

## Requerimientos Ágiles



# VALORES ÁGILES

INDIVIDUOS E  
INTERACCIONES



**SOBRE**



PROCESOS Y  
HERRAMIENTAS

SOFTWARE  
FUNCIONANDO



**SOBRE**



DOCUMENTACION  
EXTENSIVA

COLABORAR CON  
EL CLIENTE



**SOBRE**



NEGOCIACION  
CONTRACTUAL

RESPONDER  
AL CAMBIO



**SOBRE**

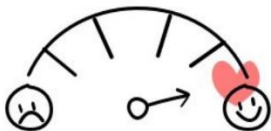


SEGUIR UN PLAN



# Los 12 principios del Manifiesto Ágil

- 1 Nuestra mayor prioridad es **satisfacer al cliente.**



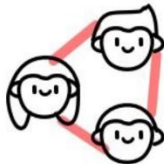
- 2 **Aceptar** que los requisitos **cambien.**



- 3 **Entregar** software funcional **frecuentemente.**



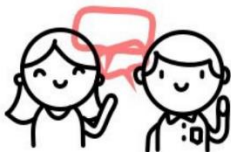
- 4 Los responsables de negocios, diseñadores y desarrolladores deben **trabajar juntos** día a día durante el proyecto.



- 5 Desarrollamos proyectos en torno a **individuos motivados.**



- 6 El método más eficiente de comunicar información es **conversaciones cara a cara.**



- 7 El **software funcionando** es la principal **medida de éxito.**



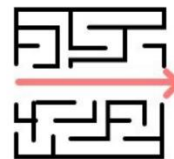
- 8 Los procesos ágiles promueven el **desarrollo sostenible.**



- 9 La **atención continua a la excelencia técnica** y al **buen diseño** mejor la Agilidad.



- 10 La **simplicidad es esencial.**



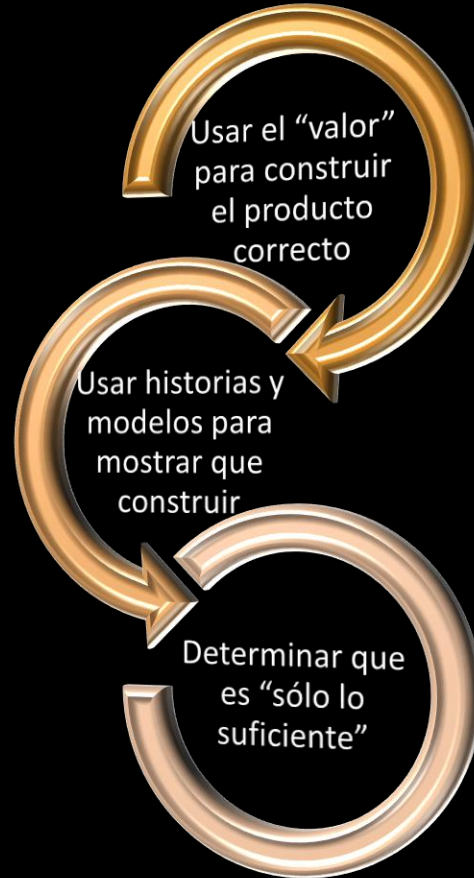
- 11 Las mejores arquitecturas, requisitos, y diseños emergen de **equipos auto-organizados.**



- 12 A intervalos regulares el equipo reflexiona sobre cómo ser más efectivo y de acuerdo a esto **ajustan su comportamiento.**

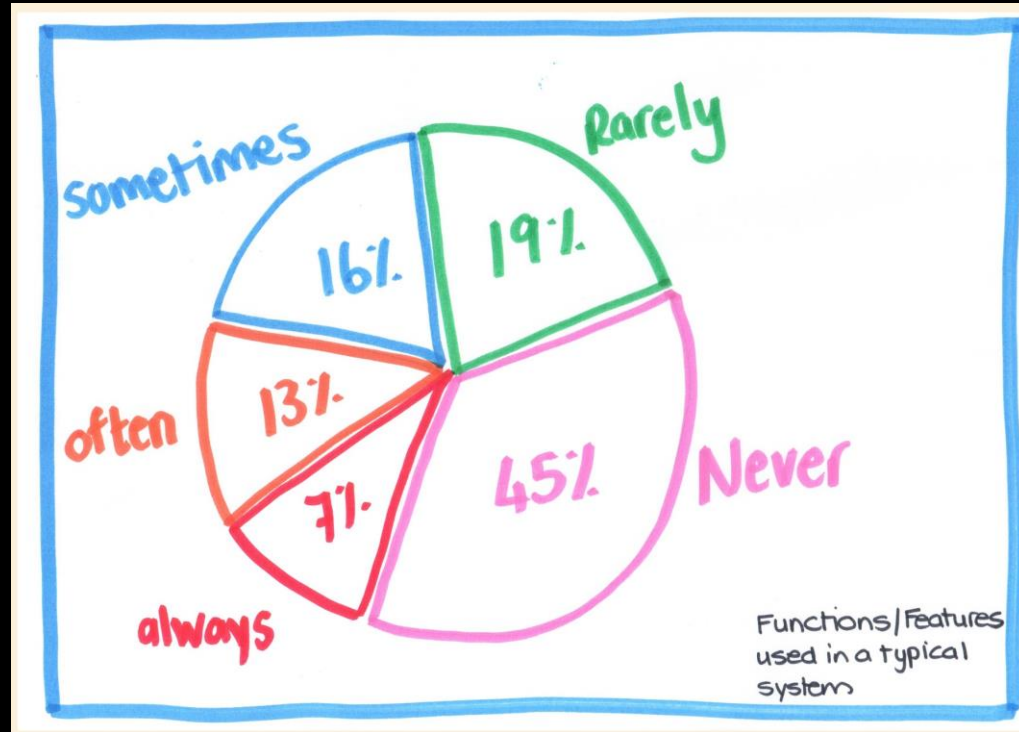


# Requerimientos en Agile



# El costo del tradicional BRUF

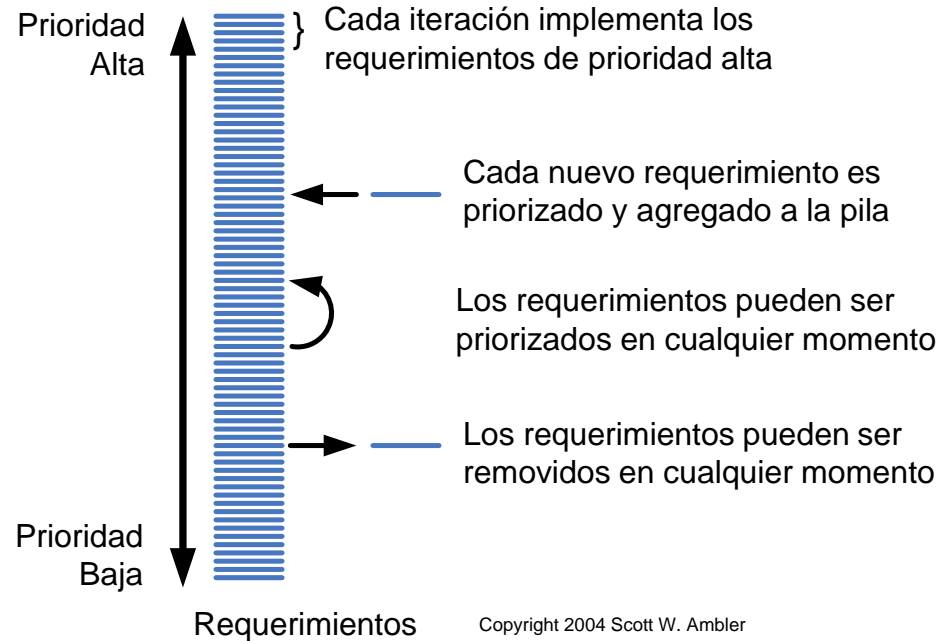
Los productos  
“Exitosos”  
también tienen  
un desperdicio  
significante



Fuente: Jim Johnson of the Standish Group, Keynote Speech XP 2002

# Gestión Ágil de Requerimientos de Software

Los requisitos cambiantes son una ventaja competitiva  
si puede actuar sobre ellos



Copyright 2004 Scott W. Ambler



Analice cuando lo  
necesite, no antes



El cara-a-cara permite que fluya información vocal, subvocal, gestual con realimentación rápida.





*“Valor es la obtención de  
beneficio **tangible** o  
**intangible**”*

Masa Maeda, Serious LeAP

*“El valor lo asociamos a la  
**utilidad, beneficio o**  
**satisfacción** que le ofreces a  
los usuarios finales por cada  
funcionalidad completa que le  
entregas”*

Pablo Lischinsky, Agile Trainer &  
Consultant, Entrepreneur

## Tradicional vs. Ágil

FIJO →

REQUISITOS

RECURSOS

TIEMPO



ESTIMADO →

RECURSOS

TIEMPO

ALCANCE

# Tipos de Requerimientos



## Requerimiento de Negocio

Disminuir un X% de tiempo invertido en procesos manuales relacionados con atención al cliente.

## Requerimiento de Usuario

Realizar consultas en línea del estado de cuenta de los clientes

## Requerimiento Funcional

Generar reporte de saldos de cuenta. Recibir notificaciones por mail.

## Requerimiento No funcional

Formato del reporte PDF. Cumplir niveles de seguridad para credenciales de usuarios según la ley bancaria 9999XX

## Requerimiento de Implementación

Servidores en la nube

**Requerimientos  
de Negocio**

**Requerimientos de  
Usuario**

**Requerimientos de Software**

**Dominio  
de la  
Solución**



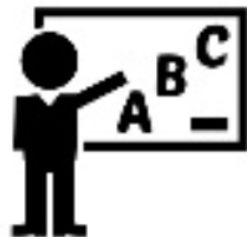
**Dominio  
del  
Problema**



## EN RESUMEN...



ENTENDIENDO LA  
NECESIDAD Y NEGOCIO...



DESCUBRIENDO LA SOLUCIÓN  
DE FORMA COLABORATIVA...



JUNTO A UN EQUIPO  
MOTIVADO Y COMPETENTE...



ENTREGAMOS  
FRECUENTEMENTE VALOR A  
LOS STAKEHOLDERS.

# Por último



Los cambios son la única constante.



Stakeholders: no son todos los que están.



Siempre se cumple eso de que: “El usuario dice lo que quiere cuando recibe lo que pidió”.



No hay técnicas ni herramientas que sirvan para todos los casos.

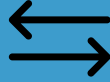


Lo importante no es entregar una salida, un requerimiento, lo importante es entregar, un resultado, una solución de “valor”.

# Principios Ágiles relacionados a los Requerimientos Ágiles



1- LA PRIORIDAD ES SATISFACER AL CLIENTE A TRAVÉS DE RELEASES TEMPRANOS Y FRECUENTES (2 SEMANAS A UN MES)



2 -RECIBIR CAMBIOS DE REQUERIMIENTOS, AUN EN ETAPAS FINALES



4 - TÉCNICOS Y NO TÉCNICOS TRABAJANDO JUNTOS TODO EL PROYECTO



6 - EL MEDIO DE COMUNICACIÓN POR EXCELENCIA ES CARA A CARA



11 - LAS MEJORES ARQUITECTURAS, DISEÑOS Y REQUERIMIENTOS EMERGEN DE EQUIPOS AUTOORGANIZADOS



# User Stories

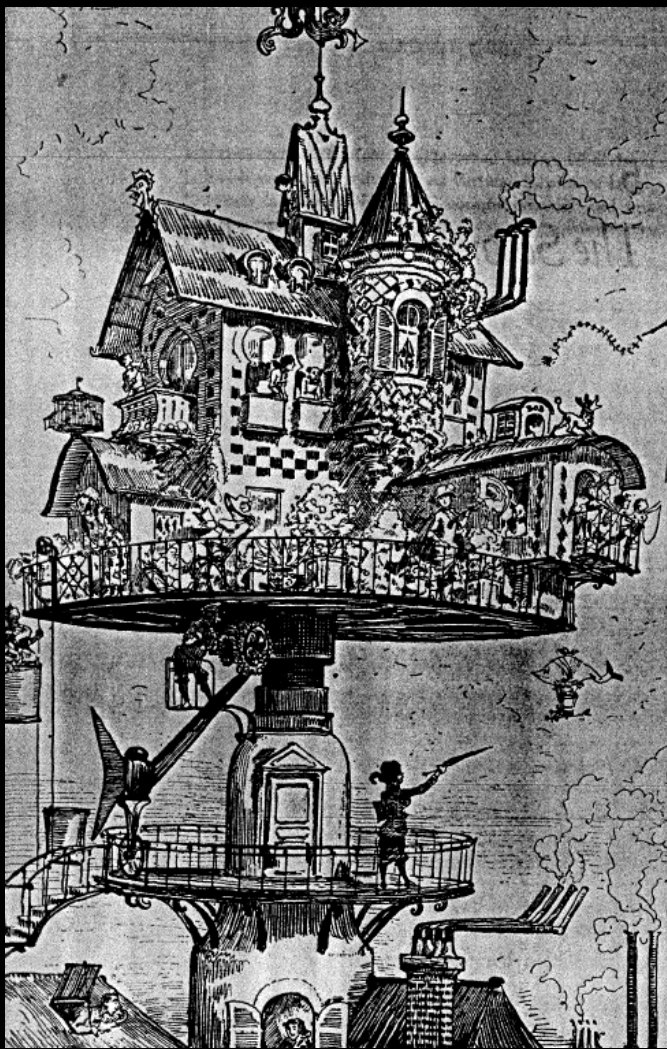
“....se las llama “stories” porque se supone que Ud. cuenta una historia. Lo que se escribe en la tarjeta no es importante, lo que Ud. habla, si!.

--- Jeff Patton, InfoQ,

create conversation.



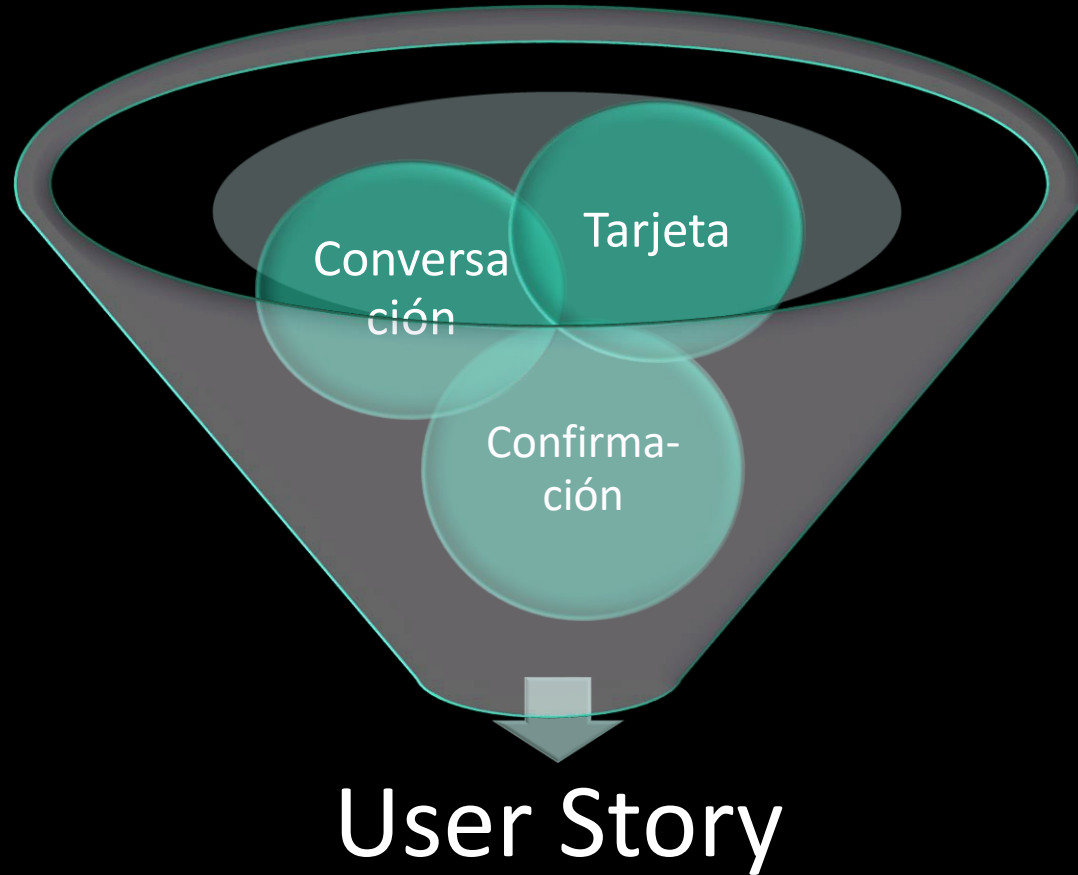
@gapingvoid



La parte más difícil de construir un sistema de software es decidir precisamente qué construir. Ninguna otra parte del trabajo conceptual es tan difícil como establecer los requerimientos técnicos detallados... Ninguna otra parte del trabajo afecta tanto el sistema resultante si se hace incorrectamente. Ninguna otra parte es tan difícil de rectificar más adelante”

Fred Brooks - “No Silver Bullet - Essence and Accidents of Software Engineering”. IEEE Computer, Abril de 1987.

¿Cuáles son las partes de una User Story?



# Forma de expresar las Historias de Usuario



Como **<nombre del rol>**,  
yo puedo **<actividad>**  
de forma tal que **<valor  
de negocio que  
recibo>**

Representa quién está realizando la acción o quién recibe el valor de la actividad.

Representa la acción que realizará el sistema

Comunica porque es necesaria la actividad

# User Story: un ejemplo de tarjeta

Frase verbal

Buscar Destino por Dirección

**Como Conductor quiero buscar un destino a partir de una calle y altura para poder llegar al lugar deseado sin perderme.**

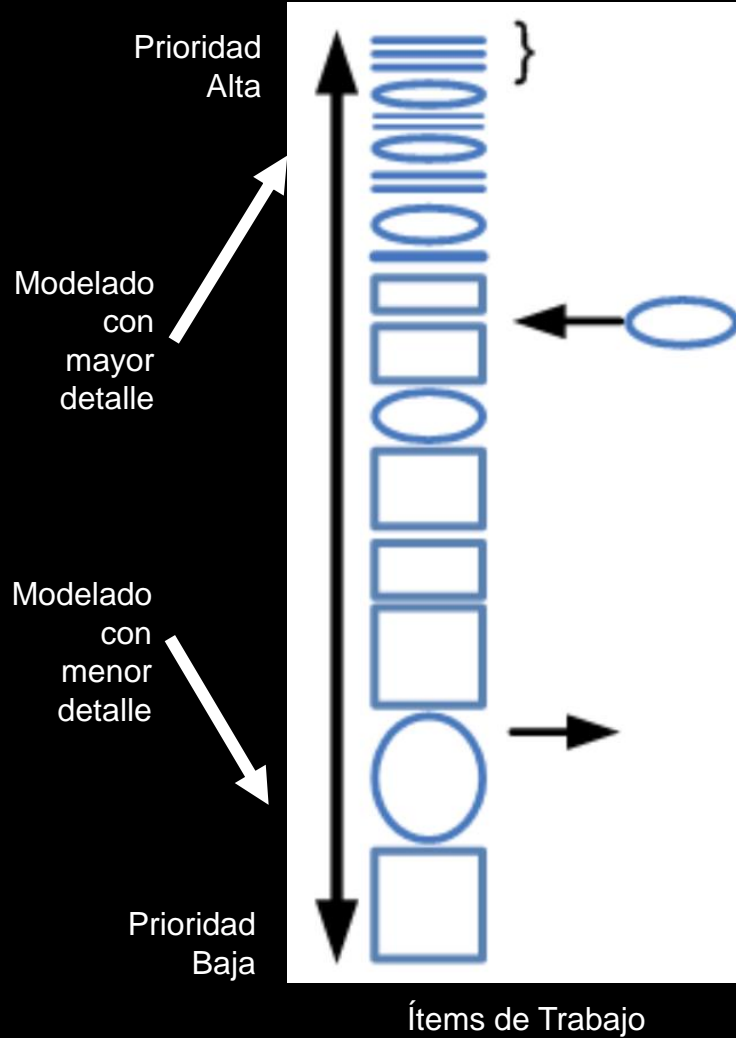
# Las User Stories son Multipropósito

- Las historias son:
  - Una necesidad del usuario
  - Una descripción del producto
  - Un ítem de planificación
  - Token para una conversación
  - Mecanismo para diferir una conversación

*\* Kent Beck coined the term user stories in Extreme Programming Explained 1<sup>st</sup> Edition, 1999*



# El Product Owner Prioriza las historias en el Product Backlog



Cada iteración implementa los ítems de trabajo de mayor prioridad

Cada nuevo requerimiento es priorizado y agregado a la pila

Los ítems de trabajos pueden ser re-priorizados en cualquier momento

Los ítems de trabajo pueden ser removidos en cualquier momento



# User stories: Porciones Verticales



Donde impacta  
"fuertemente"  
esto??

Story 1	Story 2	
GUI		
Business Logic		
Database		

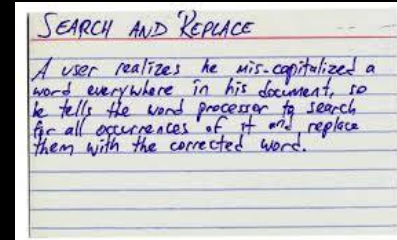
# Modelado de Roles



# Modelado de Roles: Tarjeta de Rol de Usuario

## Rol de Usuario: Reclutador Interno

Nos es un experto en computadoras, pero bastante adepto a utilizar la Web. Utilizará el software con poca frecuencia pero muy intensamente. Leerá anuncios de otras compañías para averiguar cuál es la mejor palabra para sus anuncios. La facilidad de uso es importante, pero más importante es que lo que aprenda, lo pueda recordar meses después.



# Usuarios Representantes (Proxies)

- Tipos de usuarios representantes:
  - Gerentes de Usuarios
  - Gerentes de Desarrollo
  - Alguien del grupo de marketing
  - Vendedores
  - Expertos del Dominio
  - Clientes
  - Capacitadores y personal de soporte.

**No son ideales como  
los usuarios verdaderos  
...EVITELOS !!!**

# Criterios de Aceptación de User Stories



# Criteria de Aceptación de Historias de Usuario

- Definen límites para una user story (US)
- Ayudan a que los PO respondan lo que necesitan para que la US provea valor (requerimientos funcionales mínimos)
- Ayudan a que el equipo tenga una visión compartida de la US
- Ayudan a desarrolladores y testers a derivar las pruebas.
- Ayudan a los desarrolladores a saber cuando parar de agregar funcionalidad en una US



# User Story: un ejemplo de tarjeta

## Buscar Destino por Dirección

**Como Conductor quiero buscar un destino a partir de una calle y altura para poder llegar al lugar deseado sin perderme.**

### Criterios de Aceptación:

- La altura de la calle es un número.
- La búsqueda no puede demorar más de 30 segundos.



# ¿Cuáles son los Criterios de Aceptación buenos?

- Definen una intención, no una solución
  - Ej.: El usuario debe elegir al menos una cuenta para operar
- Son independientes de la implementación
- Relativamente de alto nivel, no es necesario que se escriba cada detalle



# ¿Y los detalles? ¿Dónde van?



- Detalles como:
  - El encabezado de la columna se nombra “Saldo”
  - El formato del saldo es 999.999.999,99
  - Debería usarse una lista desplegable en lugar de un Check box.
- Estos detalles que son el resultado de las conversaciones con el PO y el equipo puede capturarlos en dos lugares:
  - Documentación interna de los equipos
  - Pruebas de aceptación automatizadas

# Pruebas de Aceptación de User Stories

## Front of Card

1B

As a student I want to purchase  
a parking pass so that I can  
drive to school

Priority: ~~High~~ Should  
Estimate: 4

## Back of Card

### Confirmations:

- ~~The student must pay the correct amount~~
- One pass for one month is issued at a time
- The student will not receive a pass if the payment isn't sufficient
- The person buying the pass must be a currently enrolled student.
- The student may only buy one pass per month.

# Pruebas de Aceptación de Historias de Usuario

---

Expresan detalles resultantes de la conversación

---

Complementan la User Story

---

Proceso de dos pasos:

1. Identificarlas al dorso de la US.
2. Diseñar las pruebas completas



# User Story: Tarjeta y Pruebas de Aceptación

## Buscar Destino por Dirección

**Como Conductor quiero buscar un destino a partir de una calle y altura para poder llegar al lugar deseado sin perderme.**

### Criterios de Aceptación:

- La altura de la calle es un número.
- La búsqueda no puede demorar más de 30 segundos.

### Pruebas de Usuario

- ❑ Probar buscar un destino en un país y ciudad existentes, de una calle existente y la altura existente (pasa).
- ❑ Probar buscar un destino en un país y ciudad existentes, de una calle inexistente (falla).
- ❑ Probar buscar un destino en un país y ciudad existentes, de una calle existente y la altura inexistente (falla).
- ❑ Probar buscar un destino en un país inexistente (falla).
- ❑ Probar buscar un destino en País existente, ciudad inexistente (falla).
- ❑ Probar buscar un destino en un país y ciudad existentes, de una calle existente y demora más de 30 segundos (falla).

## Ejemplo: para un software de un GPS

*Como Conductor quiero buscar un destino a partir de sus coordenadas para poder conocer el camino a recorrer para llegar al destino deseado.*

# Ejemplo:

Como *Conductor* quiero *buscar un destino a partir de sus coordenadas para poder conocer el camino a recorrer para llegar al destino deseado.*

**Criterios de Aceptación:** Las coordenadas se representan con tres números que indican longitud y tres números que indican latitud. Cada número representa los grados, minutos y segundos respectivamente. Además se debe indicar la orientación (norte, sur, este, oeste).

- Probar buscar un destino en un país y ciudad existentes, de dos coordenadas existentes (pasa).
- Probar buscar un destino en un país y ciudad existentes, de una coordenada inexistente (falla).
- Probar buscar un destino en un país y ciudad existentes, de dos coordenadas existentes sin indicar la orientación (falla).
- Probar ingresar coordenadas de latitud y longitud válidas (pasa).
- Probar ingresar coordenadas de latitud y longitud inválidas (falla).



# Ejemplo: User Stories / Casos de Prueba

Como compañía quiero pagar por una búsqueda de puestos con una tarjeta de crédito, así resuelvo mi necesidad en forma más eficiente.

Criterio de Aceptación:

- Se acepta Visa, MasterCard y American Express
- En compras mayores de \$100 se piden el número del dorso de la tarjeta

Probar con Visa (pasa)

Probar con MasterCard (pasa)

Probar con American Express (pasa)

Probar con Dinner's Club (falla)

Probar con números de tarjeta buenos

Probar con números de tarjeta malos

Probar con números de tarjeta faltantes

Probar con tarjetas vencidas

Probar con montos menores de \$100

Probar con montos mayores de \$100

# Definición de listo – Definition of Ready



# Definición de Hecho – Definition of Done



# \*INVEST Model

- **Independent** – calendarizables e implementables en cualquier orden
- **Negotiable** – el “qué” no el “cómo”
- **Valuable** – debe tener valor para el cliente
- **Estimatable** – para ayudar al cliente a armar un ranking basado en costos
- **Small** – deben ser “consumidas” en una iteración
- **Testable** – demostrar que fueron implementadas

\* “INVEST in Stories” – Bill Wake

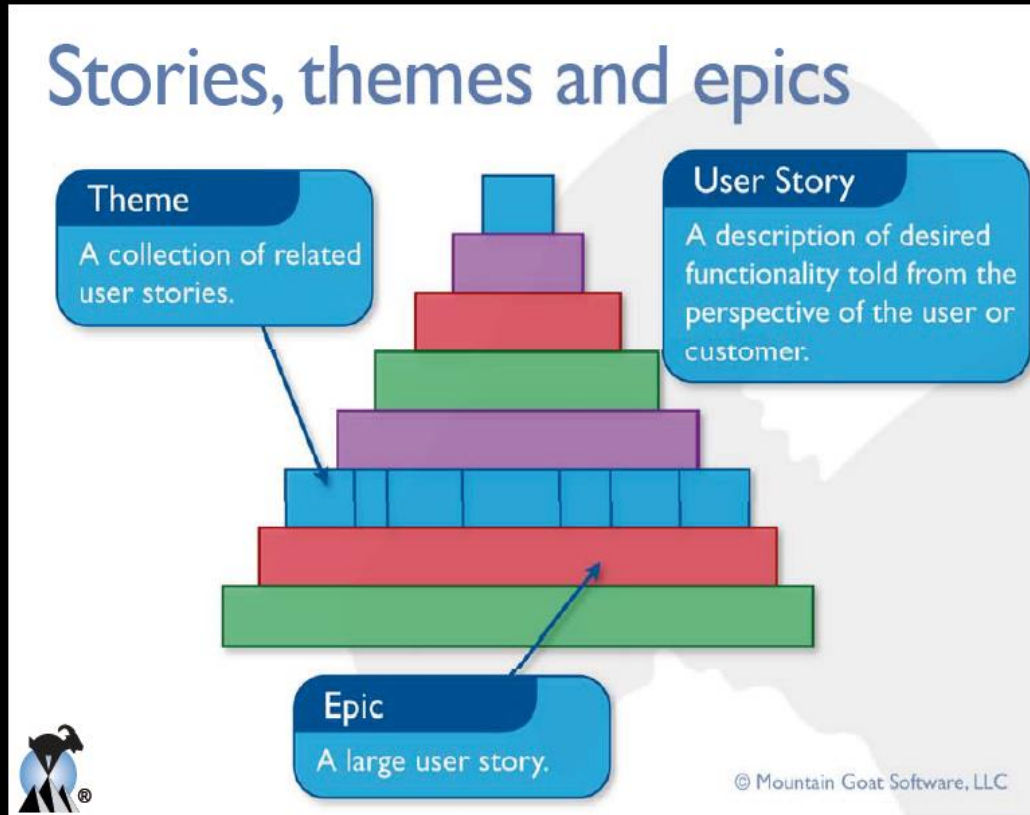
<http://xp123.com/xplor/xp0308/index.shtml>



# Algo más sobre las User Stories...

- No son especificaciones detalladas de requerimientos (como los casos de uso)
- Son expresiones de intención, “es necesario que haga algo como esto...”
- No están detallados al principio del proyecto, elaborados evitando especificaciones anticipadas, demoras en el desarrollo, inventario de requerimientos y una definición limitada de la solución.
- Necesita poco o nulo mantenimiento y puede descartarse después de la implementación.
- Junto con el código, sirven de entrada a la documentación que se desarrolla incrementalmente después.

# Diferentes niveles de abstracción



Idea

Problema

Necesidad

Cambios en el Negocio

Ideas / Cambios en  
el Software

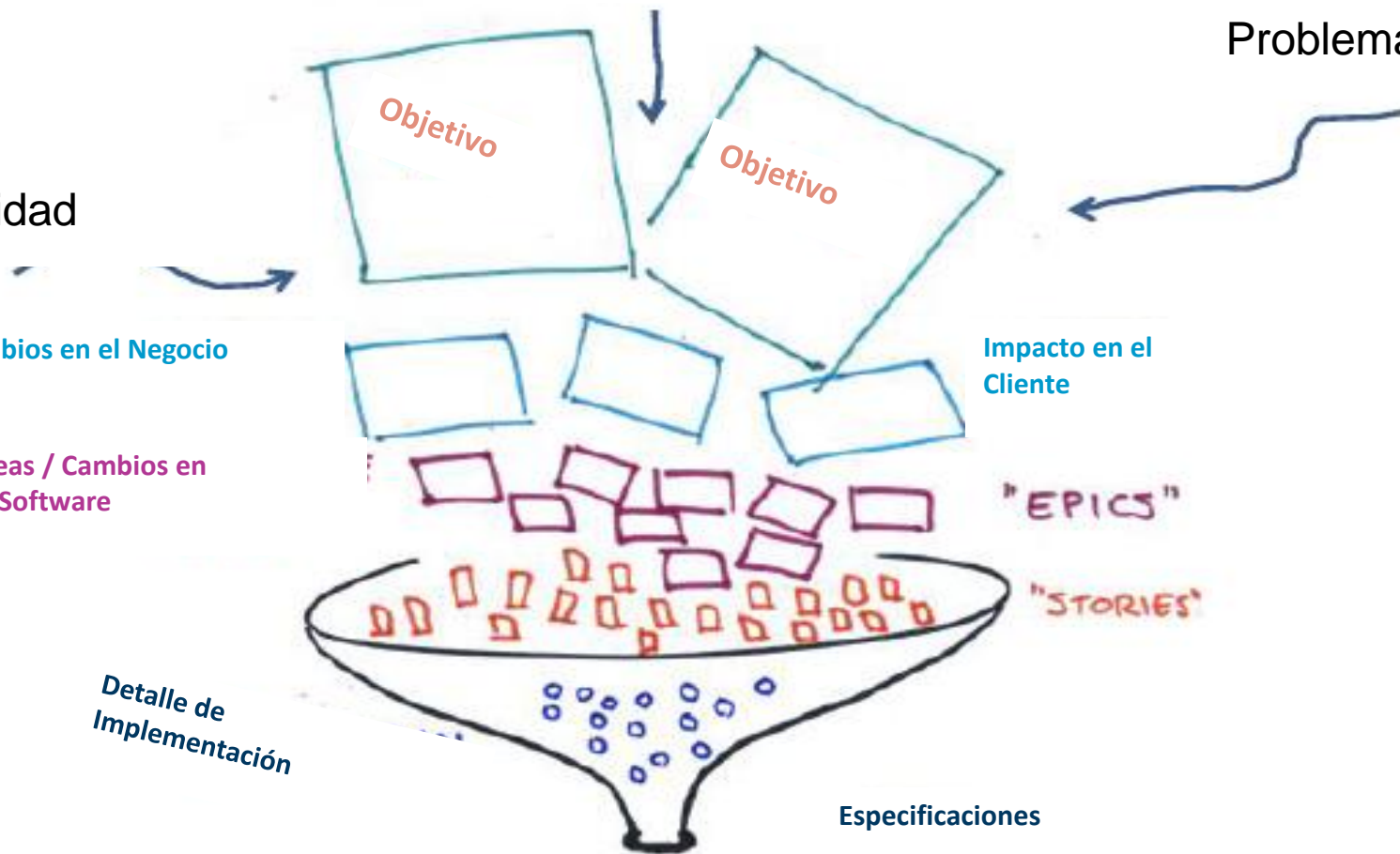
Impacto en el  
Cliente

"EPICS"

"STORIES"

Detalle de  
Implementación

Especificaciones



Qué no son las user stories.....

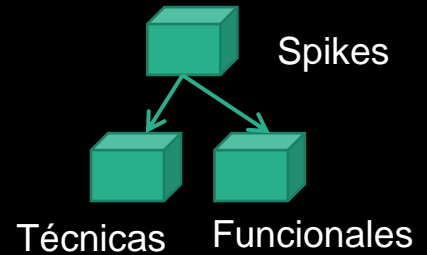




# Spikes

- Tipo especial de historia, utilizado para quitar riesgo e incertidumbre de una User Story u otra faceta del proyecto.
- Se clasifican en : técnicas y funcionales.
- Pueden utilizarse para:
  - Inversión básica para familiarizar al equipo con una nueva tecnología o dominio.
  - Analizar un comportamiento de una historia compleja y poder así dividirla en piezas manejables.
  - Ganar confianza frente a riesgos tecnológicos, investigando o prototipando para ganar confianza.
  - Frente a riesgos funcionales, donde no está claro como el sistema debe resolverla interacción con el usuario para alcanzar el beneficio esperado.

# Spikes



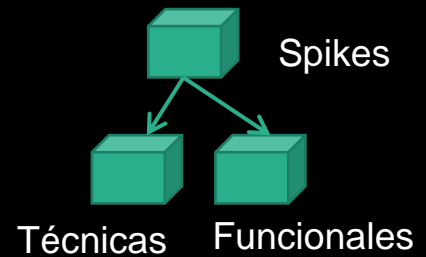
## Técnicas

- Utilizadas para investigar enfoques técnicos en el dominio de la solución.
  - Evaluar performance potencial
  - Decisión hacer o comprar
  - Evaluar la implementación de cierta tecnología.
- Cualquier situación en la que el equipo necesite una comprensión más fiable antes de comprometerse a una nueva funcionalidad en un tiempo fijo.

## Funcionales

- Utilizadas cuando hay cierta incertidumbre respecto de cómo el usuario interactuará con el sistema.
- Usualmente son mejor evaluadas con prototipos para obtener realimentación de los usuarios o involucrados.

# Spikes



- Algunas User Stories requieren de ambos tipos de spikes. Por ejemplo:
  - Como un cliente, quiero ver mi uso diario de energía en un histograma, para poder comprender rápidamente mi consumo de energía pasado, presente y proyectado.
- En este caso un equipo puede crear dos spikes:
  - Spike Técnico:
    - Investigar cuanto tiempo requiere actualizar un display de un cliente al uso actual, determinando requerimientos de comunicación, ancho de banda y si los datos se actualizan en formato push o pull.
  - Spike Funcional:
    - Crear un prototipo de histograma en el portal web y obtener la retroalimentación de algunos usuarios respecto del tamaño, el estilo de la presentación y los atributos gráficos.

# Lineamientos para Spikes

Estimables, demostrables, y aceptables

La excepción, no la regla

- Toda historia tiene incertidumbre y riesgos.
- El objetivo del equipo es aprender a aceptar y resolver cierta incertidumbre en cada iteración.
- Los spikes deben dejarse para incógnitas mas críticas y grandes.
- Utilizar spikes como última opción.

Implementar la spike en una iteración separada de las historias resultantes

- Salvo que el spike sea pequeño y sencillo y sea probable encontrar una solución rápida en cuyo caso, spike e historia pueden incluirse en la misma iteración.

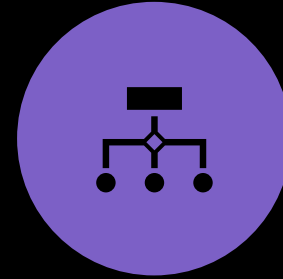
# Algunas cosas para dejar en claro



DIFERIR EL ANÁLISIS  
DETALLADO TAN TARDE  
COMO SEA POSIBLE, LO QUE  
ES JUSTO ANTES DE QUE EL  
TRABAJO COMIENCE.



HASTA ENTONCES, SE  
CAPTURAN  
REQUERIMIENTOS EN LA  
FORMA DE "USER STORIES".



LAS USER STORIES NO SON  
REQUERIMIENTOS DE  
SOFTWARE, NO NECESITAN  
SER DESCRIPCIONES  
EXHAUSTIVAS DE LA  
FUNCIONALIDAD DEL  
SISTEMA.

# Tips para que las user stories sean útiles para el equipo



Un paso a la vez (evitar la palabra “Y”)



Usar palabras claras en los criterios de aceptación



No olvides la parte invisible: la conversación



Las user stories se escriben desde la perspectiva del usuario



No forzar todo para escribirlo como user stories

# Material Bibliográfico de Referencia

- Libro:
  - Cohn, Mike - USER STORIES APPLIED – Editorial Addison Wesley 2004- Capítulos 1, 2 y 6
- Papers
  - Dean Leffingwell and Pete Behrens – A user story primer (2009)
- Link
  - <http://www.mountaingoatsoftware.com/>