Raccolta dei Requisiti con le User Stories

Corso di Ingegneria del Software
Anno Accademico 2012/2013

Introduzione (1/4)

- **Le metodologie agili si basano sull'osservazione che né il cliente né lo sviluppatore hanno una conoscenza completa del sistema nelle prime fasi del processo di sviluppo. Essi imparano man mano che il sistema evolve**
- **E' fondamentale adottare delle pratiche che permettano ad entrambi di poter sfruttare la propria crescente competenza durante il processo di sviluppo**

Introduzione (2/4)

- ***The hardest part of the software task is arriving at a complete and consistent specification, and much of the essence of building a program is in fact the debugging of the specification***
  - **(Fred Brooks, 1987)**

Introduzione (3/4)

- In un processo tradizionale, i requisiti devono essere definiti dal cliente in modo molto preciso nella prima fase di analisi
- Apportare delle modifiche man mano che il processo di sviluppo avanza diventa sempre più difficile e oneroso
- I requisiti sono documentati in modo rigoroso e formale

Introduzione (4/4)

- **Le MA invece lasciano al cliente la libertà di decidere in qualunque momento quali funzionalità realizzare e di richiedere modifiche delle funzionalità esistenti**
- **La parola *Agile* si riferisce alla possibilità di far fronte al continuo cambiamento dei requisiti**
- **Identificare i requisiti in modo rigoroso e formale è un processo lungo e costoso, e risulta spesso poco utile perché i requisiti possono cambiare molto velocemente**

Il contesto determina l'approccio

- **Progetti stabili con implicazioni *safety-critical* possono essere realizzati meglio seguendo un approccio tradizionale, utilizzando una specifica dei requisiti ben documentata**
- **Progetti soggetti a una gran quantità di cambiamenti possono essere affrontati meglio utilizzando l'approccio *Agile***

Requisiti Agili (1/3)

- **Sono espressi ad alto livello, con frasi brevi che contengono solo le informazioni più importanti**
- **Nessuna forma di documentazione cerimoniosa, formale e *voluminosa***

Requisiti Agili (2/3)

- **Massima interazione tra gli *stakeholder***
  - **Permette agli sviluppatori di capire i dettagli di cosa il cliente vuole veramente**
  - **Nel caso migliore: *real customer involvement* (il cliente è disponibile durante tutto il processo di sviluppo)**
  - **In alternativa: si una un *proxy* del cliente *on-site*, cioè una persona che conosce il problema, ed agisce per conto del cliente**

Requisiti Agili (3/3)

- **Rilasci frequenti del software**
  - **Solo se il cliente può usare una versione funzionante del sistema può fornire *feedback* e specificare meglio i requisiti**
  - **Lo sviluppo si basa su iterazioni di una/due settimane**
  - **Almeno ogni tre mesi viene rilasciata una versione funzionante del software**

Esprimere i Requisiti Agili

- **Feature o Caratteristica - MMF (Scrum e FDD)**
  - **Minimum Marketable Feature**
  - **E' una breve frase che descrive cosa il cliente vuole (funzionalità o requisito non funzionale)**
  - **La chiarezza è essenziale**
    - **Esempio: *gli oggetti all'asta possono essere aggiunti, modificati e cancellati***
- **User Story (XP)**
  - **Breve descrizione dell'interazione di un utente o sistema esterno col sistema**
- **Entrambe guidano lo sviluppo!**

User Stories (1/3)

- **Una User Story, o Storia, è una breve descrizione del comportamento del sistema, dal punto di vista dell'utente (brevi Casi d'Uso)**
- **E' scritta in linguaggio naturale**
- **Per ogni funzionalità principale del sistema, deve essere scritta almeno una storia**
- **Ogni Storia deve fornire valore al cliente**
- **Sono scritte dal cliente, in genere con l'assistenza degli sviluppatori**

User Stories (2/3)

- **Sono scritte su *index card* (schede):**
  - **schede in formato A6 (10x15)**
- **Dovrebbero essere indipendenti le une dalle altre, per favorire lo scheduling**
- **Sono stimate dagli sviluppatori, cioè gli sviluppatori devono essere in grado di prevedere quanto tempo impiegheranno per realizzarla**

User Stories (3/3)

- **Appartengono al cliente, che ne definisce la priorità**
- **Ogni storia deve essere testabile, questo evita che si scrivano storie ambigue**
- **Non devono essere esaustive, ma sono *occasioni per un incontro col cliente***
- **In pratica, devono essere spiegate dal cliente agli sviluppatori, che possono rivolgersi al cliente anche in seguito, per ulteriori chiarimenti**

User Stories vs Use Cases

- **Entrambi descrivono funzionalità del sistema**
- **User Stories NON è un altro modo di chiamare gli Use Cases**
- **Le User Stories sono dei documenti molto più semplici dei casi d'uso, e non sono esaustivi**
- **Le storie sono scritte dal cliente**
- **I casi d'uso sono scritti dagli analisti, sentendo le esigenze del cliente**
- **I casi d'uso riportano una descrizione più completa del requisito relativo**

Raccogliere le Storie (1/2)

- **In un processo Agile, i requisiti si raccolgono durante tutta la durata del progetto**
- **Nella prima fase i requisiti sono raccolti attraverso una serie di riunioni, alle quali partecipano sia il cliente che gli sviluppatori**

Raccogliere le Storie (2/2)

- **Due approcci**
  - **Approccio *Goal-oriented***
    - **Punto di partenza: obiettivo del sistema**
    - **Quali passi sono necessari per raggiungere tale obiettivo?**
    - **I passi sono scritti nelle user stories**
  - **Approccio *Scattergun***
    - **Non viene data nessuna struttura alle riunioni**
    - **Le storie vengono scritte sulla base di ciò che emerge dalla riunione**

Esempio di US dal progetto C3

* Cost of Living Allowance (COLA)
  * EEs may receive COLA. Some unions pay a fixed $ amount per pay. Some pay a percentage of base. Amount or percentage varies by union and location. COLA rates change at specified times for different unions.
* Each EE receives COLA in each check.

Il TA relativo (dal progetto C3)

* Acceptance Test
  * Provide EEs from each union and from no union. Pay each EE for three pay periods, test whether COLA amount properly calculated.
  * Determine whether COLA amount properly displayed on check or EFT stub.

Esempio: User Story

Title: Login		
Acceptance Test: LoginTest	Priority: 2	Story Points: 2
A user has to input his/her nickname and password to login the system. If the combination of the nickname and password is not correct, an error message will show, and the user has to input the nickname and password again. Personal page and bid functionality are available only when the user logs in successfully. Administrative functionality is available only when the user with administrative privileges logs in successfully.		

Elementi delle schede

- ***Titolo*: due o tre parole; dovrebbe essere un verbo attivo al presente (simile al nome di un use case)**
- ***Acceptance Test*: Lista dei test di accettazione associati alla storia**
- ***Priorità*: Il cliente decide quali storie devono essere implementate per prime. Solitamente si usa una scala 1-2-3, dove 1 è il max**
- ***Story Points*: Sforzo di sviluppo (*Effort*) stimato**
- ***Descrizione*: alcune frasi che esprimono i passi da eseguire per raggiungere l'obiettivo**

Stima di una storia (1/2)

- **E' preferibile stimare la dimensione delle storie usando come unità di misura i *punti***
- **Un punto corrisponde spesso (ma non sempre) a una giornata di lavoro**
- **Altri considerano come unità di misura la mezza giornata, altri la settimana di lavoro...**
- **Se il programmatore stima che la storia richiederà più di due settimane di lavoro (cioè più di 10 punti) dovrà richiedere al cliente di suddividere la storia in parti più semplici (*split*)**

Stima di una storia (2/2)

- **La stima può essere semplice nel caso in cui si disponga di storie simili già implementate**
- **Quando non si è in grado di fare una stima ragionevole della storia si fanno degli esperimenti, noti come *spike solution* (prototipizzazione rapida)**
  - **Obiettivo delle spike solution: sperimentare soluzioni e avere un'idea più chiara dei tempi necessari per implementare una storia**
  - **Il codice prodotto viene poi di solito scartato**

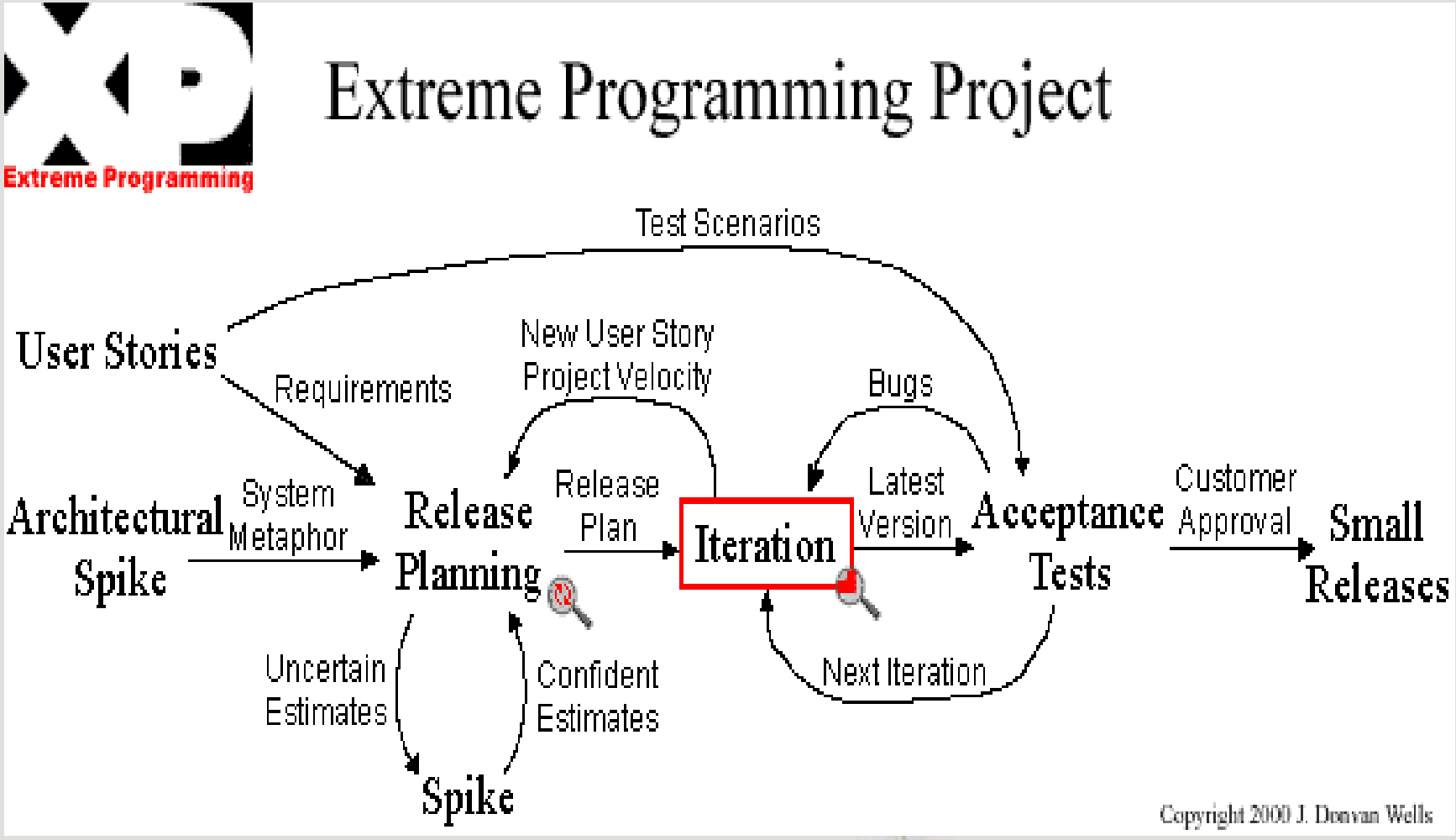
Test di Accettazione

- **Sono scritti dal cliente, per ogni storia può essere definito anche più di un test di accettazione**
- **Sono automatizzati dal team di sviluppo e devono essere ripetibili**
- **Verificano che la funzionalità richiesta dal cliente sia stata implementata correttamente**
- **Una storia si ritiene completata quando vengono superati con successo tutti i test di accettazione ad essa relativi**

Altri Requisiti

- I casi d'uso e le storie, come è stato detto, documentano i requisiti funzionali
- I requisiti non-funzionali e i vincoli non sono facilmente esprimibili come user stories
- Nelle user stories potrebbero essere esplicitati una serie di requisiti non funzionali e di vincoli
  - Esempio: il sistema deve rispondere a determinate interrogazioni in meno di 300 ms.

User Stories e ciclo di vita XP (1/5)



User Stories e ciclo di vita XP (2/5)

- **Il cliente scrive le user stories su index card**
- **Il cliente definisce i test di accettazione per le storie**
- **Il team di sviluppo stima le storie**
- **Il cliente sceglie le storie per il rilascio**
  - **XP è un processo incrementale e iterativo, i rilasci sono frequenti**
  - **Un rilascio avviene di solito ogni 3 mesi (ciclo trimestrale)**
  - **Per ogni rilascio, si effettuano iterazioni di una settimana (ciclo settimanale)**

User Stories e ciclo di vita XP (3/5)

- **Ogni ciclo trimestrale viene rilasciata una versione funzionante del sistema, che implementa un certo numero di User Stories**
- **All'inizio di ogni rilascio si effettua una riunione in cui il cliente sceglie le storie da implementare**
  - **Si rivede brevemente lo stato di tutto il progetto**
  - **Il cliente può modificare, aggiungere o eliminare storie. La modifica e l'eliminazione di storie già implementate sono considerate nuove storie**
  - **Gli sviluppatori possono stimare nuovamente le storie, in base ai risultati del rilascio precedente**

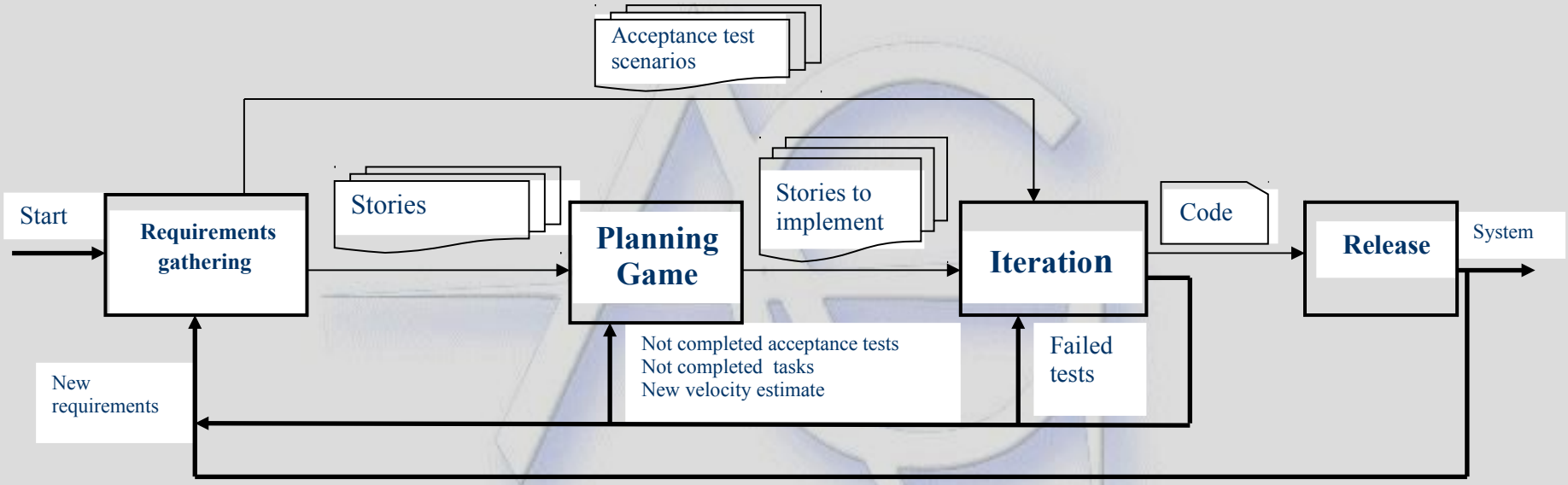
User Stories e ciclo di vita XP (4/5)

- **Nei cicli settimanali viene pianificato lo sviluppo per la nuova iterazione. Ogni ciclo inizia con una riunione di pianificazione**
- **Durante la riunione di pianificazione dell'iterazione:**
  - **Il cliente sceglie le storie da implementare in quella settimana**
  - **Il cliente può aggiungere/modificare/eliminare storie. La modifica e l'eliminazione di storie già implementate sono considerate nuove storie**

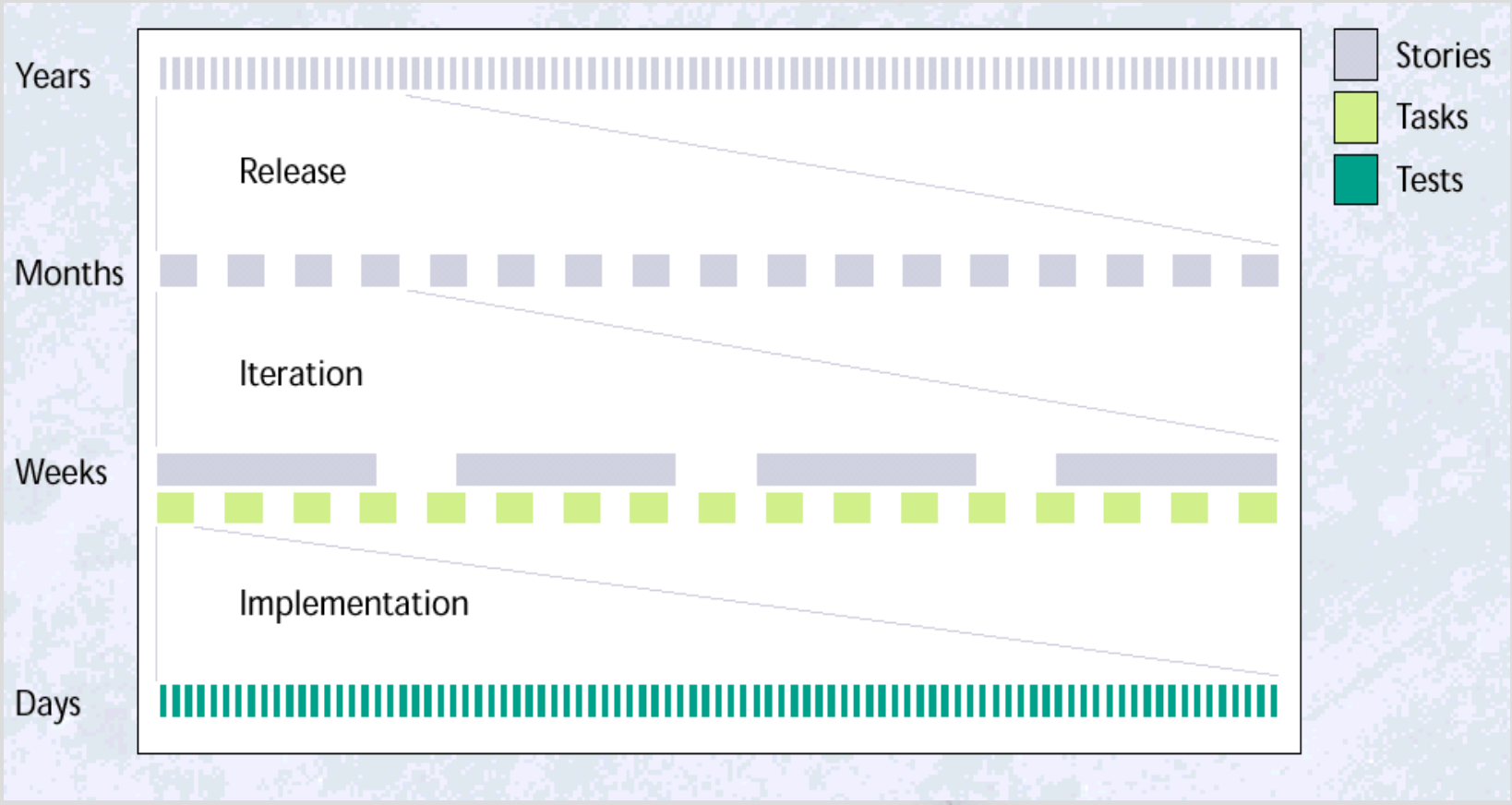
User Stories e ciclo di vita XP (5/5)

- **Durante la riunione di pianificazione dell'iterazione:**
  - **Gli sviluppatori suddividono ciascuna storia in parti più semplici, noti come Task**
  - **Gli sviluppatori stimano i task e i TA**
- **Ogni sviluppatore accetta dei task e/o dei TA**
- **Lo sviluppatore implementa i task (con un compagno: Pair Programming)**
- **Il cliente accetta le storie completate**

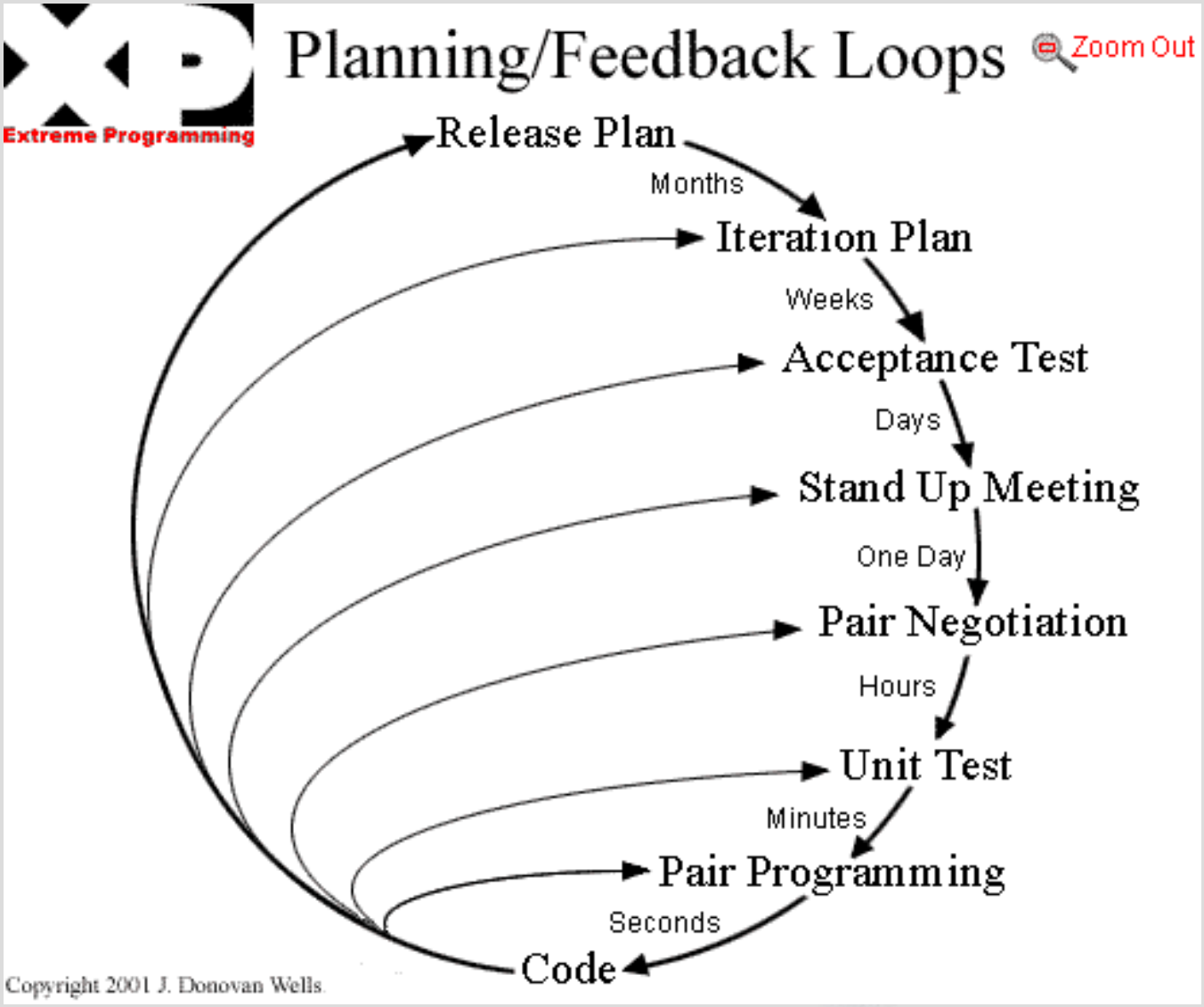
XP Process


Acceptance test scenarios
Start
Requirements gathering
Stories
Planning Game
Stories to implement
Iteration
Code
Release
System
New requirements
Not completed acceptance tests
Not completed tasks
New velocity estimate
Failed tests

XP process at different time scales



XP Cycles


Extreme Programming
Planning/Feedback Loops
Zoom Out
Release Plan
Months
Iteration Plan
Weeks
Acceptance Test
Days
Stand Up Meeting
One Day
Pair Negotiation
Hours
Unit Test
Minutes
Pair Programming
Seconds
Code
Copyright 2001 J. Donovan Wells

The XP Planning Game

- XP proceeds by implementing *User Stories*
- USs are pieces of requirements describing an interaction of the system with a user
- Think of a short Use Case
- The *Planning Game* creates the schedule
- User Stories are prioritized by the customer and estimated by the team in *Story Points*
- *Iterations* of 1-2 weeks

Story points

- Each story is assigned an estimate in story points
- Based on a combination of the size and complexity of the work
- Unitless but numerically relevant estimates
  - A 10-point user story is expected to take twice as long as a 5-point user story
- To start, SP might be the estimated man-days or man-hours needed to complete the story

Estimating in story points:

- Forces the use of relative estimating
  - Studies have shown we're better at this†
- Focuses us on estimating the size, not the duration
  - We derive duration empirically by seeing how much we complete per iteration
- Puts estimates in units that we can add together
  - Time based estimates are not additive

Planning Poker (*planningpoker.com*)

- An iterative approach to estimating, loosely based on wideband Delphi
  1. Each estimator is given a deck of cards, each card has a valid estimate written on it
  2. Customer/Product owner reads a story and it's discussed briefly
  3. Each estimator selects a card that's his or her estimate
  4. Cards are turned over so all can see them
  5. Discuss differences (especially outliers)
  6. Re-estimate until estimates converge

Planning Poker

Estimator	Round 1	Round 2
Susan	3	5
Vadim	8	5
Ann	2	5
Chris	5	8

The Planning Game

- The *Project Velocity* must be estimated (No. of Story Points the team can implement in one iteration)
- Each iteration is assigned the USs with highest priority – *the customer decides!*
- Iterations are *time-boxed*
- If some USs cannot be completed in an iteration, it is moved to the next
- The PV is updated accordingly

The Planning Game

- In this way, the development proceeds implementing increments of the system, with the most important parts first
- An *Iteration Planning Meeting* starts each iteration
- A *Stand-up Meeting* at the beginning of each day
- An *End of Iteration meeting* concludes each iteration, where the work is evaluated and feedback is obtained.

The Four Variables

- Extreme Programming delivers, and steers by Business Value
- Business Value belongs to the customer!
- Four variables:
  - Scope
  - Time
  - Resources (very non-linear)
  - Quality (fixed at maximum)

The contract between the customer and developers

- The customer can set 3 of the 4 variables (in fact, quality and resources are already fixed)
- Developers set the 4th variable
- Usually, the customer defines stories (scope) and sets the time
- Developers say how many stories they can do in that time – using Story Points
- The customer chooses the stories to develop
- *The customer cannot set all 4 variables!!*

Planning game process

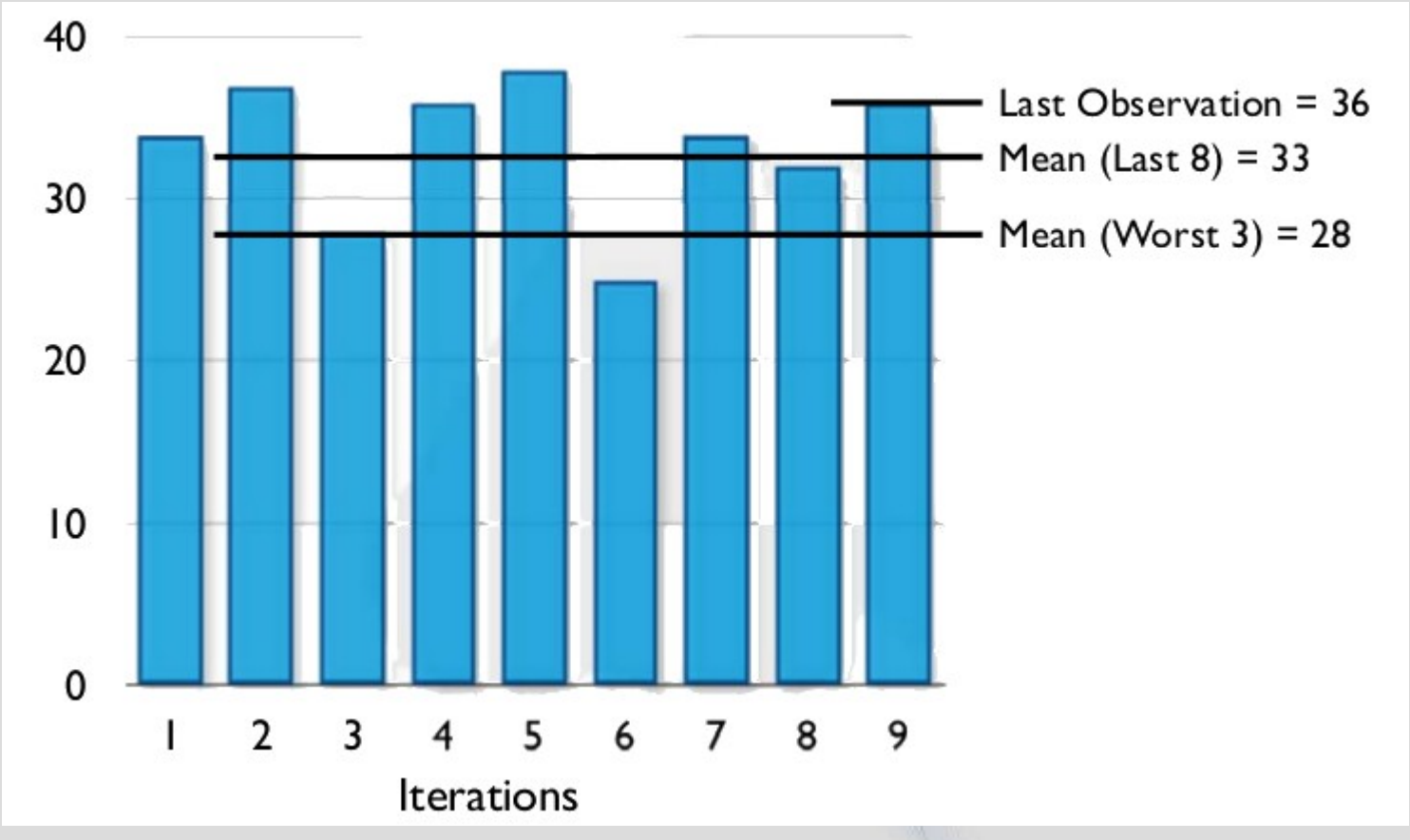
- Customer writes stories on cards
- Customer defines acceptance tests for story
- Programmer estimates stories
- Customer selects stories for release
- Customer selects story for iteration
- Programmer defines tasks for story
- Programmer signs up and estimates tasks
- Programmer does tasks (with partner)
- Customer accepts completed stories

Estimating Project Velocity

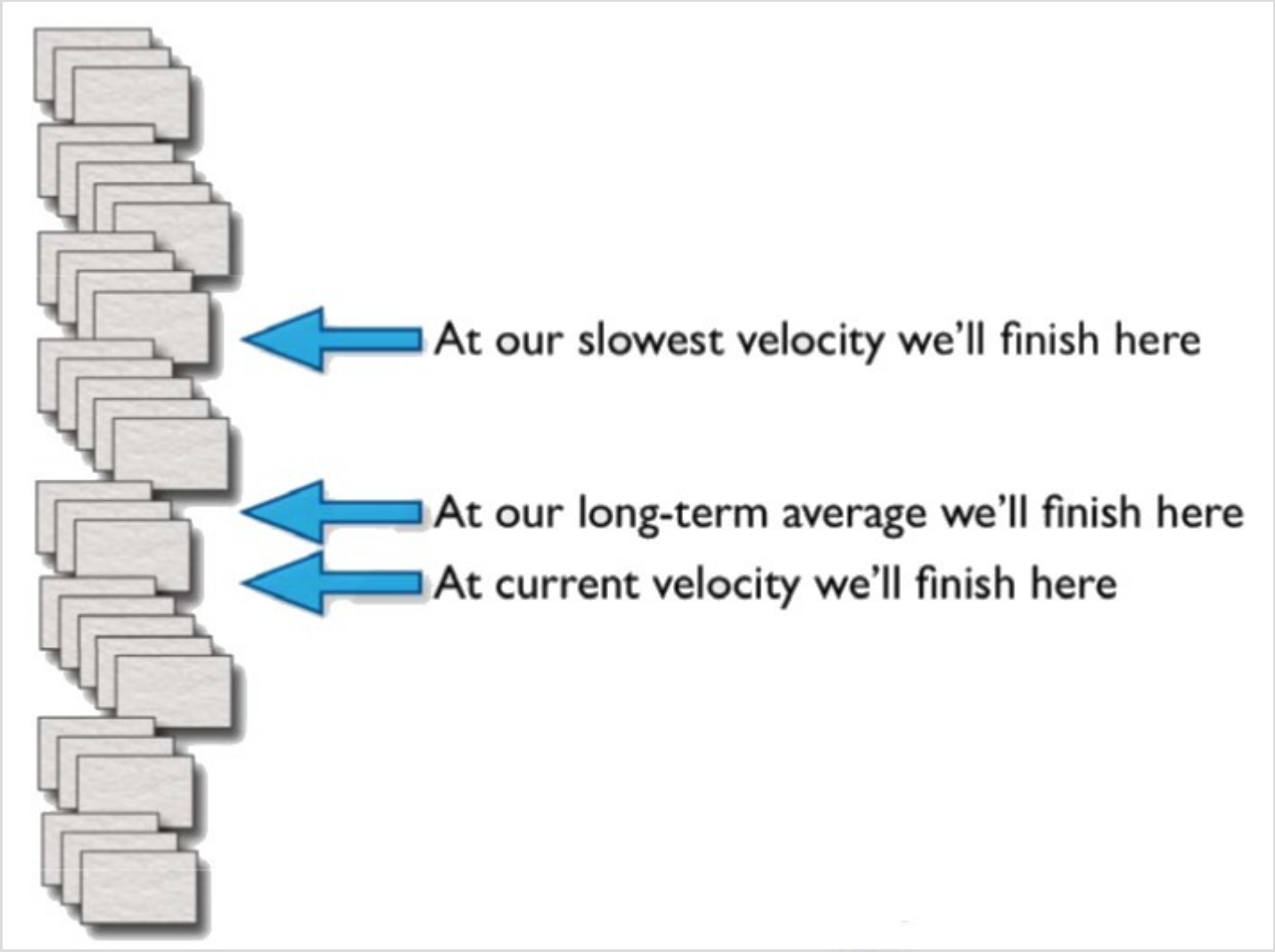
- **Project Velocity = the number of story points that can be completed in one iteration**
- **For the first iteration:**
  - **Estimated according to team evaluation**
- **For the next iterations:**
  - **Estimated using the number of story points completed in past iterations**
- **You can use the last iteration, and average on last *n* iterations, worst-case iterations...**

Estimating Project Velocity

- **Use multiple views of Project Velocity**



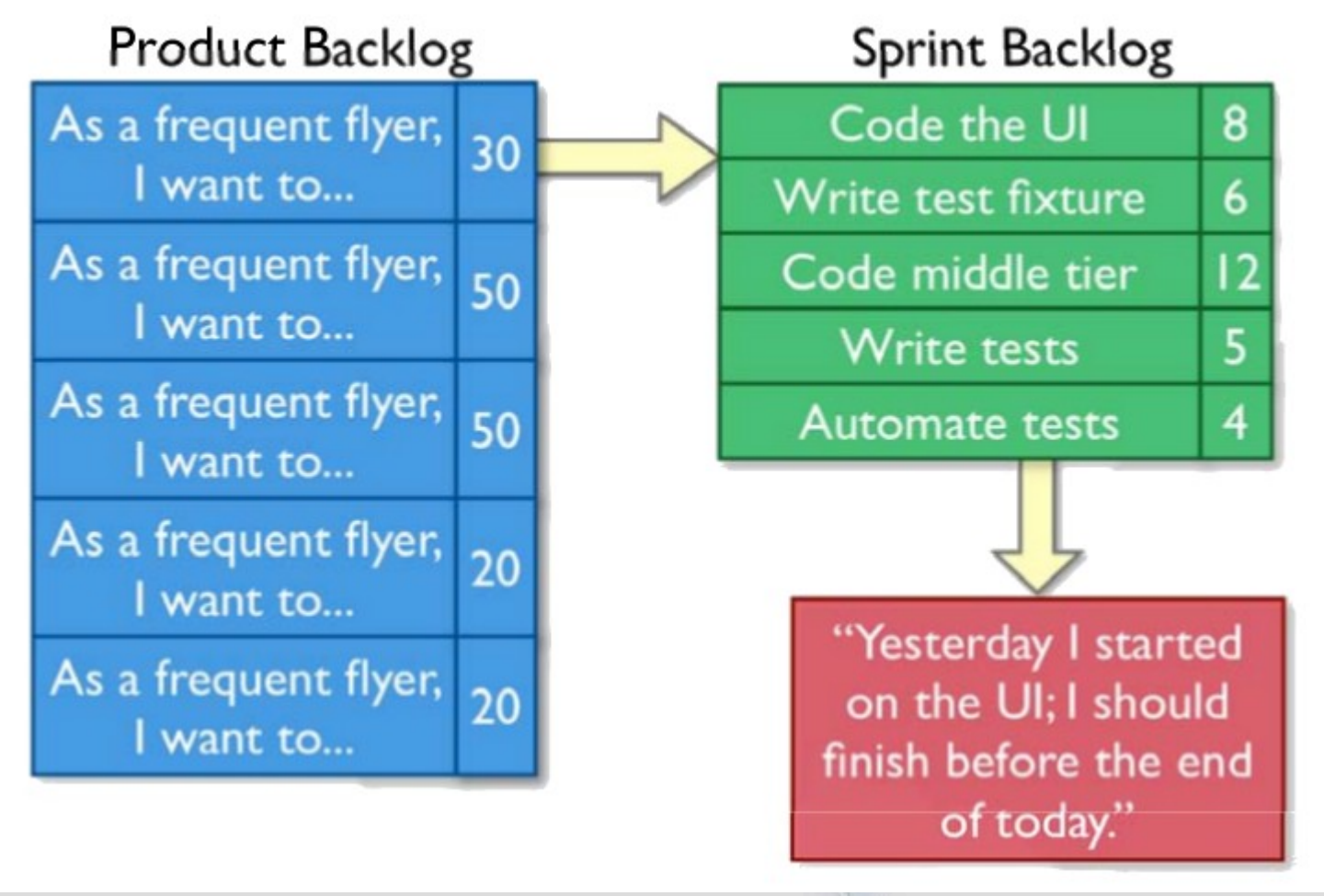
Extrapolate from velocity



Iteration planning meeting

- This meeting is held at the beginnig of each iteration
- The stories to implement are chosen according to their priority, given by the customer
- The team discusses each story and divide it in Tasks, that are in turn estimated (in SP or hours)
- Developers sign up tasks
- The list of tasks is the Sprint Backlog (in Scrum jargon)

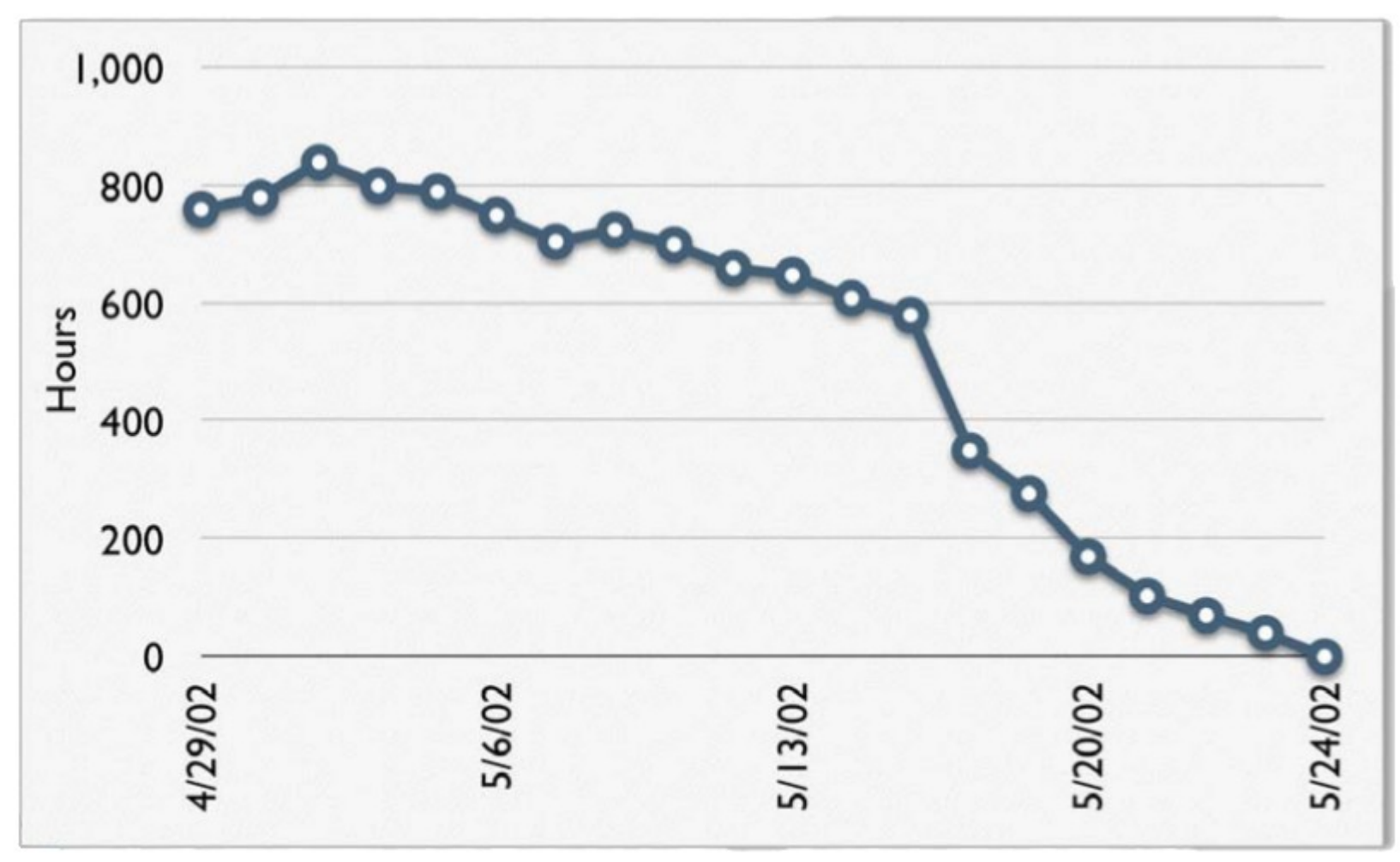
A story and its tasks



Burndown Charts

- Draw the number of Story Points, or hours of works, that remains to be done each day
- This number might even increase, due to:
  - Addition of extra stories to implement
  - Re-estimation of existing stories
- Often, the Iteration burndown is in hours, the Release burndown is in story points
- The Burndown Chart gives the team a feeling of the progress
- It is a typical Scrum tool

An iteration burndown chart



A release burndown chart