



UNIVERSITÀ DEGLI STUDI DI NAPOLI  
**FEDERICO II**

# Ingegneria del Software – Modelli di Processo

Prof. Sergio Di Martino

# Obiettivi della lezione

- Comprendere il concetto di Modello di Processo Software
- Comprendere il Modello di Processo a Cascata
- Comprendere la Metodologia SCRUM

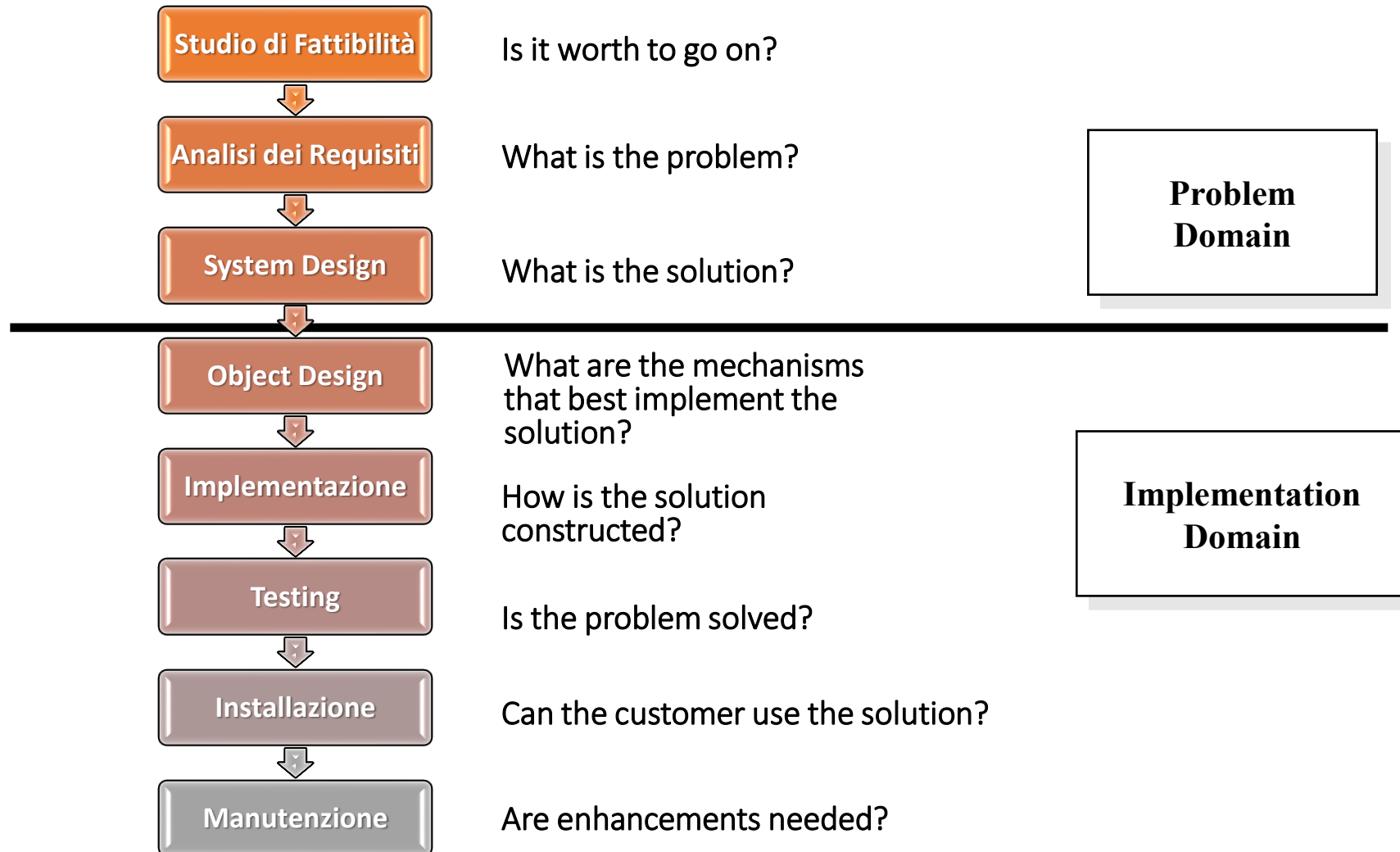
# Processo

- “Un processo è un particolare metodo per fare qualcosa costituito da una sequenza di passi che coinvolgono attività, vincoli e risorse” (Pfleeger)
- “Processo: una particolare metodologia operativa che nella tecnica definisce le singole operazioni fondamentali per ottenere un prodotto industriale” (Zingarelli)
- “Processo software: un metodo per sviluppare del software” (Sommerville)

# Processo software: Standard IEEE 610.12-1990

- “Software development process: The process by which user needs are translated into a software product. The process involves translating user needs into software requirements, transforming the software requirements into design, implementing the design in code, testing the code, and sometimes, installing and checking out the software for operational use.
  - Note: These activities may overlap or be performed iteratively.

# Attività richieste nel processo di sviluppo software



# Problemi nel processo di sviluppo del software

- E' qualcosa di altamente intellettuale e creativo, basato su giudizi delle persone
  - Non è possibile (almeno con i sistemi attuali) automatizzarlo
- I requisiti sono complessi e ambigui
  - Il cliente non sa bene, dall'inizio, cosa deve fare il software
- I requisiti sono variabili
  - Cambiamenti tecnologici, organizzativi, etc...
- Modifiche frequenti sono difficili da gestire
  - Difficile stimare i costi ed identificare cosa consegnare al cliente

I modelli di processo

# Cosa sono i Modelli di Processo

- Come organizziamo la sequenza di passi del processo software, per massimizzare alcune caratteristiche?
- Il processo di sviluppo deve essere modellato esplicitamente, per poter essere gestito e monitorato
- I modelli di processo software sono descrizioni precise e formalizzate delle attività, delle trasformazioni, dei deliverables e degli eventi per realizzare e/o ottenere l'evoluzione del software
- Obiettivo:
  - introdurre stabilità, controllo e organizzazione in una attività tendenzialmente caotica, massimizzando alcune delle caratteristiche del processo



# Modelli di processo: caratteristiche (1)

- Una strutturazione dell'organizzazione del lavoro nelle fabbriche del software in:
  - fasi della produzione,
  - tipi di attività,
  - collegamento ed interfacciamento,
  - controllo e misura,
- ma anche linee guida per: organizzare, pianificare, dimensionare personale, assegnare budget, schedulare e gestire, ...

# Modelli di processo: caratteristiche (2)

- definire e prescrivere prodotti e documenti da rilasciare al committente (deliverables)
- determinare e classificare metodi e strumenti più adatti a supportare le attività previste
- framework per analizzare, stimare, migliorare ...

# Diverse tipologie di Modelli di processo

- Esistono vari Modelli di processo, definiti negli ultimi 30 anni
  - Waterfall (cascata)
  - Evolutivi
  - Trasformatzionale
  - Modelli Agili (XP, SCRUM, etc.)
- Molti aspetti influenzano la definizione del modello
  - specificità dell'organizzazione produttrice
  - know-how
  - area applicativa e particolare progetto
  - strumenti di supporto
  - diversi ruoli produttore/ committente

# Come valutare un modello di processo?

- Così come esistono dei parametri di qualità per un software, esistono dei parametri per valutare la “bontà” di un modello per sviluppare software:
  - **Comprensibilità:** Quanto è facile da apprendere?
  - **Visibilità:** Quanto facilmente si capisce a che punto dello sviluppo si è giunti?
  - **Supportabilità** (CASE tools): Esistono tool di supporto? Quali e quante fasi sono supportate? A che livello?
  - **Accettabilità:** Le persone coinvolte manifestano il loro consenso?
  - **Affidabilità:** Il processo facilita l’individuazione di errori? Il software prodotto è di alta qualità?
  - **Robustezza:** Quanto è robusto il processo nel gestire cambiamenti in corso d’opera?
  - **Manutenibilità:** Il processo è in grado di adattarsi ai cambiamenti nell’organizzazione che lo usa?
  - **Rapidità:** Il processo porta ad un rapido sviluppo del software?

# Il più semplice (e peggiore) modello di processo



- Molto veloce, feedback rapido
- Molti strumenti disponibili
- Specializzato per codifica
- Non incoraggia la documentazione
- Non scala (in the large, in the many)
- Ingestibile durante manutenzione

# Valutiamo l'Edit-Compile-Test

- Comprensibilità:
- Visibilità:
- Supportabilità:
- Accettabilità:
- Affidabilità:
- Robustezza:
- Manutenibilità:
- Rapidità:

Il modello a cascata

# L'approccio Tayloristico



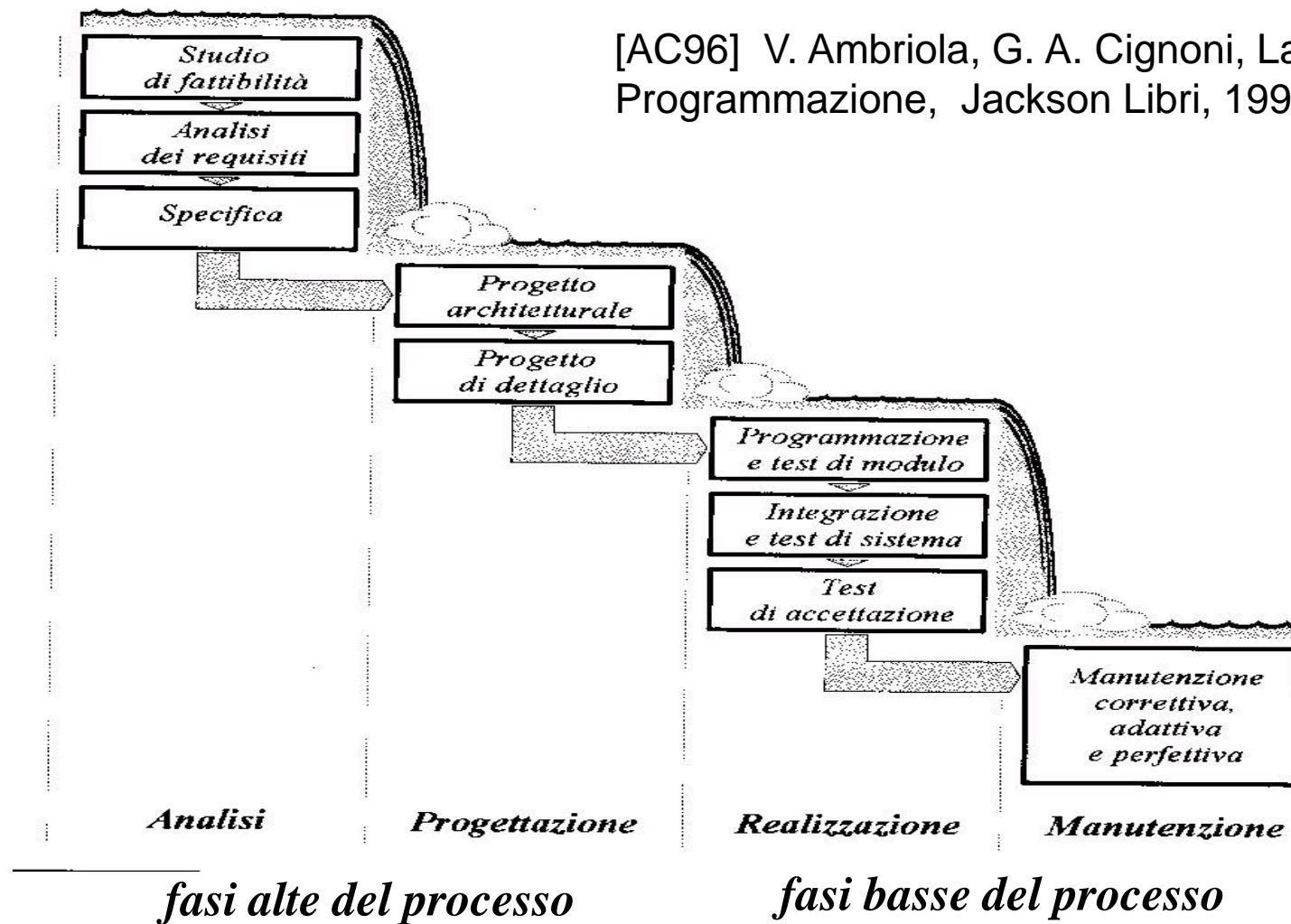
- Frederick Taylor – “The Principles of Scientific Management” (1911)
  - Gestione gerarchica
  - Passi fissi, non fluidi
  - Separare nettamente le postazioni di lavoro
  - Il lavoro, una volta suddiviso, diventa specializzato in un singolo task
  - Processo orientato al prodotto, non al cliente
    - “Any customer can have a car painted any color that he wants so long as it is black” - Henry Ford
  - Processo ripetibile
- E' stato per decenni il fondamento dell'industria mondiale



# Modelli a Cascata (Waterfall) - '70

- Applicazione dell'approccio Tayloristico all'informatica
- Popolare negli anni '70
  - reazione al "code and fix" originario
- Modello sequenziale lineare
  - progressione sequenziale (in cascata) di fasi, senza ricicli, al fine di meglio controllare tempi e costi
  - definisce e separa le varie fasi e attività del processo
    - nullo (o minimo) overlap fra le fasi
  - uscite intermedie: semilavorati del processo (documentazione di tipo cartaceo, programmi)
    - formalizzati in struttura e contenuti
  - consente un controllo dell'evoluzione del processo
    - attività trasversali alle diverse fasi

# Modello Waterfall [AC96]



# Modelli a Cascata: organizzazione sequenziale delle fasi

- Ogni fase raccoglie un insieme di attività omogenee per metodi, tecnologie, skill del personale, etc.
- Ogni fase è caratterizzata da:
  - Attività (tasks),
  - Prodotti di tali attività (deliverables),
  - Controlli di qualità/stato (quality control measures)
- La fine di ogni fase è un punto rilevante del processo (milestone)
- I semilavorati output di una fase sono input alla fase successiva
- I prodotti di una fase vengono “congelati”, ovvero non sono più modificabili se non innescando un processo formale e sistematico di modifica

# I documenti prodotti con il modello Waterfall

<b>Activity</b>	<b>Output documents</b>
Requirements analysis	Feasibility study, Outline requirements
Requirements definition	Requirements document
System specification	Functional specification, Acceptance test plan Draft user manual
Architectural design	Architectural specification, System test plan
Interface design	Interface specification, Integration test plan
Detailed design	Design specification, Unit test plan
Coding	Program code
Unit testing	Unit test report
Module testing	Module test report
Integration testing	Integration test report, Final user manual
System testing	System test report
Acceptance testing	Final system plus documentation

# Valutiamo il modello a cascata

- Comprensibilità:
- Visibilità :
- Supportabilità:
- Accettabilità:
- Affidabilità:
- Robustezza:
- Mantenibilità:
- Rapidità:

# Modello a cascata: vantaggi e svantaggi

- Pro

- ha definito molti concetti utili (semilavorati, fasi ecc.)
- ha rappresentato un punto di partenza importante per lo studio dei processi SW
- facilmente comprensibile e applicabile
- E' un modello molto rigoroso: si applica bene in qualunque contesto in cui i requisiti siano stabili e chiari

- Contro

- interazione con il committente solo all'inizio e alla fine
  - requisiti congelati alla fine della fase di analisi
  - requisiti utente spesso imprecisi: "l'utente sa quello che vuole solo quando lo vede"
  - Errori nei requisiti scoperti solo alla fine del processo
- il nuovo sistema software diventa installabile solo quando è totalmente finito
  - né l'utente né il management possono giudicare prima della fine dell'adesione del sistema alle proprie aspettative

# Nella realtà ...

- l'applicazione evolve durante tutte le fasi
  - overlap e loop sono inevitabili!
  - in alcuni casi è auspicabile sviluppare prima una parte del sistema e poi completarlo (utente finale= mercato)
  - ... la manutenzione non può essere considerata marginale

# I Modelli Agili




# Agile Methodologies

- In many cases, software development doesn't fit well with traditional "engineering" approaches.
- Main reasons for failures:
  - Lack of input from stakeholders
  - Lack of details in requirements
  - Change of requirements during development
- Traditional "predictive" approaches are not able to deal with these issues.
- The solution: Smaller Incremental development!
- Agile Methodologies (or lightweight) = a development methodology that involves the client as much as possible, in order to minimize the risk of failure.
- How to have the client involved? By continuously providing deliverables, each of them developed in a iteration.

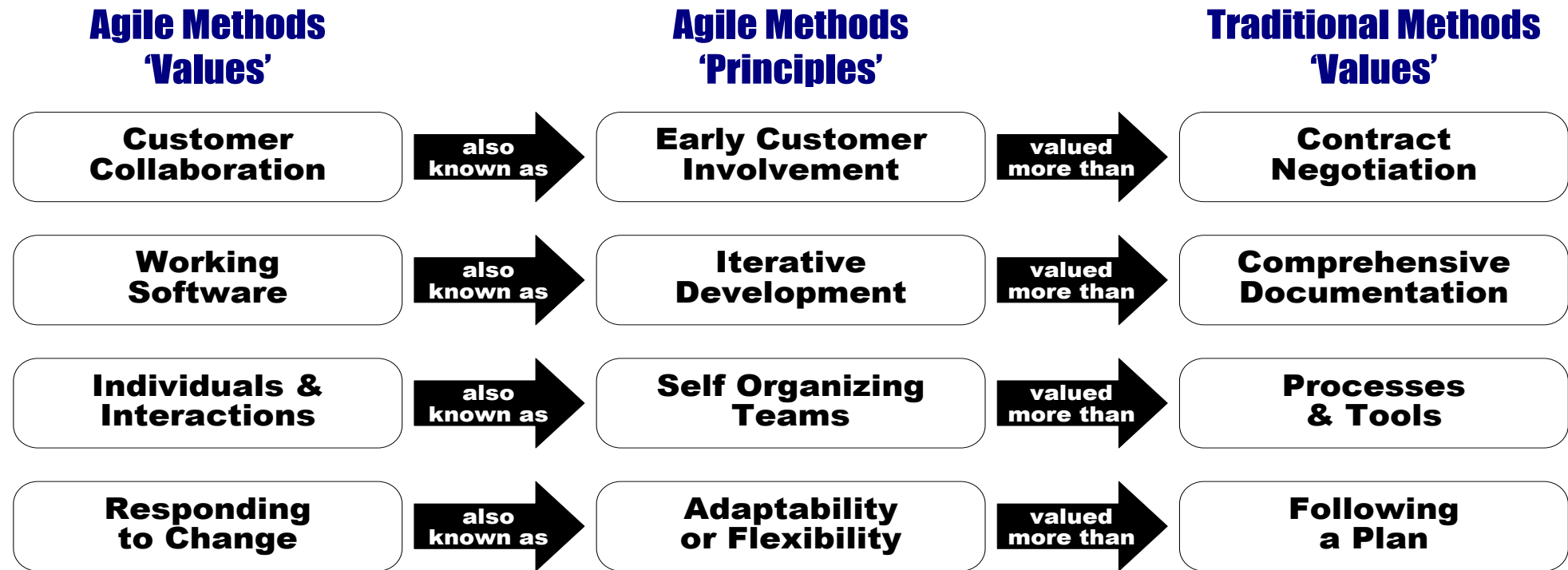
# Iterations

- Agile methods require to develop software in many time-spans: the “Iterations”.
- Each Iteration is a kind of stand-alone project, composed of:
  - Requirement Analysis
  - Planning
  - Implementation and Testing
  - Documentation
- The goal: provide the client as soon as possible and continuously with some “working” stuff, in order to get feedbacks

# Agile Methodologies - history

- During 1990'ies:
    - SCRUM (Ken Schwaber)
    - DSDM (DSDM-consortium)
    - Adaptive Software Development (Jim High Smith)
    - Crystal (Alistair Cockburn)
    - Feature Driven Development
    - Pragmatic Programming
    - Extreme Programming (Kent Beck)
    - ...
  - 11th of February 2001:
    - Snowbird ski resort in Utah mountains,
    - 17 'methodology' people, "The Agile Alliance"
    - "Agile" in stead of "light"
- 
- The Agile Manifesto

# LightWeight vs. HeavyWeight



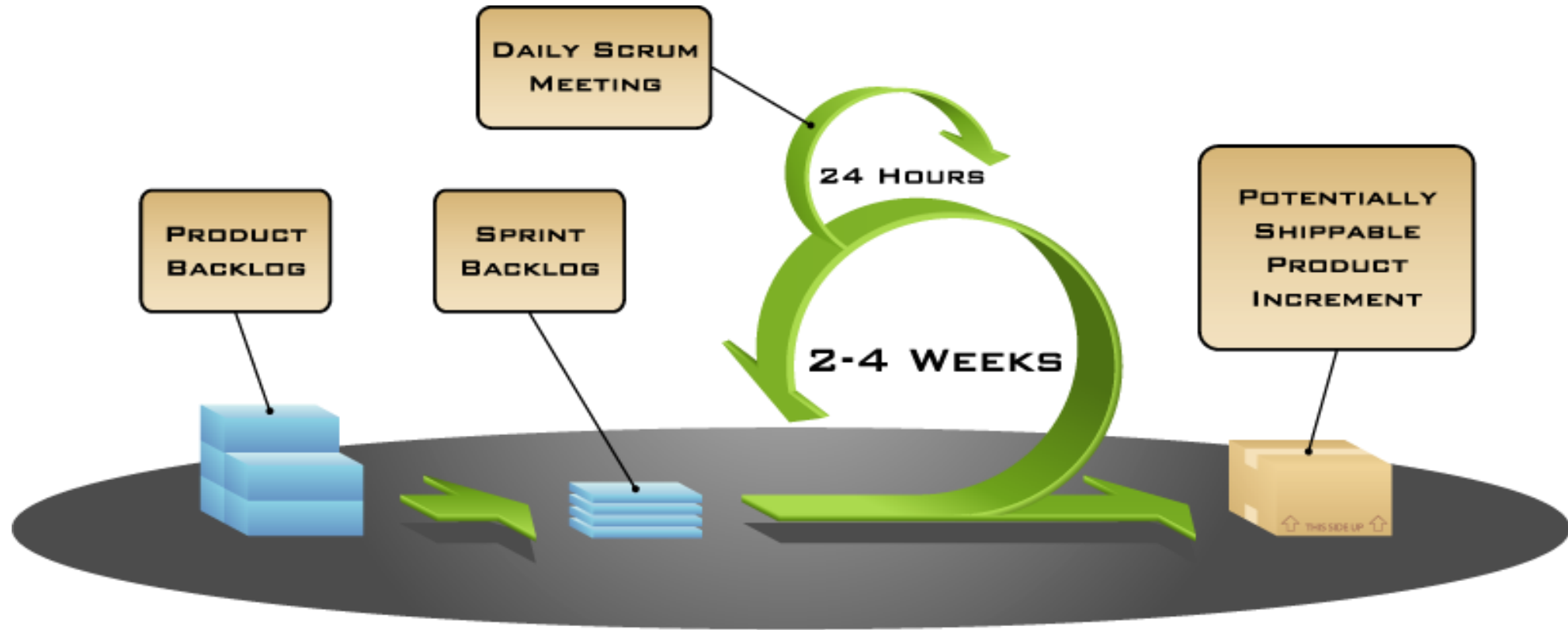
## Scrum in 100 words

- Scrum is an agile process that allows us to focus on delivering the highest business value in the shortest time.
- It allows us to rapidly and repeatedly inspect actual working software (every two weeks to one month).
- The business sets the priorities. Teams self-organize to determine the best way to deliver the highest priority features.
- Every two weeks to a month anyone can see real working software and decide to release it as is or continue to enhance it for another sprint.

# Characteristics

- Product progresses in a series of (about) one month-long “sprints”
- Requirements are captured as items in a list of “product backlog”
- No specific engineering practices prescribed
- Uses generative rules to create an agile environment for delivering projects
- Self-organizing teams
- One of the “agile processes”

# Putting it all together



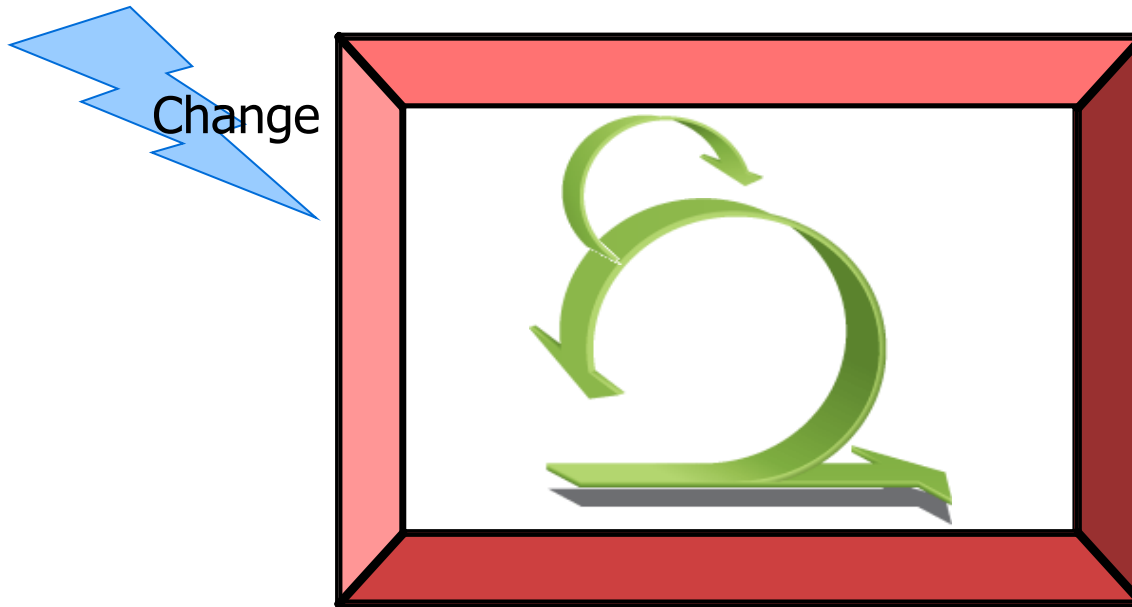
# Sprints

- Scrum projects make progress in a series of “sprints”
  - Analogous to Extreme Programming iterations
- Typical duration is 2–4 weeks or a calendar month at most
- A constant duration leads to a better rhythm
- Product is designed, coded, and tested during the sprint



# No changes during a sprint

- Plan sprint durations around how long you can commit to keeping change out of the sprint



# Scrum framework

## Roles

- Product owner
- ScrumMaster
- Team

## Ceremonies

- Sprint planning
- Sprint review
- Sprint retrospective
- Daily scrum meeting

## Artifacts

- Product backlog
- Sprint backlog
- Burndown charts

# Scrum framework

## Roles

- Product owner
- ScrumMaster
- Team

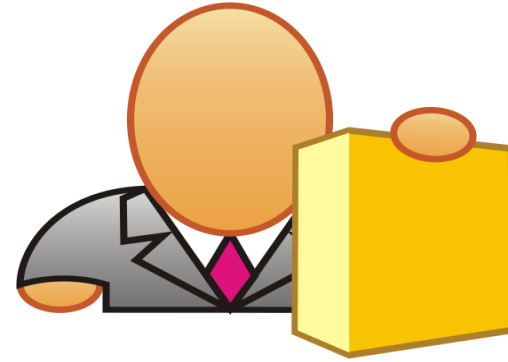
## Ceremonies

- Sprint planning
- Sprint review
- Sprint retrospective
- Daily scrum meeting

## Artifacts

- Product backlog
- Sprint backlog
- Burndown charts

# Product owner



- Define the features of the product
- Decide on release date and content
- Be responsible for the profitability of the product (ROI)
- Prioritize features according to market value
- Adjust features and priority every iteration, as needed
- Accept or reject work results

# The ScrumMaster



- Represents management to the project
- Responsible for enacting Scrum values and practices
- Removes impediments
- Ensure that the team is fully functional and productive
- Enable close cooperation across all roles and functions
- Shield the team from external interferences

# The team



- Typically 5-9 people
- Cross-functional:
  - Programmers, testers, user experience designers, etc.
- Members should be full-time
  - May be exceptions (e.g., database administrator)
- Teams are self-organizing
  - Ideally, no titles but rarely a possibility
- Membership should change only between sprints

# Scrum framework

## Roles

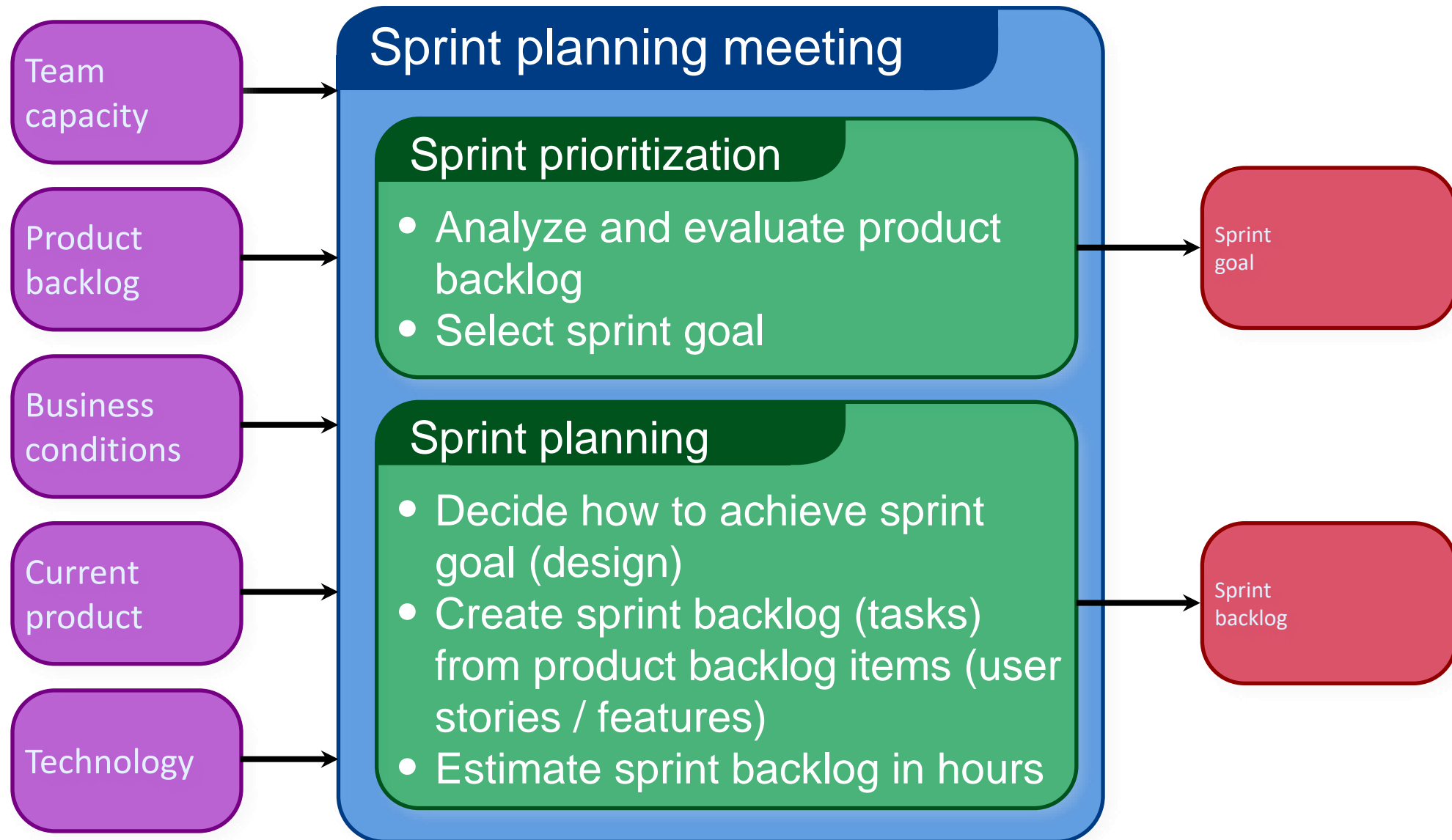
- Product owner
- ScrumMaster
- Team

## Ceremonies

- Sprint planning
- Sprint review
- Sprint retrospective
- Daily scrum meeting

## Artifacts

- Product backlog
- Sprint backlog
- Burndown charts





# Sprint planning

- Team selects items from the product backlog they can commit to completing
- Sprint backlog is created
  - Tasks are identified and each is estimated (1-16 hours)
  - Collaboratively, not done alone by the ScrumMaster
- High-level design is considered

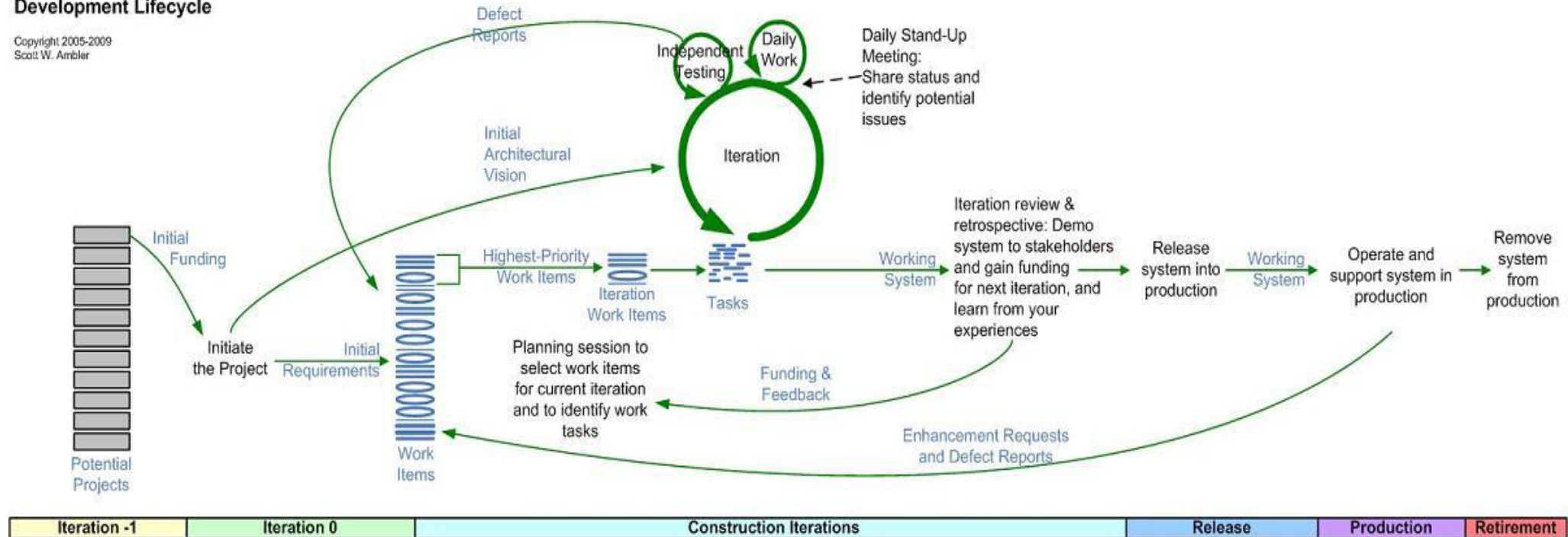
As a vacation planner,  
I want to see photos of the  
hotels.

Code the middle tier (8 hours)  
Code the user interface (4)  
Write test fixtures (4)  
Code the foo class (6)  
Update performance tests (4)

# Detailed view of the Scrum Development Lifecycle

## Agile System Development Lifecycle

Copyright 2005-2009  
Scott W. Ambler



# The daily scrum

- Parameters
  - Daily
  - 15-minutes
  - Stand-up
- Not for problem solving
  - Whole world is invited
  - Only team members, ScrumMaster, product owner, can talk
- Helps avoid other unnecessary meetings



# Everyone answers 3 questions

1

What did you do yesterday?

2

What will you do today?

3

Is anything in your way?

- These are *not* status for the ScrumMaster
  - They are commitments in front of peers

# The sprint review

- At the end of each sprint, the Team presents what it accomplished during the sprint
- Typically takes the form of a demo of new features or underlying architecture
- Informal
  - 2-hour prep time rule
  - No slides
- Whole team participates
- Invite the world



# Sprint retrospective

- Improve the process, by taking a look at what is and is not working
- Typically 15–30 minutes
- Done after every sprint
- Whole team participates
  - ScrumMaster
  - Product owner
  - Team
  - Possibly customers and others

# Sprint retrospective: Start/ Stop/ Continue

- Whole team gathers and discusses what they'd like to:

Start doing

Stop doing

Continue doing

This is just one  
of many ways to  
do a sprint  
retrospective.

# Scrum framework

## Roles

- Product owner
- ScrumMaster
- Team

## Ceremonies

- Sprint planning
- Sprint review
- Sprint retrospective
- Daily scrum meeting

## Artifacts

- Product backlog
- Sprint backlog
- Burndown charts



# Product backlog

- The Backlog is the set of requirements
- A list of all desired work on the project
- Ideally expressed such that each item has value to the users or customers of the product
- Prioritized by the product owner
- Reprioritized at the start of each sprint



This is the product backlog

# A sample product backlog

Backlog item	Estimate
Allow a guest to make a reservation	3
As a guest, I want to cancel a reservation.	5
As a guest, I want to change the dates of a reservation.	3
As a hotel employee, I can run RevPAR reports (revenue-per-available-room)	8
Improve exception handling	8
...	30
...	50

# The sprint goal

- A short statement of what the work will be focused on during the sprint

## Database Application

Make the application run on SQL Server in addition to Oracle.

## Life Sciences

Support features necessary for population genetics studies.

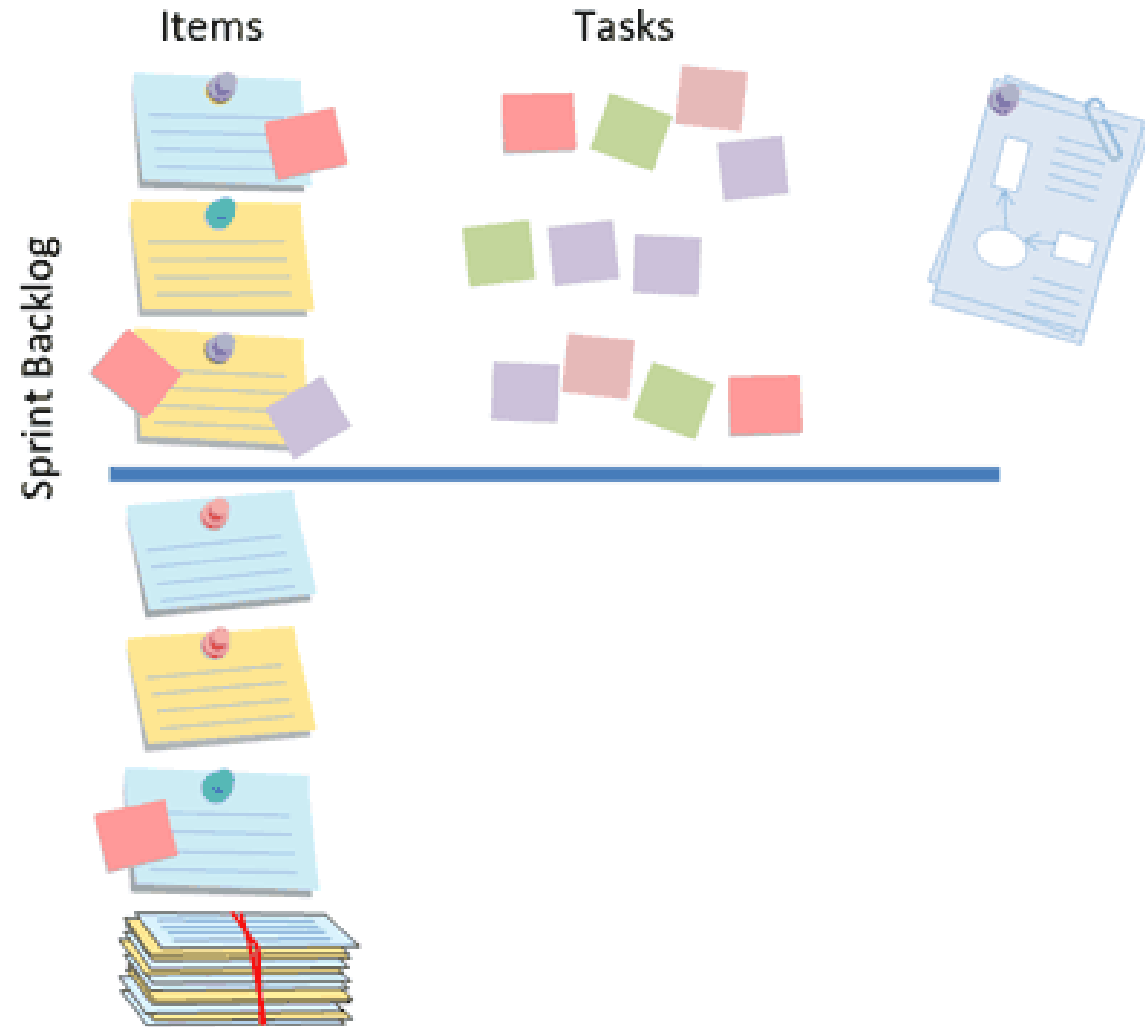
## Financial services

Support more technical indicators than company ABC with real-time, streaming data.

# Managing the sprint backlog

- Individuals sign up for work of their own choosing
  - Work is never assigned
- Estimated work remaining is updated daily
- Any team member can add, delete or change the sprint backlog
- Work for the sprint emerges
- If work is unclear, define a sprint backlog item with a larger amount of time and break it down later
- Update work remaining as more becomes known

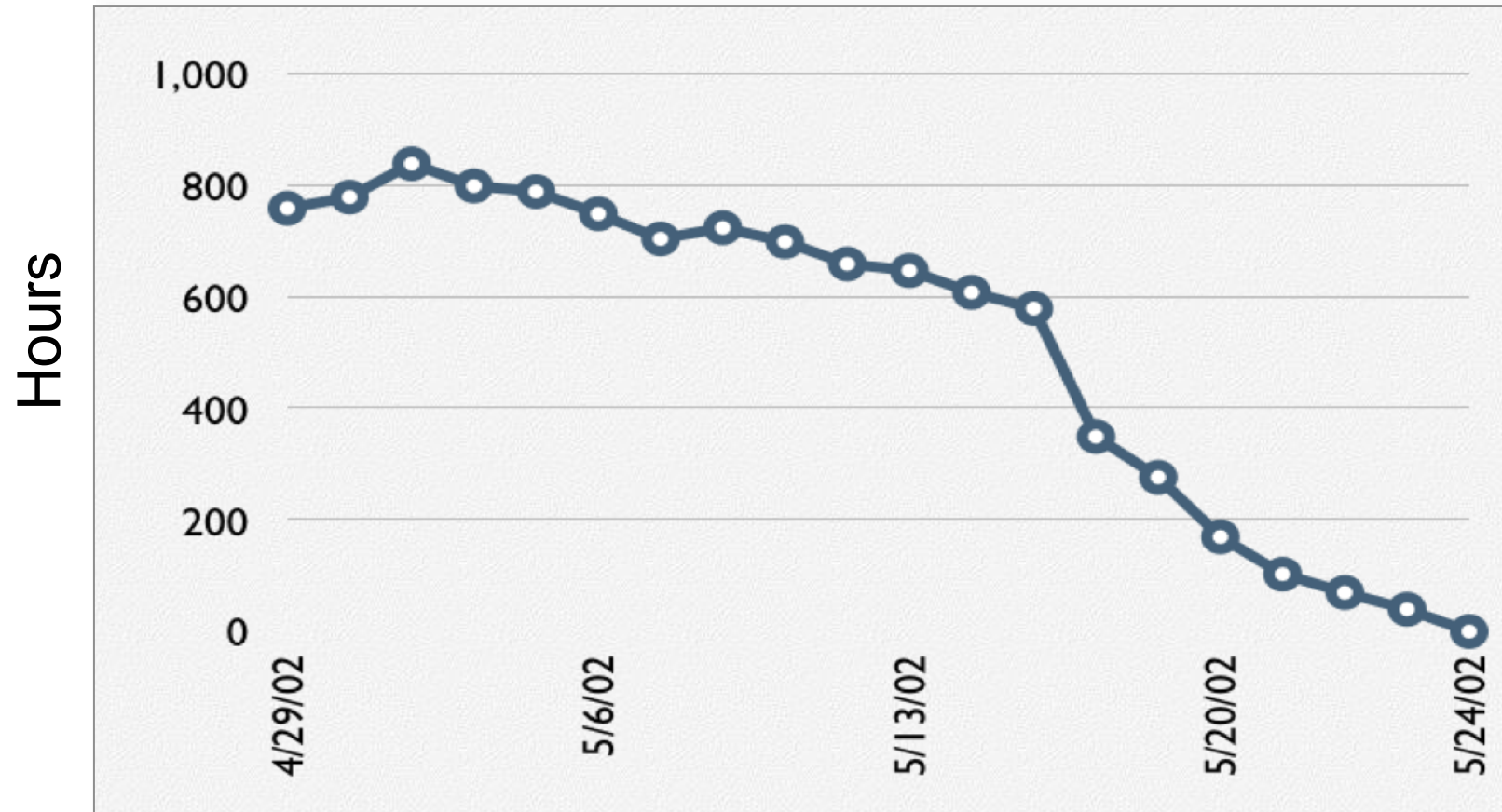
# Sprint Backlog



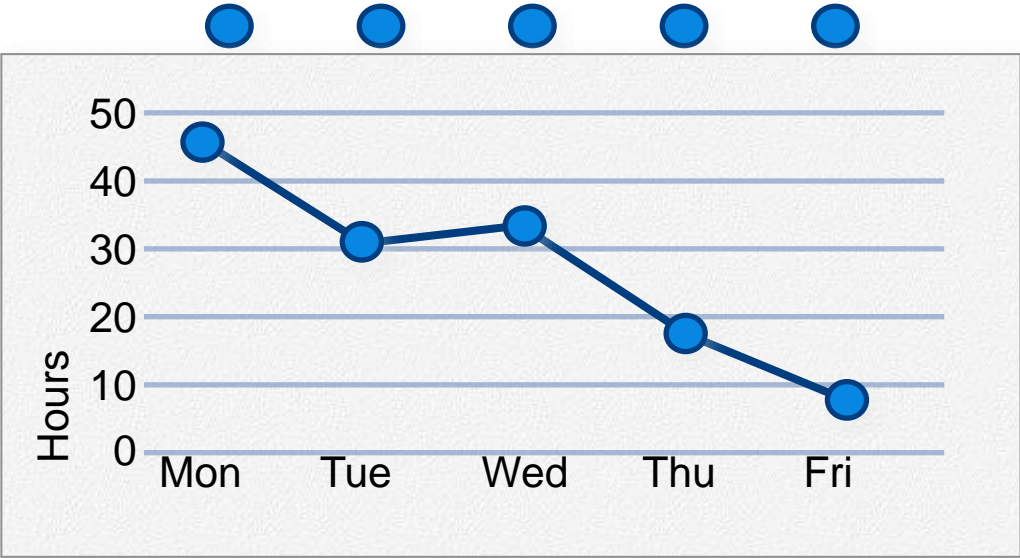
# A sprint backlog

Tasks	Mon	Tues	Wed	Thur	Fri
Code the user interface	8	4	8		
Code the middle tier	16	12	10	4	
Test the middle tier	8	16	16	11	8
Write online help	12				
Write the foo class	8	8	8	8	8
Add error logging			8	4	

# A sprint burndown chart



Tasks	Mon	Tues	Wed	Thur	Fri
Code the user interface	8	4	8		
Code the middle tier	16	12	10	7	
Test the middle tier	8	16	16	11	8
Write online help	12				





# Valutiamo SCRUM

- Comprensibilità:
- Visibilità:
- Supportabilità:
- Accettabilità:
- Affidabilità:
- Robustezza:
- Manutenibilità:
- Rapidità:

# A Scrum reading list

- Agile and Iterative Development: A Manager's Guide by Craig Larman
- Agile Estimating and Planning by Mike Cohn
- Agile Project Management with Scrum by Ken Schwaber
- Agile Retrospectives by Esther Derby and Diana Larsen

# A Scrum reading list

- Agile Software Development Ecosystems by Jim Highsmith
- Agile Software Development with Scrum by Ken Schwaber and Mike Beedle
- Scrum and The Enterprise by Ken Schwaber
- Succeeding with Agile by Mike Cohn
- User Stories Applied for Agile Software Development by Mike Cohn