escenarios: Hay diferentes formas de escribirlos, las más usadas utilizan el lenguaje específico para las descripciones de comportamiento de software conocido como gherkin. La sintaxis de gherkin es la siguiente:

Dado que [**Contexto**] y adicionalmente [**Contexto**],

cuando [**Evento**],

entonces [**Resultado / Comportamiento esperado**]

  - Ejemplo criterio de aceptación de la HU “Visualizar delivery”:

Dado que **el gps está activado**,
cuando se despliegue el listado de deliverys,
entonces **el sistema desplegará una lista con todos los deliverys ordenados según la valoración y la ubicación que se encuentra el usuario**

Redacción de una Historia de Usuario

- Se recomienda que toda Historia de Usuario cumpla con 6 características que se pueden recordar bajo la regla mnemotécnica “INVEST”:
  - Independiente (I): En la medida de lo posible, se debe tener cuidado para evitar la introducción de dependencias entre historias de usuario.
  - Negociable (N): los detalles de una historia de usuario se negociarán en una conversación entre el cliente y el equipo de desarrollo.
  - Valiosa (V): para los clientes o usuarios del software
  - Estimable (E)
    - HU es demasiado grande
    - Falta de conocimiento funcional
    - Falta de conocimiento técnico
  - Pequeña (Small): en esfuerzo
  - Verificable (Testable): una historia necesita poder probarse para poder ser **confirmada**.

Requisitos no funcionales

- Newkirk y Martin (2001) recomiendan la práctica de anotar una historia con la palabra CONSTRAINT para cualquier historia que debe ser obedecida en lugar de implementada directamente.

El software debe ejecutarse en todas las versiones de Windows

CONSTRAINT

Historias de usuario tipo SPIKE

- Tareas de investigación durante una iteración. Por ejemplo: investigar, diseñar, explorar, comprender mejor un requisito, o aumentar la fiabilidad de estimación de una HU.
- Suelen ser de dos tipos: técnicos o funcionales.
  - Técnicos: se utilizan para determinar la viabilidad.
  - Funcionales: se utilizan para analizar funcionalidad, su riesgo y complejidad.

http://www.scaledagileframework.com/spikes

Ejercicio - Redacción de Historias de Usuario