

# Requisitos

# Dos tipos

- ▶ Funcionales
  - ▶ qué es lo que el producto o sistema debe ser capaz de hacer (features)
- ▶ No Funcionales
  - ▶ restricciones adicionales mas orientadas al cómo que al qué
  - ▶ performance, usabilidad, mantenibilidad, plataforma

# Documento de Requisitos

## 1. Introduction

- 1.1 Purpose
- 1.2 Document conventions
- 1.3 Intended audience
- 1.4 Additional information
- 1.5 Contact information/SRS team members
- 1.6 References

## 2. Overall Description

- 2.1 Product perspective
- 2.2 Product functions
- 2.3 User classes and characteristics
- 2.4 Operating environment
- 2.5 User environment
- 2.6 Design/implementation constraints
- 2.7 Assumptions and dependencies

## 3. External Interface Requirements

- 3.1 User interfaces
- 3.2 Hardware interfaces
- 3.3 Software interfaces
- 3.4 Communication protocols and interfaces

## 4. System Features

- 4.1 System feature A
  - 4.1.1 Description and priority
  - 4.1.2 Action/result
  - 4.1.3 Functional requirements
- 4.2 System feature B

## 5. Other Nonfunctional Requirements

- 5.1 Performance requirements
- 5.2 Safety requirements
- 5.3 Security requirements
- 5.4 Software quality attributes
- 5.5 Project documentation
- 5.6 User documentation

## 6. Other Requirements

- Appendix A: Terminology/Glossary/Definitions list
- Appendix B: To be determined

# El problema de las especificaciones

*Customer:* Hello, I'd like to order a cake.

*Employee:* Sure, what would you like written on it?

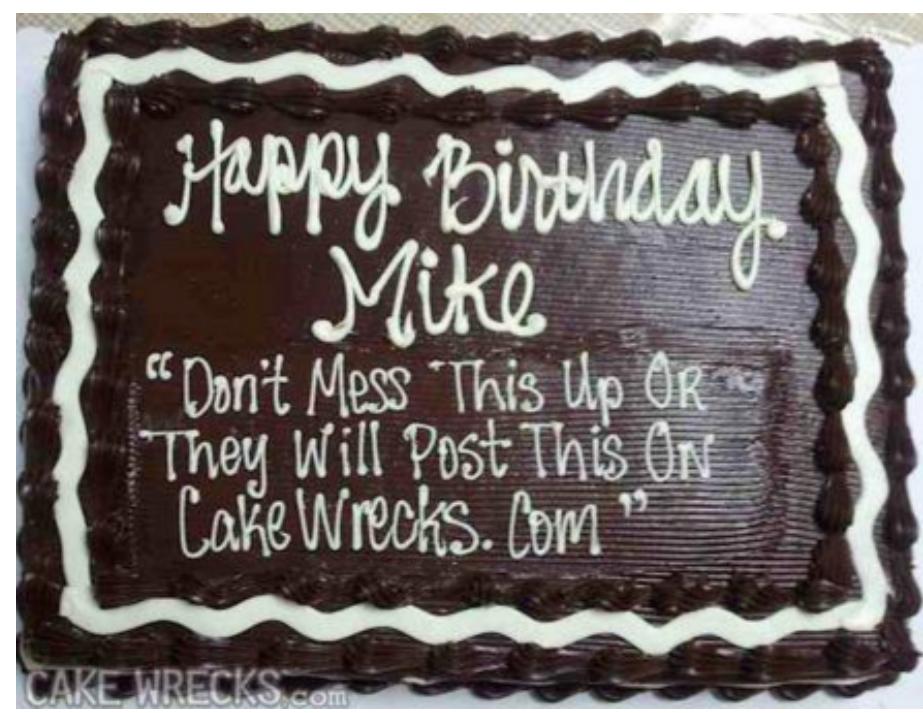
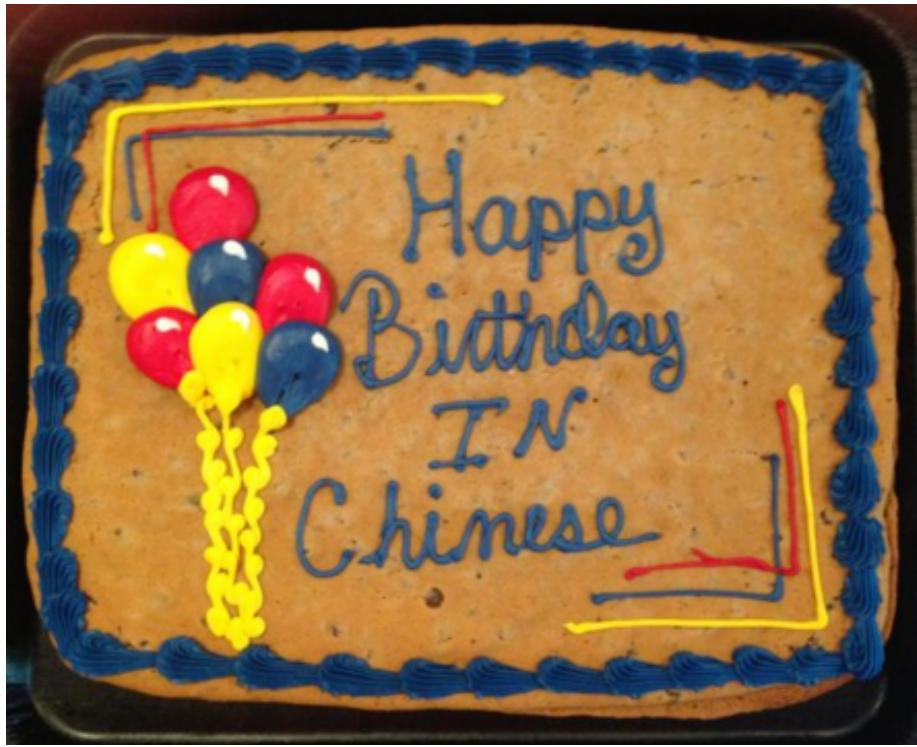
*Customer:* Could you write "So long, Alicia" in purple?

*Employee:* Sure.

*Customer:* And put stars around it?

*Employee:* No problem. I've written this up, and will hand it to my cake decorator right away. We'll have it for you in the morning.





# Levantamiento de Requisitos Funcionales

- ▶ Idea - para cada tipo de usuario capturar la forma en que se espera que utilice el producto o sistema
- ▶ Diseño centrado en el usuario
- ▶ Hay dos metodologías ampliamente usadas
  - ▶ Casos de Uso (Use Cases)
  - ▶ Relatos de Usuario (User Stories)



2-10

© 1996 Farcus Cartoonist, by Universal Press Syndicate

WAISGLASS/COULTHART



74777-33016@compuserve.com

**"We've located the problem ...  
it's a defective user."**



www.dilbert.com

11/1/01 © 2001 United Feature Syndicate, Inc.

# Relatos de Usuario

- ▶ documentos compartidos NO IMPLICA entendimiento compartido
- ▶ el objetivo de los relatos de usuario es el entendimiento compartido
- ▶ usar todos los medios para recordar la conversación

# Un relato de usuario es como una foto de vacaciones



la foto tiene  
detalles  
adicionales  
que el relato  
no participó

# Lecciones Importantes

- ▶ los relatos de usuario NO SON un nuevo formato de requerimientos escritos sino un mecanismo para adquirir un entendimiento compartido
- ▶ los relatos representan discusiones sobre solución de problemas de nuestros usuarios y clientes y que llevarán a acuerdos sobre qué construir
- ▶ el objetivo NO ES construir más software más rápido sino **maximizar el outcome** y el impacto

# User Stories (Relatos de Usuario)

- Muy usada en procesos ágiles (Scrum)
- Idea es usar una “conversación” para capturar los requerimientos de usuario
- Puede ser interpretado como una promesa de conversación con el usuario
- Inicialmente se usaron tarjetas de 3x5 pulgadas (muy populares en USA)

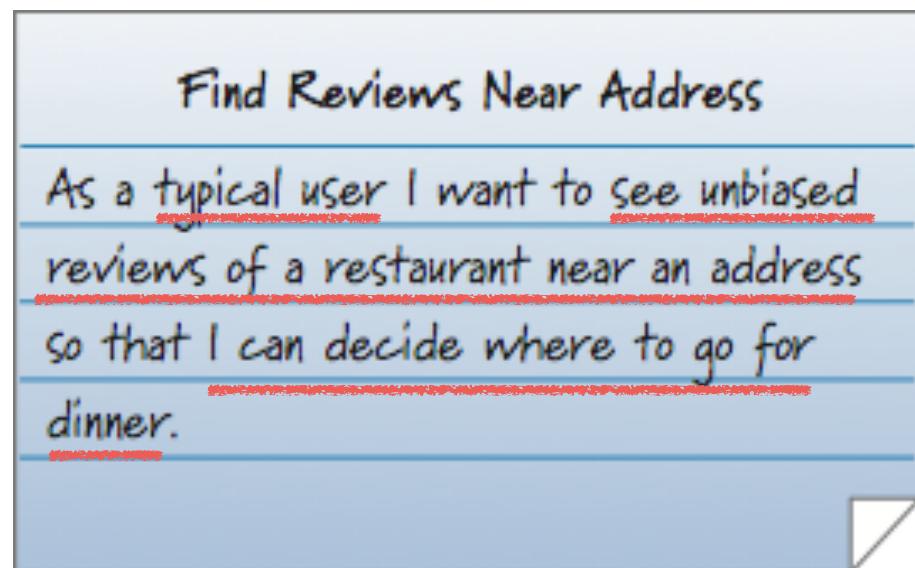
# Fundamentos tras la idea

- ▶ requerimientos a través de colaboración
- ▶ requerimientos escritos y pasados en un documento tienen todas las decisiones tomadas
- ▶ con tiempos de entrega cortos es imposible especificar cada detalle de antemano
- ▶ responsabilidad de escribir correctamente los requerimientos se distribuye

# US como Promesa de Conversación

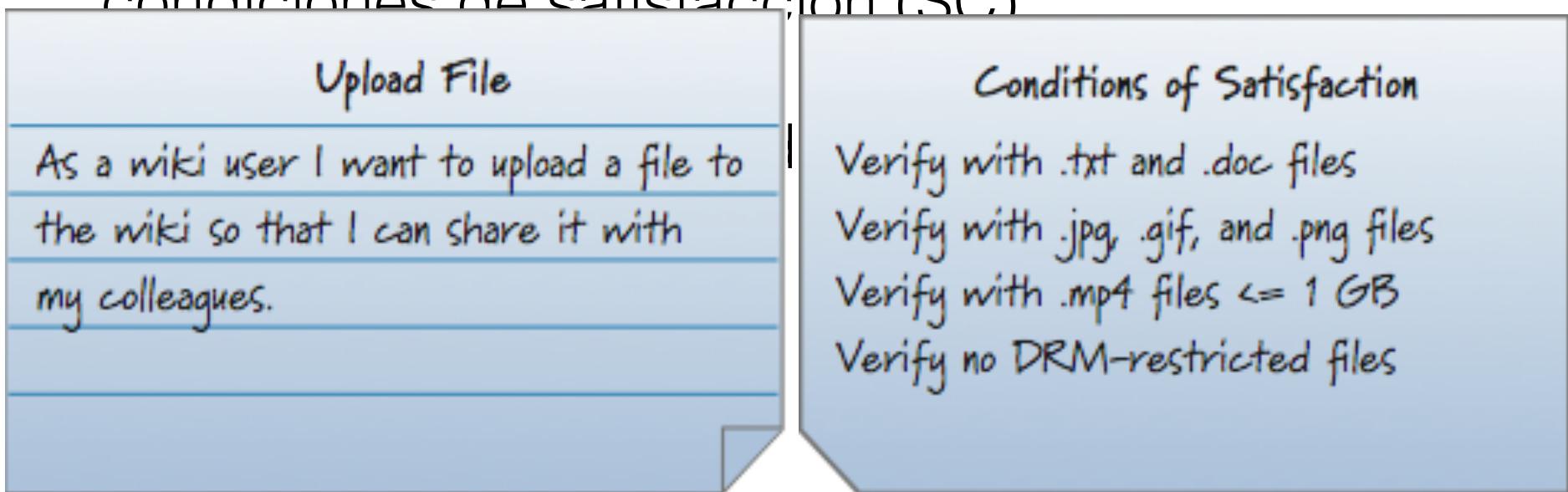
- ▶ Las tarjetas son solo un punto de partida en el levantamiento de lo que se requiere
- ▶ User Story constituye una promesa de llevar a cabo una conversación (pueden ser varias) con los interesados
- ▶ resultados de la conversación podrían producir algunos documentos adicionales (detalles de la interfaz)

# Card, Conversation, Confirmation



# Confirmación

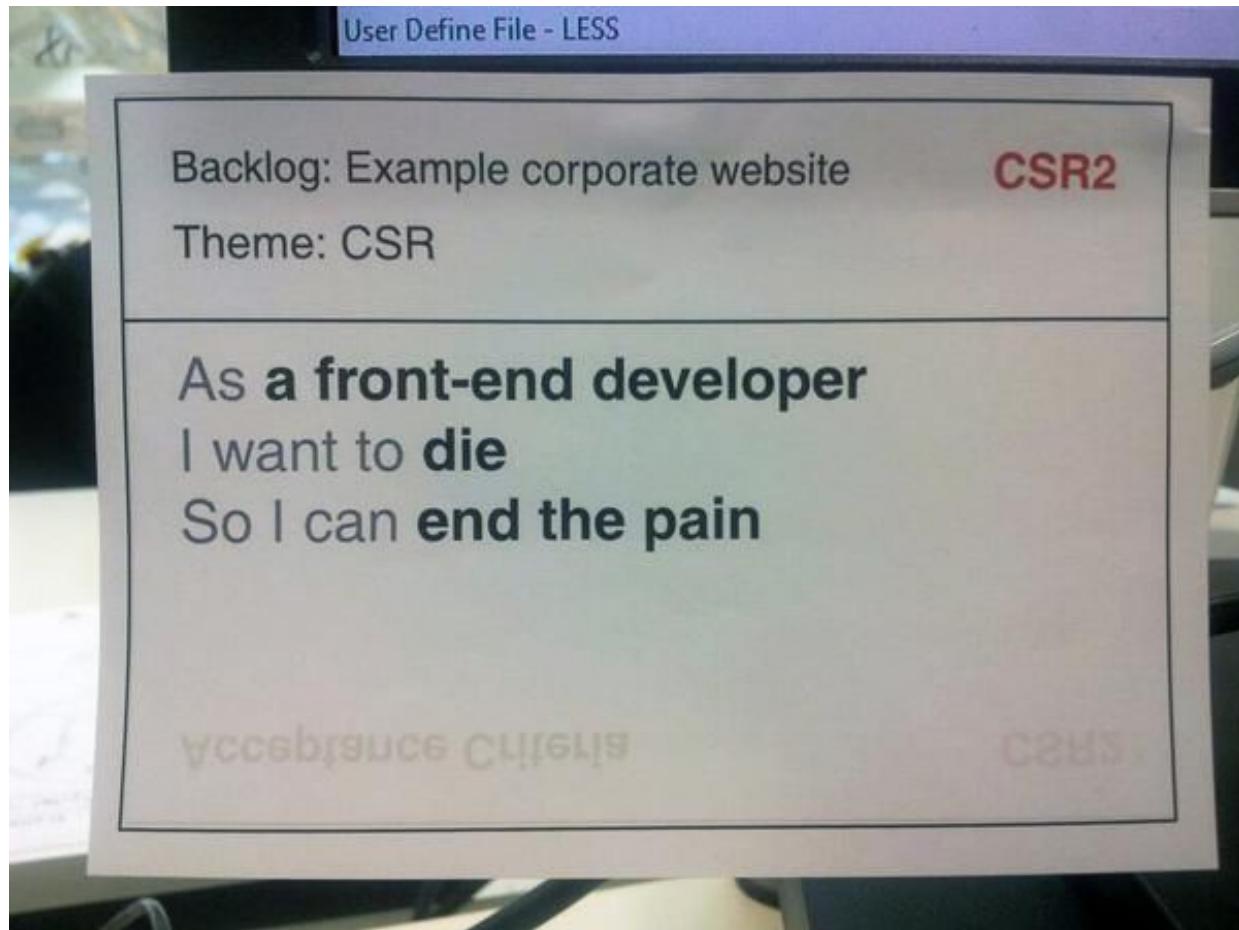
- ▶ En el reverso de la tarjeta se ponen las condiciones de satisfacción (SC)



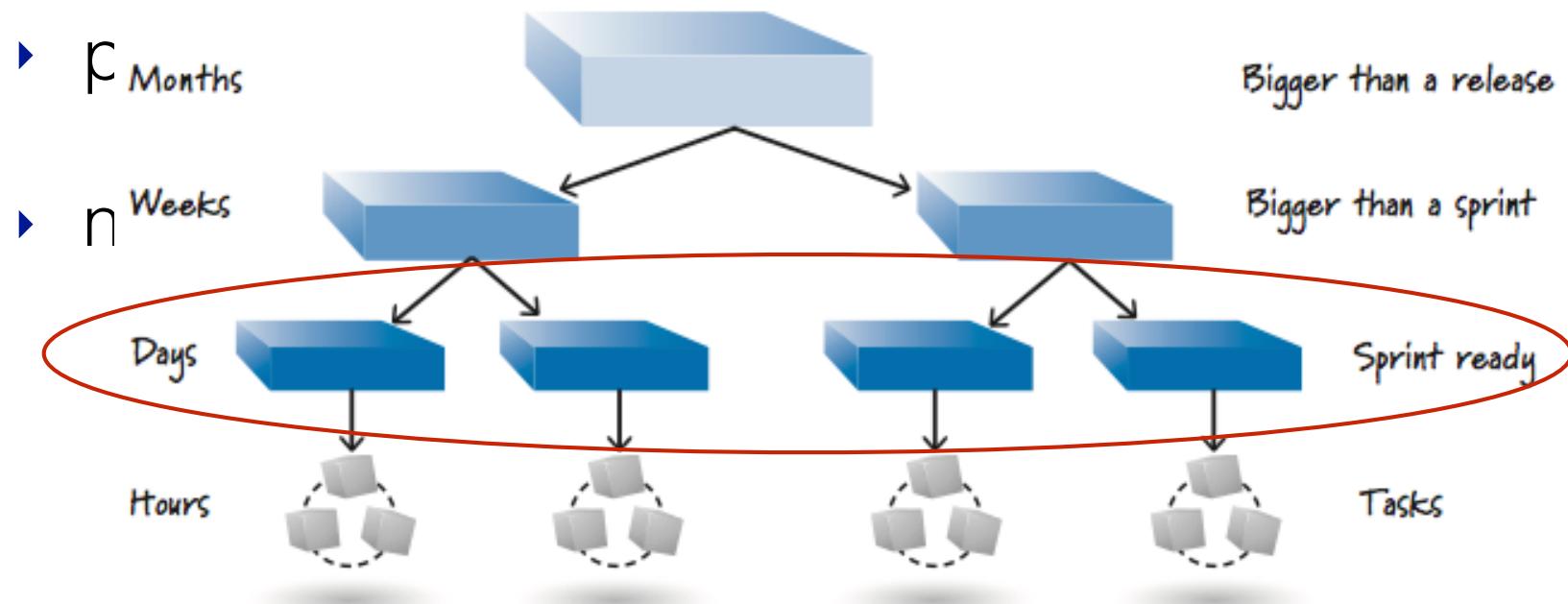
As a customer, I want to sign-in using my Facebook account, so that I can easily access the service.

Daddy can you read me the story about princesses instead?



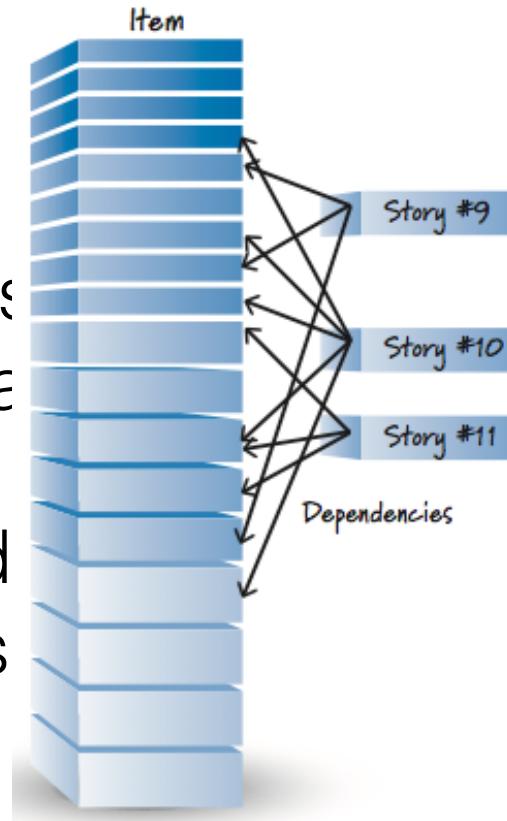


# Nivel de Detalle



# Ojalá sean independientes

- ▶ Se hace muy difícil priorizar si dependencias entre las tareas
- ▶ tratar de escribir las tareas de minimicen las dependencias



# Deben tener un valor para el cliente

- ▶ si hay una US que no es de valor para nadie no se debe hacer
- ▶ cuidado con cosas que son de valor para desarrolladores (valor técnico)
  - ▶ migrar código a última versión de Rails
- ▶ en esos casos es posible a veces reformularlo de forma que tenga un claro valor para el cliente
  - ▶ migrar código para mitigar riesgo ....

# Estimable

- ▶ Si el equipo no es capaz de estimar el esfuerzo para desarrollar un US puede ser que
  - ▶ US sea muy grande
  - ▶ US sea ambiguo

# ¿Como escribir las historias ?

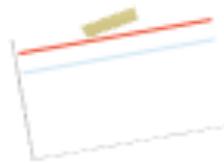
- ▶ incorporar a los usuarios al equipo en esta etapa
- ▶ workshop - brainstorming con participación de usuarios (al menos product owner)
- ▶ puede ser de unas horas hasta unos pocos días
- ▶ la idea es generar un set inicial y no un set completo
- ▶ comenzar identificando roles de las historias para poder escribir “As a <user role>, I want to ...”
- ▶ top-down o bottom-up

# Algunos vicios

- ▶ tratar de evitar el genérico "as a user" (ser más específico)
- ▶ US es muy pequeño
  - ▶ resultados deben guardarse como archivo XML
  - ▶ resultados deben guardarse como HTML
- ▶ US con interdependencias (difícil planeación)
- ▶ Goldplating (haciendo más de lo necesario)
  - ▶ además de tabs se pueden arrastrar
- ▶ Demasiado detalle (por eso se usan tarjetas pequeñas)
- ▶ Detalles de interfaz de usuario demasiado pronto
- ▶ traspasar requisitos formales a relatos de usuario

# Relatos vs Documentos de Requisitos

User stories



Lean, accurate, just-in-time

Specifications &  
requirement docs



Heavy, inaccurate, out-of-date

# El propósito no es el software !!

- ▶ el software es el *output* del proceso
- ▶ lo que nos interesa realmente es el *outcome*
- ▶ el outcome es el resultado para los usuarios
  - ▶ podrán lograr sus metas con mayor facilidad ?
  - ▶ hemos mejorado en algo sus vidas ?
  - ▶ hay un impacto ? en qué usuarios ?
- ▶ un buen relato no es solo acerca de qué sino de quien y por qué

# Maximizar Outcome

- ▶ Siempre habrá más que construir que el tiempo y los recursos disponibles
- ▶ Es un error intentar maximizar el output (más features, producto más completo)
- ▶ Debemos apuntar a maximizar el outcome y el impacto ojalá minimizando el output
- ▶ Implica escuchar cuidadosamente a la gente cuyo problema tratamos de resolver

Es posible dejar a la gente correcta muy contenta  
incluso construyendo una pequeña  
fracción de lo que estaba inicialmente planeado

# Idea tras los relatos de Usuario

- el problema de requisitos es realmente un problema de comunicación ...
- ... entre quienes van a utilizar el software y quienes lo tienen que construir

# Balancear el mundo de los negocios con el mundo técnico

- ▶ Si el lado del negocio es muy dominante ...
  - ▶ funcionalidad y fechas quedan determinadas sin considerar si desarrolladores entienden los requisitos
- ▶ Si el lado técnico es muy dominante
  - ▶ mucha jerga técnica, desarrolladores pierden oportunidades de aprender al escuchar de los interesados

# Planificación es Imperfecta

- ▶ Es muy difícil si no imposible predecir tiempos de desarrollo en forma precisa
  - ▶ a medida que usuarios ven el software salen nuevas ideas
  - ▶ desarrolladores son malos estimadores
- ▶ Esto hace difícil predecir que es lo que finalmente será entregado

# Las 3 C's de los Relatos

- ▶ Card
  - ▶ los relatos son escritos en tarjetas (pueden ser anotados con estimaciones, notas, etc)
- ▶ Conversation
  - ▶ los detalles tras el relato salen en conversaciones con el product owner
- ▶ Confirmation
  - ▶ test de aceptación conforma que el relato estaba en lo correcto

# Ejemplo de Relatos

As a user, I want to  
reserve a hotel room.

As a vacation traveler,  
I want to see photos of  
the hotels.

As a user, I want to  
cancel a reservation.

As a frequent flyer, I  
want to rebook a past trip  
so that I save time  
booking trips I take often.

# Y los detalles



As a user, I want to  
cancel a reservation.

- ▶ se le debe hacer una devolución completa ?
  - ▶ si pagó con tarjeta ?
- ▶ hasta cuando se acepta la cancelación
  - ▶ hay diferencias entre hoteles ?
  - ▶ hay diferencias entre usuarios (viajeros frecuentes) ?
- ▶ se le debe dar una conformación al usuario ?

# Condiciones de Satisfacción

- ▶ Pueden user usadas para agregar detalles al relato
  - ▶ verificar que un miembro premium pueda cancelar el mismo día sin cargo
  - ▶ verificar que miembros no premium deben pagar un 10% si cancelan el mismo día
  - ▶ verificar que el email de conformación se envía
  - ▶ verificar que el hotel sea notificado de la cancelación

# A veces el detalle aparece en sub-relatos

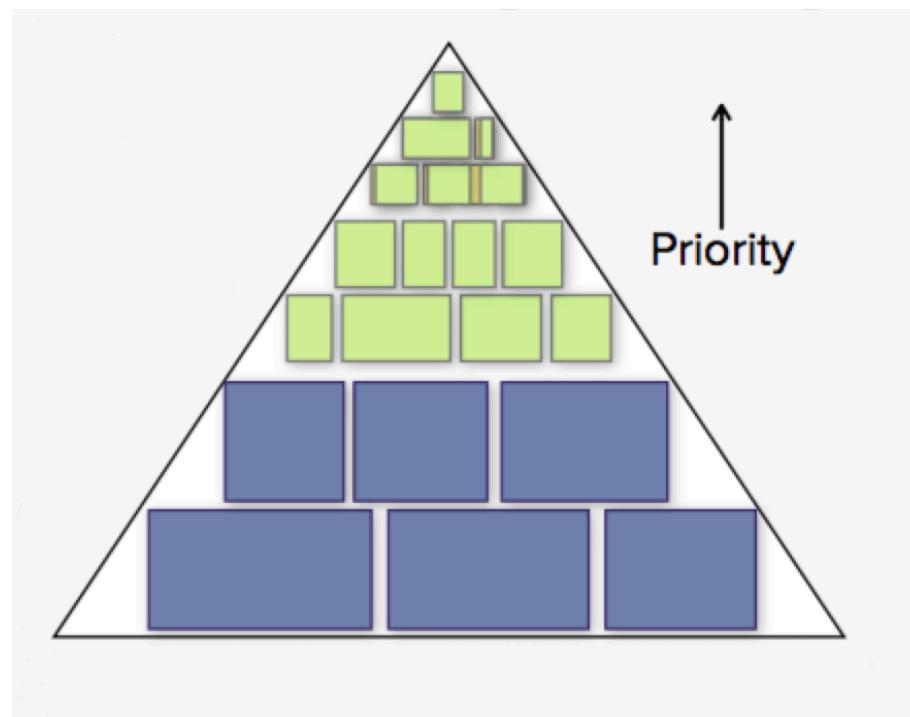
As a user, I can cancel a reservation.

As a premium site member, I can cancel a reservation up to the last minute.

As a non-premium member, I can cancel up to 24 hours in advance.

As a site visitor, I am emailed a confirmation of any cancelled reservation.

# El backlog del producto



# Epicas, Temas y Relatos

- ▶ La idea es ver los relatos de usuario en un contexto mayor que una sola pequeña parte
- ▶ Layout en una pared, o en un tablero
- ▶ Agrupar en *épicas* y *temas*
- ▶ Imaginar al producto como si existiese y describir una historia completa
  - ▶ comprar un ticket de avión a San Francisco
  - ▶ tomar mis cursos del semestre 2'2016 (Banner)
- ▶ Las tareas necesarias para llevar a cabo ese relato se ponen de izquierda a derecha y de arriba a abajo en tarjetas

# Epicas vs Temas

- Una épica corresponde a un objetivo mayor que requiere ser separado en varios relatos que deben hacerse en secuencia
- Por lo general los relatos que son parte de una épica no tienen mucho valor en sí mismos
- Un tema puede ser cualquier agrupación de relatos (variaciones de un mismo objetivo, etc)
- Los relatos de un tema tienen valor en forma independiente

# Para ponerlo en términos muy sencillos

- ▶ Tema : una colección de relatos relacionados de algún modo
- ▶ Epica : un relato grande que puede ser dividido en varios pasos

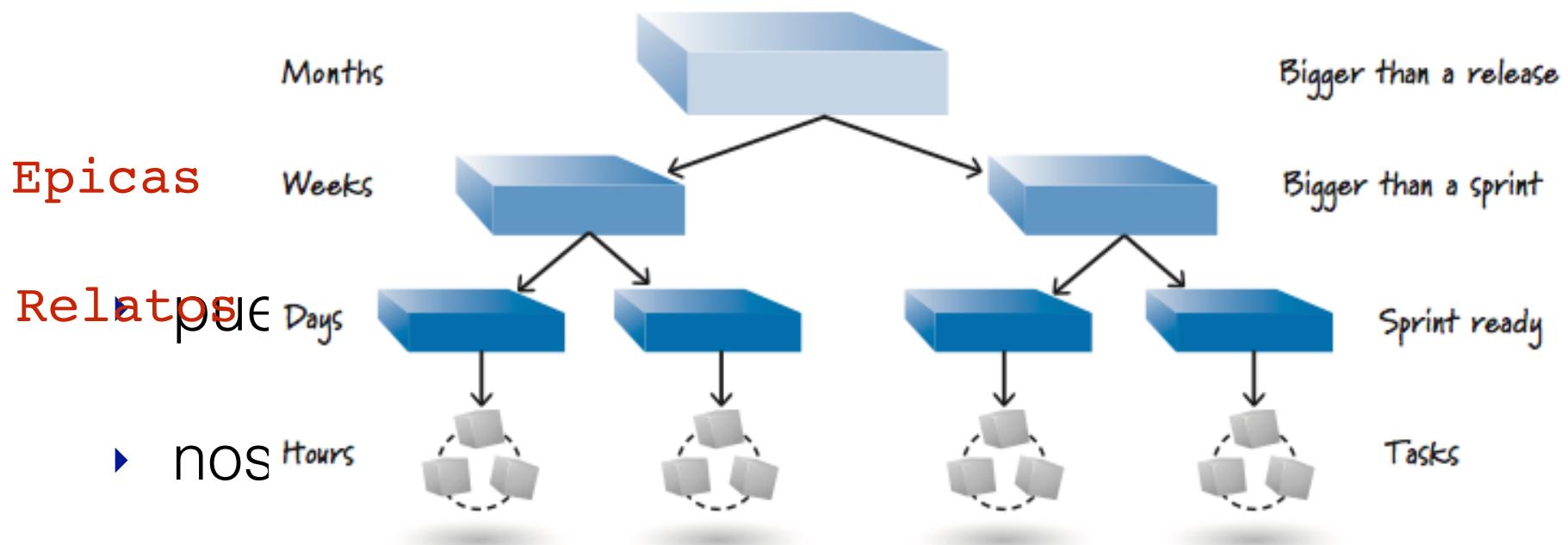
# Ejemplo

- ▶ Epica : Realizar una transferencia bancaria
  - ▶ autenticarse como cliente del banco
  - ▶ especificar el destino de la transferencia
  - ▶ especificar la cuenta de origen
  - ▶ realizar la transferencia propiamente tal

# Posibles temas

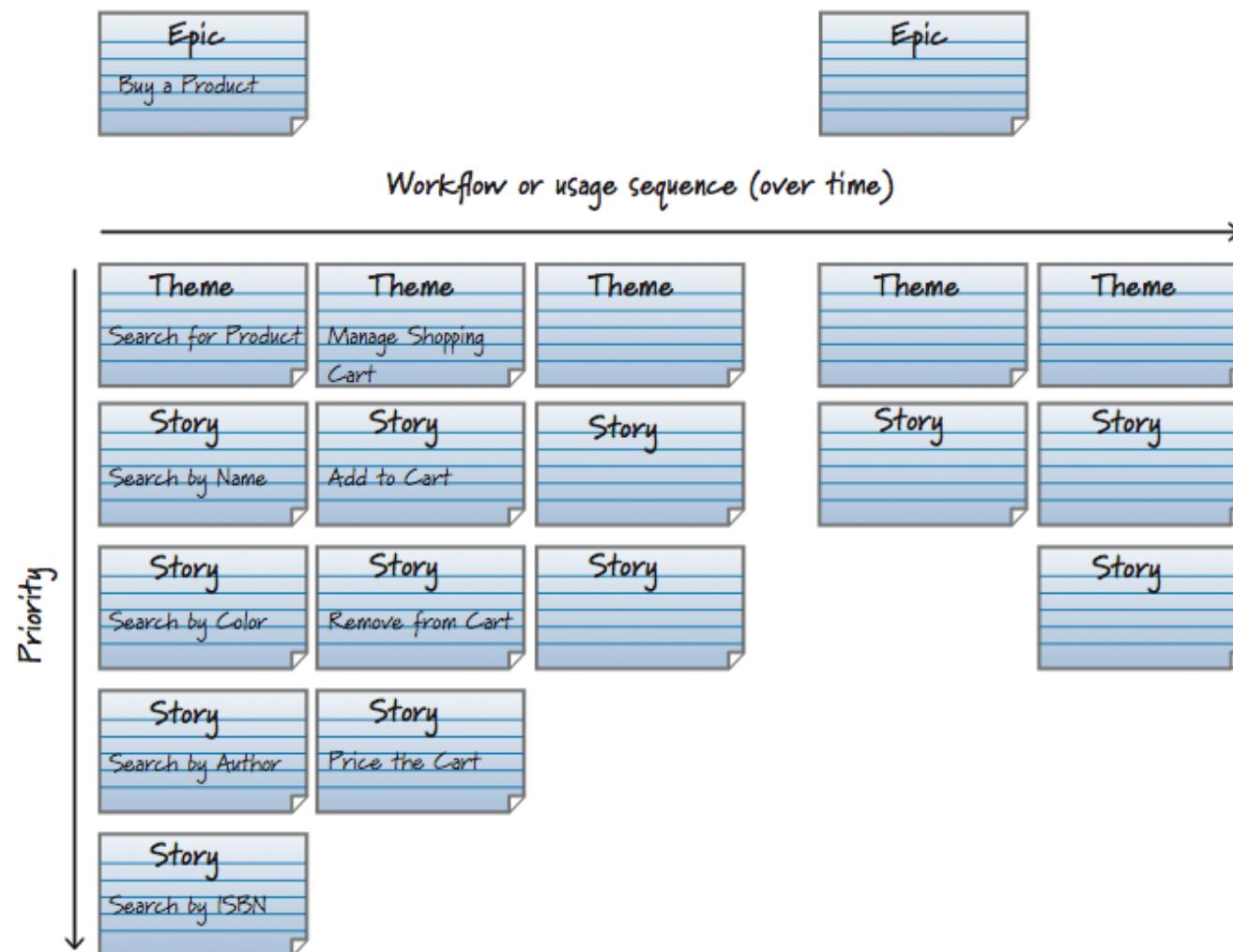
- ▶ Especificar el destino
  - ▶ ingresar datos de nuevo receptor
  - ▶ indicar datos de un receptor ya usado antes
  - ▶ ingresar mensaje asociado
- ▶ Realizar transferencia
  - ▶ ingresar pinpass
  - ▶ ingresar coordenadas

# Una épica corresponde a un relato demasiado grande



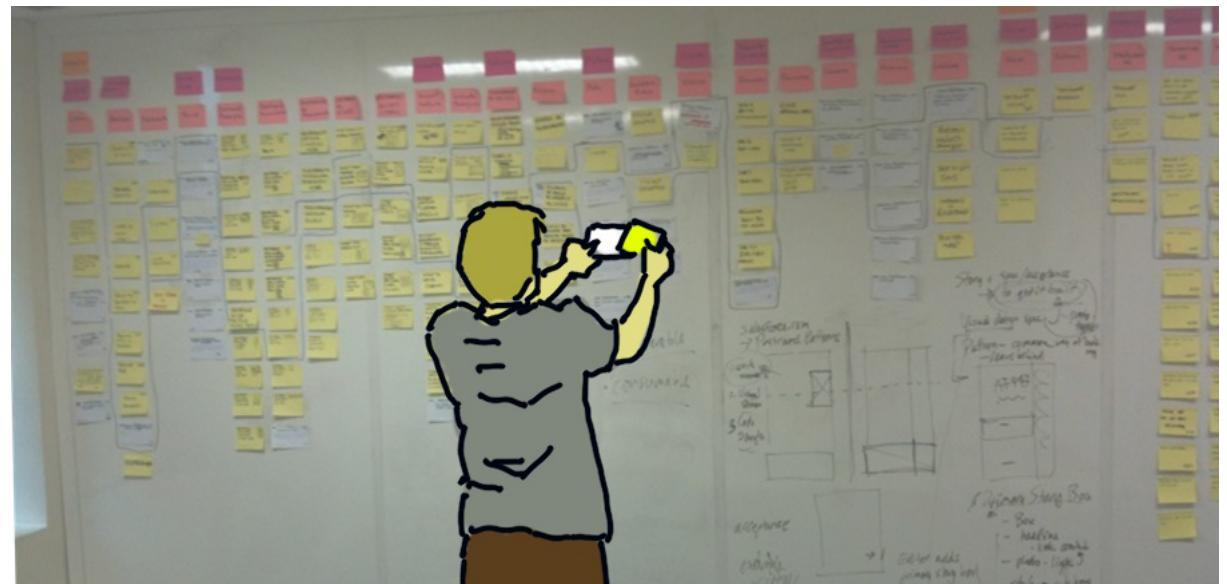
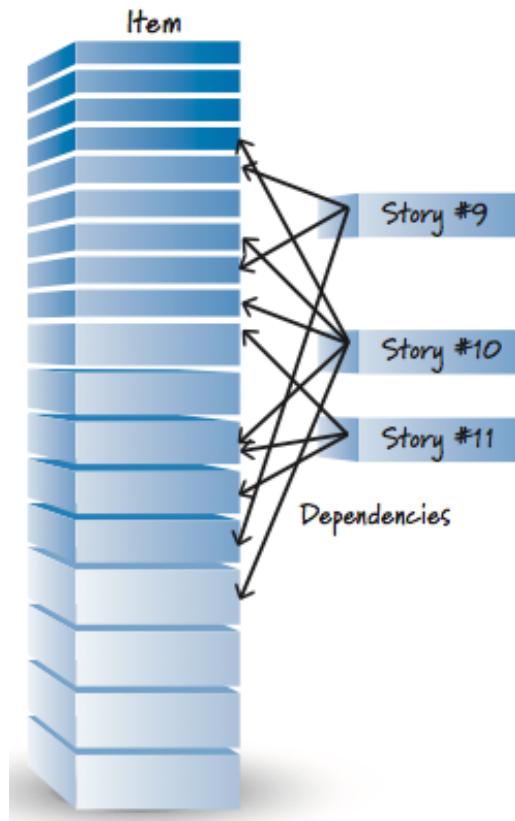
# Story Mapping

- ▶ Una épica es un relato mayor compuestos de muchos relatos pequeños
- ▶ Relatos pequeños pueden agruparse en temas



# Visualización concreta de lo que hay que hacer

- ▶ Se hace muy difícil priorizar si hay muchas dependencias entre las tareas
- ▶ tratar de escribir las tareas de modo que se minimicen las dependencias



No importa mucho el término, al final siempre son relatos (stories)

