

Ingegneria del Software

Ingegneria del software e
Processi di sviluppo

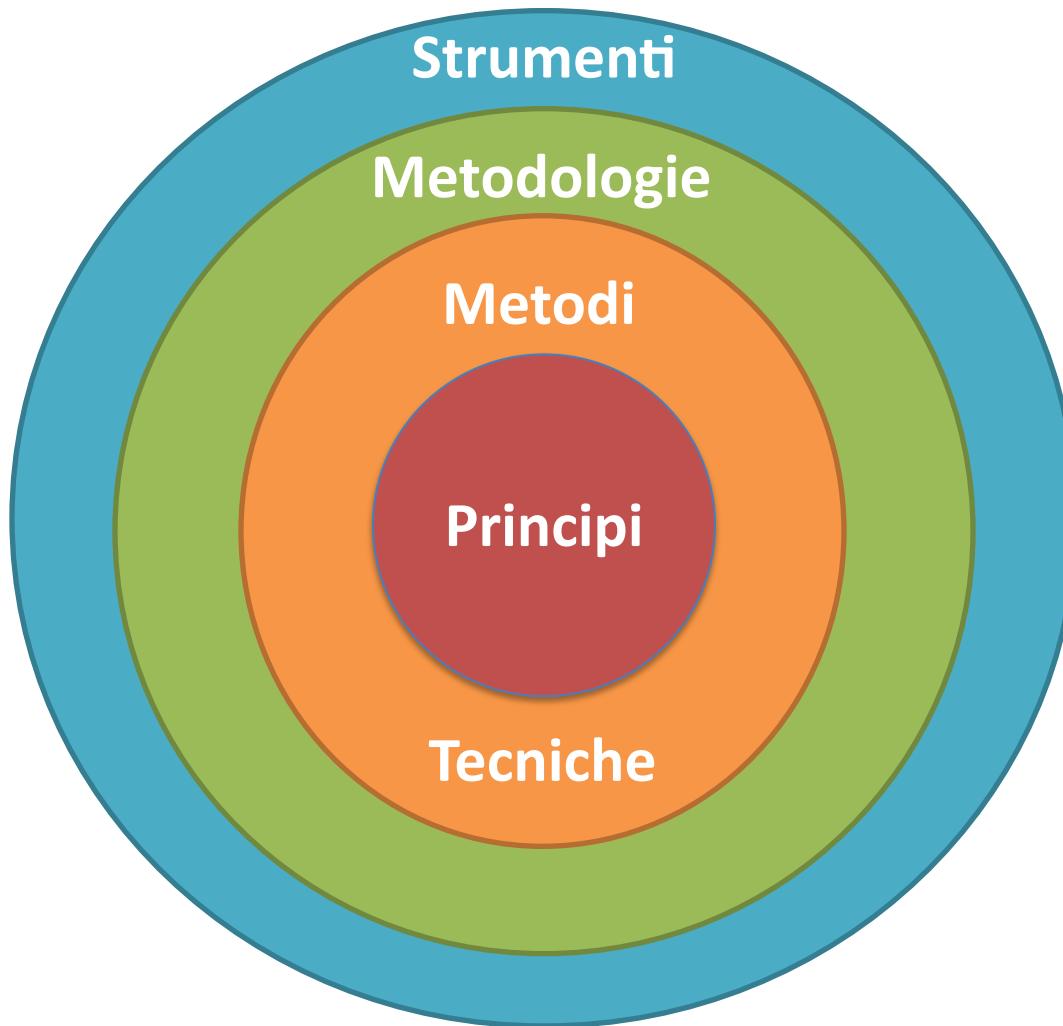
Prof. Paolo Giorgini

A.A. 2024/2025

Alcune definizioni di IS

- Fritz Bauer [1969]: un punto di partenza
 - L'ingegneria del software è l'istituzione e l'impegno di principi ingegneristici ben fondati, allo scopo di ottenere in modo economico software affidabile ed efficiente
- Scott Whitmire
 - Più che una disciplina o insieme di conoscenze, è un modo di affrontare un problema
- IEEE [1993]
 - (1) applicazione di una strategia sistematica, disciplinata e misurabile allo sviluppo, esercizio e manutenzione del software; (2) studio di strategie di cui al punto (1).

IS come tecnologia stratificata



Principi

- Rigorosità e formalità
- Separazione dei problemi
- Modularità
- Astrazione
- Provisióne dei cambiamenti
- Generalità
- Incrementalità

Rigorosità e Formalità

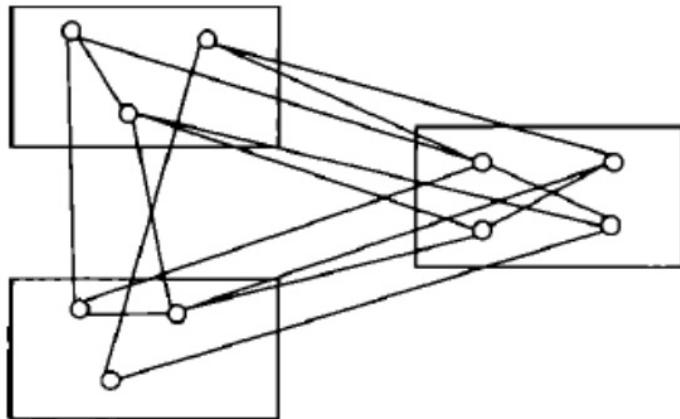
- Ingegneria del software
 - Creativa ma ... richiede sistematicità
- Rigorosità e formalità
 - Necessario complemento a creatività per aumentare il livello di confidenza
- Esempi
 - Prova formale di correttezza
 - Documentazione precisa

Separazione dei problemi

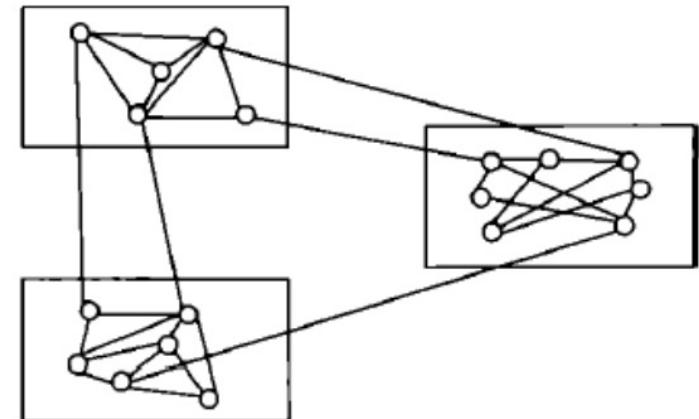
- Separare i problemi per gestire la complessità
 - Dividi et impara
- Esempi
 - Diverse fasi del processo
 - Front-end / back-end di un web app
 - Diverse tecniche di test per diversi problemi

Modularità

- Dividere un sistema in pezzi semplici: moduli
 - Alta coesione
 - Basso accoppiamento



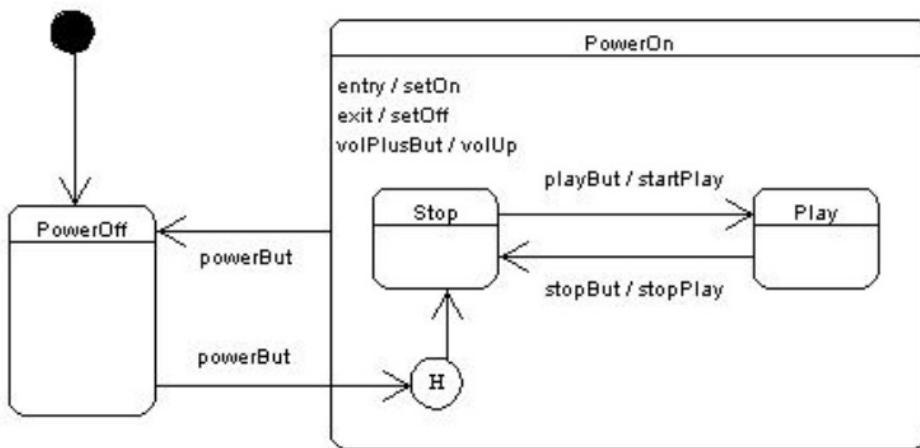
(a)



(b)

Astrazione

- Ignorare i dettagli
 - Premessa per i modelli



```
class CPlayer {  
    CPlayerState state; // state object  
    AbsOnState onHistory; // for History state  
    // References for all the state objects  
    PowerOff offState; PowerOn onState;  
    Stop stopState; Play playState;  
    CPlayer() { // constructor  
        // create state objects only once  
        offState = new PowerOff(this); onState = new PowerOn(this);  
        stopState= new Stop(this,onState);  
        playState = new Play(this,onstate);  
        state = offState; // sets the default state  
        onHistory = stopState; // setting the history for the first time }
```

```
void setState(CPlayerState st) { // setting new state  
    state = st;  
    //sets the most recent active substate of PowerOn state  
    if(state.equals(onState)) { onState.substate = onHistory; }  
    state.entry(); // executes the entry action }  
// Delegates incoming Events to concrete state class  
void powerBut() { state.powerBut(); }  
void volPlusBut() { state.volPlusBut(); }  
void playBut() { state.playBut(); }  
void stopBut() { state.stopBut(); }  
// Actions becomes methods in Super Context class  
void setOn() {...}  
void setOff() {...}  
void volUp() {...}  
void startPlay() {...}  
void stopPlay() {...}  
}  
class CPlayerState { // Abstract Class  
    CPlayer cp; // reference to the context object  
    .... // Declaring abstract methods  
}  
class PowerOff extends CPlayerState {  
    void powerBut() {  
        exit(); // executes the exit action of current state  
        cp.setState(cp.onState); // change to PowerOn state }  
}  
class PowerOn extends CPlayerState {  
    AbsOnState substate; // reference for nested statechart  
    void entry() {  
        substate.entry(); // executes entry action of active substate  
        cp.setOn(); // executes entry action }  
    void exit() {  
        cp.setOff(); // executes exit action  
        cp.onHistory = substate; // adjusting the history node }  
    void volPlusBut() { // Internal Transition }
```

Previsione dei cambiamenti

- Capacità di affrontare l'evoluzione
 - Gestire quello che potrebbe cambiare in futuro
 - Prima e dopo la realizzazione
- Esempio
 - Gestione delle configurazioni
 - Mantiene traccia delle relazione fra i vari moduli e le evoluzioni delle configurazioni dei vari moduli
 - La verione 3.0 e' stata derivata dalla 2.95 in funzione di certi tipi di cambiamenti, la 2.95 e' compatibile con la 2.7 di un altro modulo

Generalità

- Risolvere il problema generale, non solo l'istanza
 - Riusabilità: possiamo riusare la soluzione generale in istanze e situazioni diverse
 - Attenzione ai costi
 - Generalizzare ha un costo e chiaramente non vogliamo avere la soluzione per tutti i problemi

Incrementalità

- Procedere a passi successivi
- Esempi
 - Sviluppo per prototipi incrementali
 - Iterazioni successive con il cliente che vede crescere il sistema
 - Aumento dei costi / tempi

Processo di sviluppo

- Stabilisce quando e come qualcuno fa cosa, per raggiungere un determinato obiettivo



Scelta e adattamento

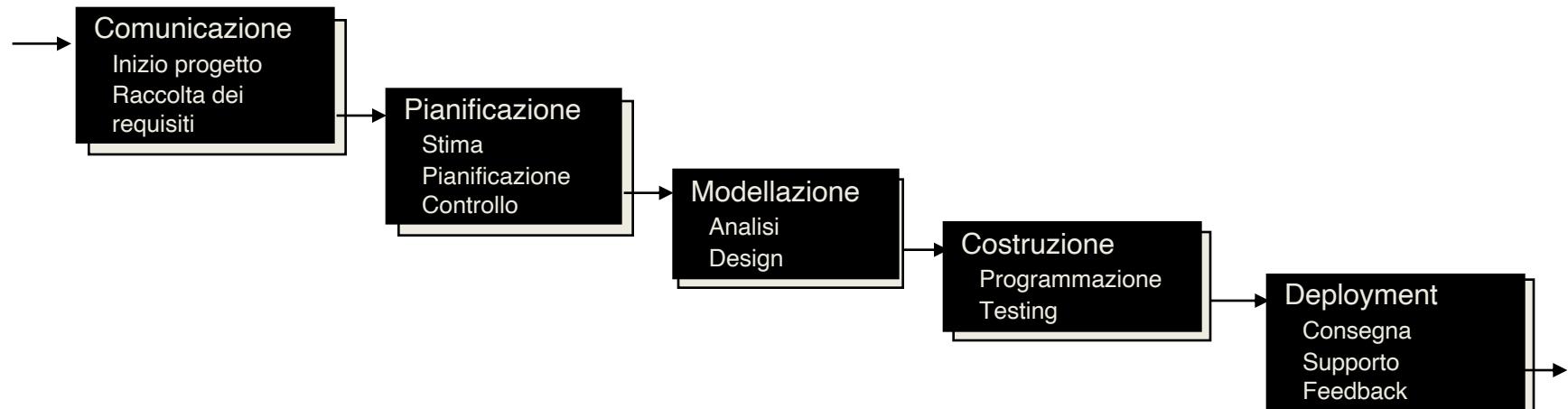
- Legato al problema, team, cultura aziendale
- Differenze tra modelli
 - Flusso delle attività, dei task e loro interdipendenza
 - Attività di controllo del progetto
 - Dettaglio e rigore del processo
 - Coinvolgimento degli stakeholders
 - Autonomia del team
 - Prescrizione dei ruoli e loro organizzazione
- Modelli agili
 - Alta adattabilità

Valutazione del processo

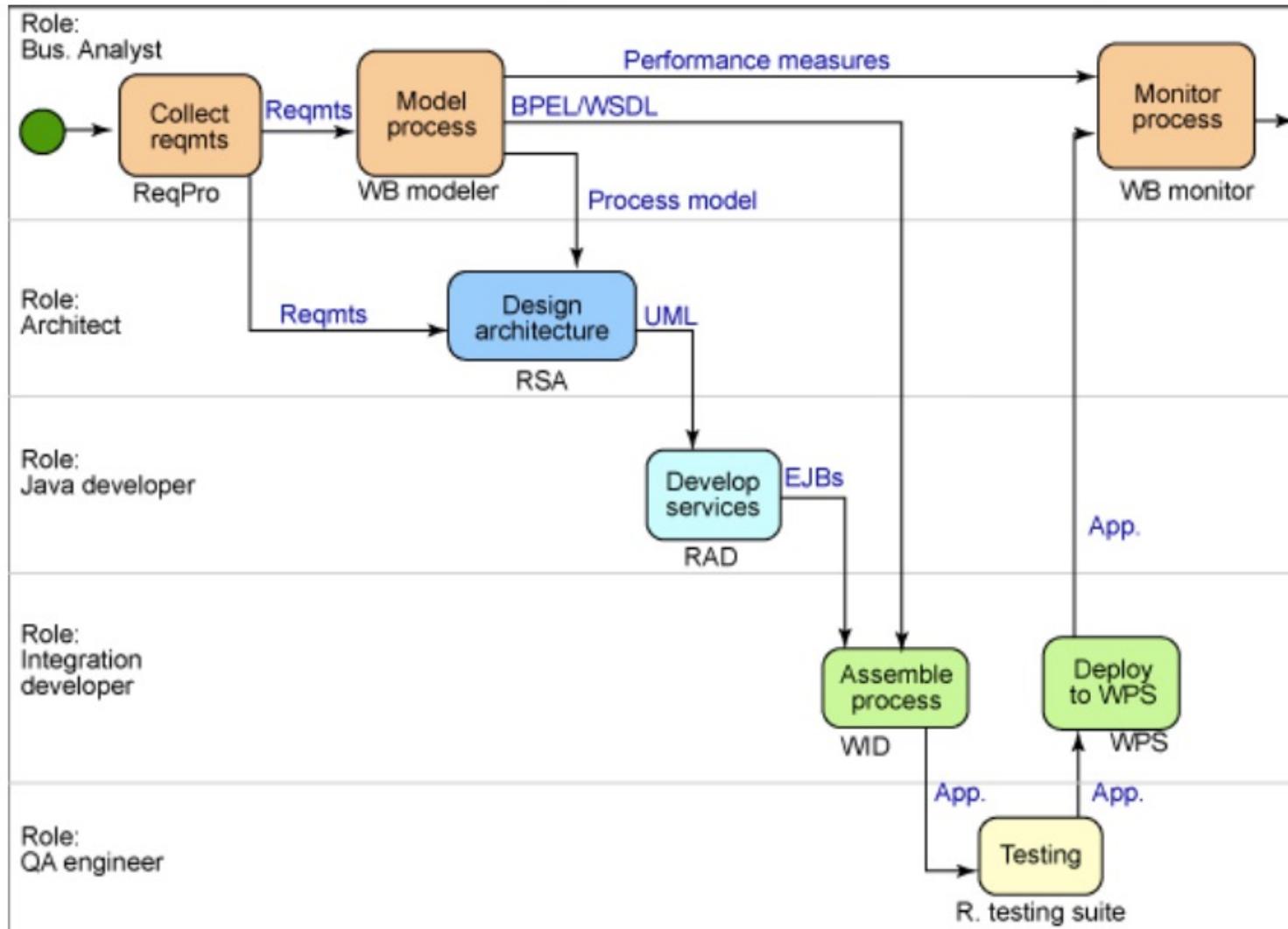
- Valutare il processo per garantirsi che determinati livelli di qualità del software
 - SCAMP: Standard CMMI Assesment Method for Process Improvement
 - CBA IPI: CMM-based Appraisal for Internal Process Improvement
 - SPICE: ISO/IEC15504
 - ISO 9001:2000 for software

Modello di Processo a cascata

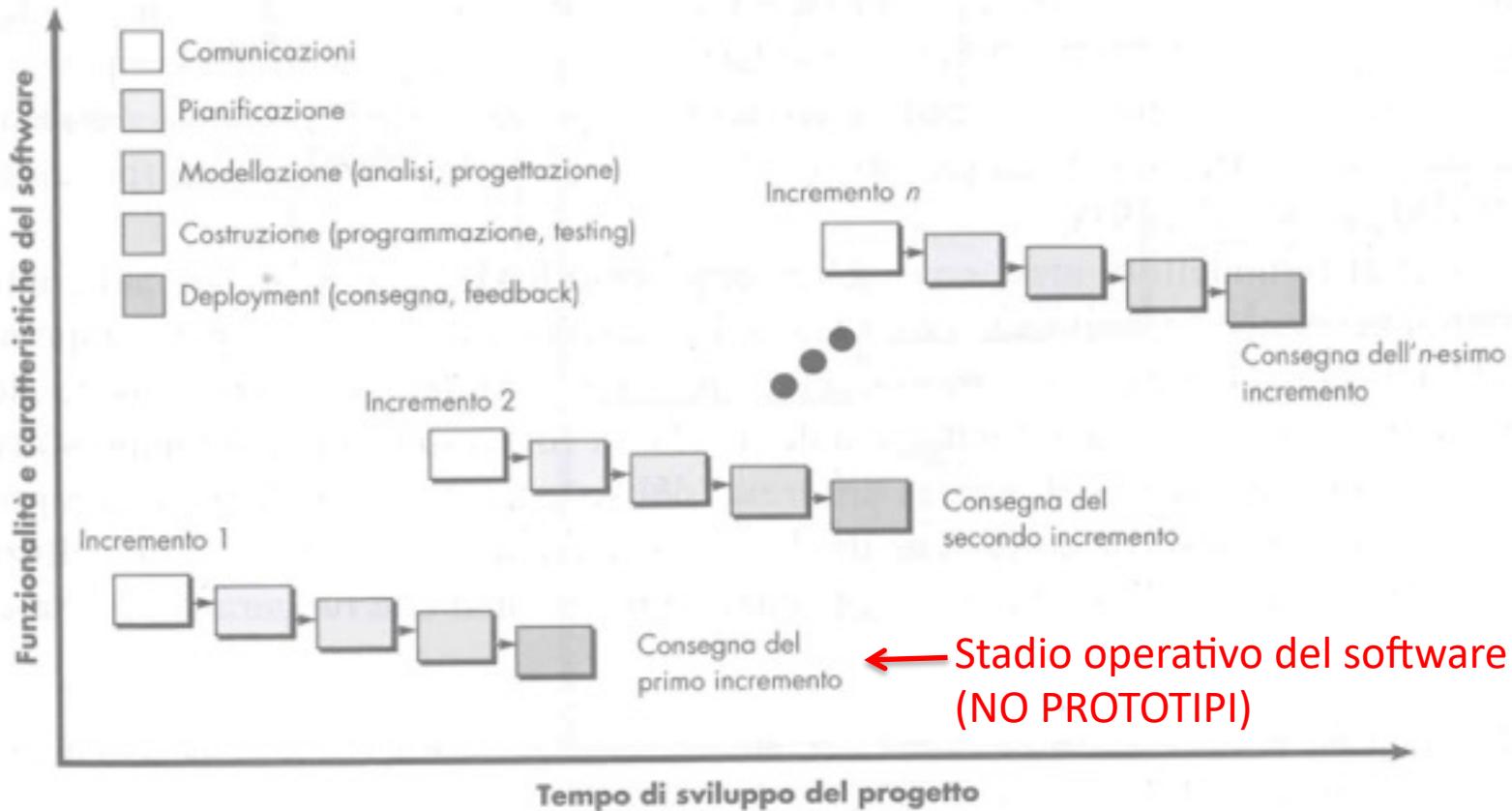
- Requisiti noti e il lavoro può procedere in maniera lineare
 - adattamenti o estensioni ben definite ad un sistema esistente
- Modello Waterfall (classico)



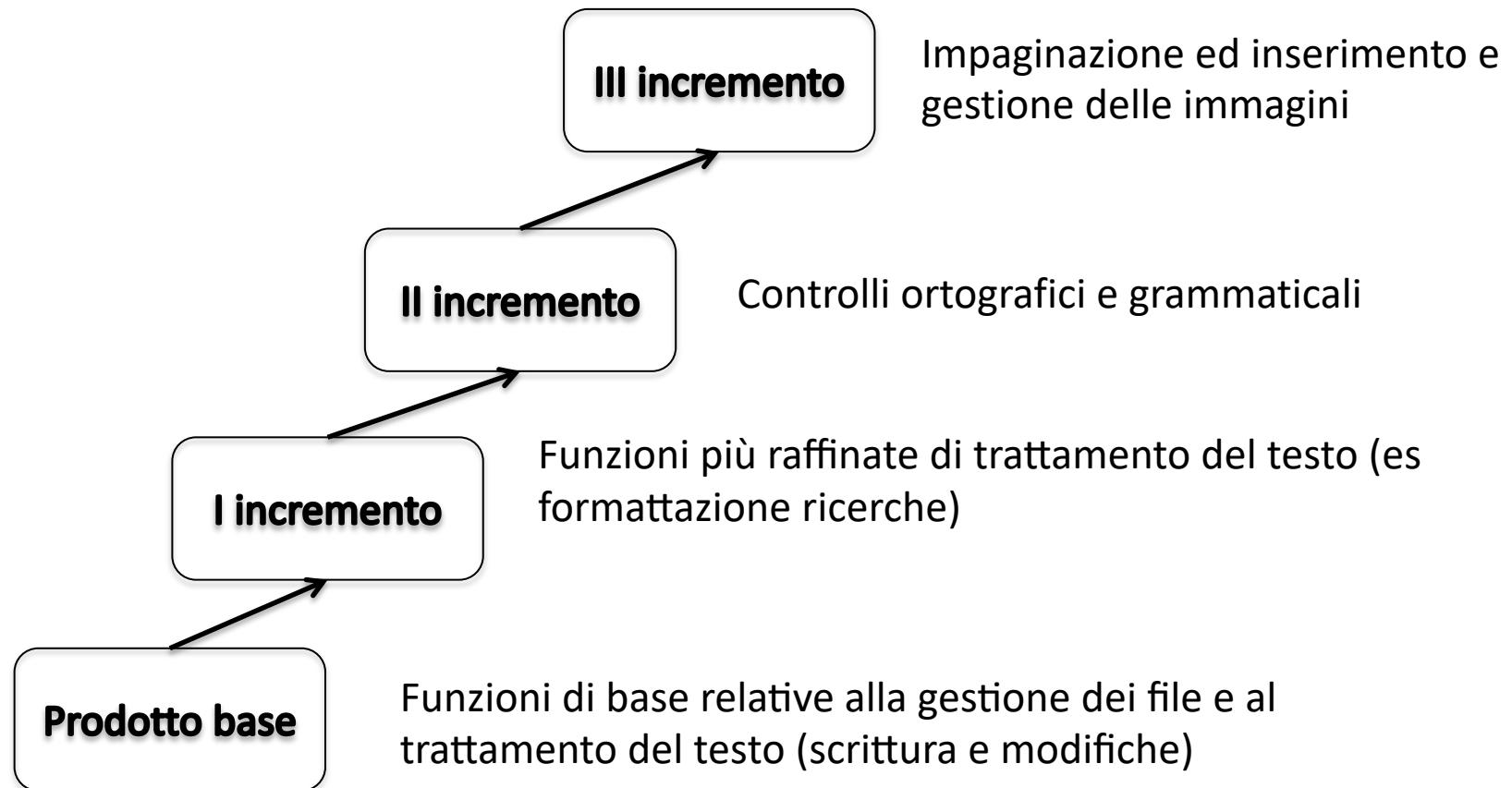
Esempio processo a cascata (IBM WebSphere)



Modello di processo incrementale

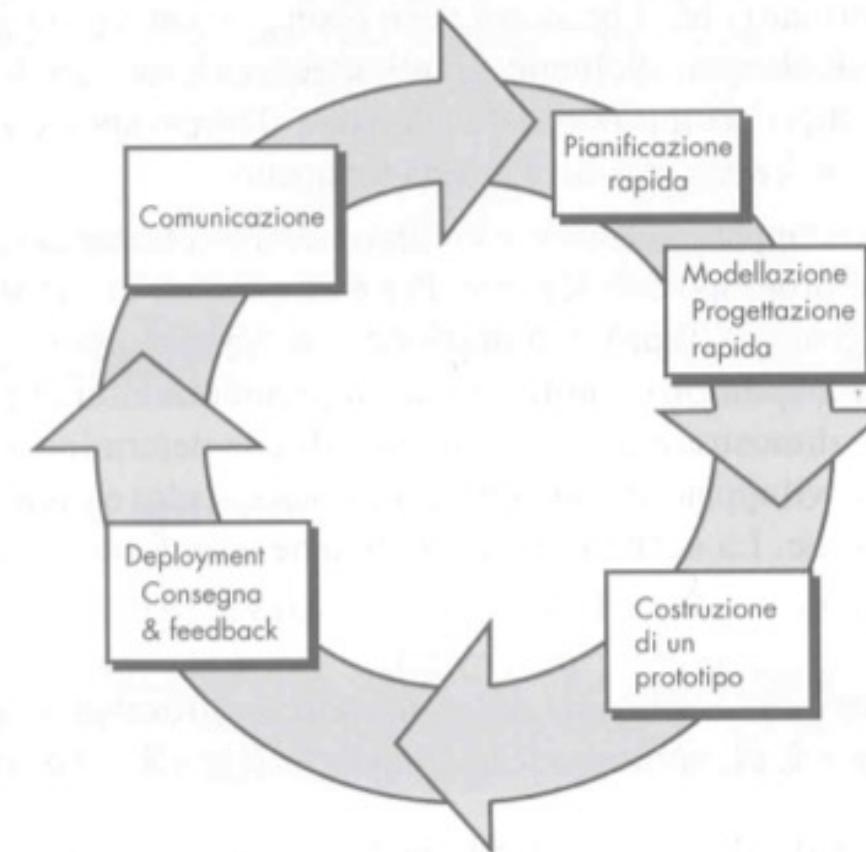


Es programma di elaborazione di testi



Modello a prototipi

Quando il cliente ha un bisogno legittimo ma non riesce a fornire indicazioni precise sui dettagli, il primo passo consiste nello sviluppo di un prototipo



Prototipi (def)

- Def ISO 13407
 - una rappresentazione di un prodotto o di un sistema, o di una sua parte, che, anche se in qualche modo limitata, può essere utilizzata a scopo di **valutazione***
- un prototipo non deve necessariamente essere un sistema funzionante, spesso può essere utile anche un semplice modello “finto” (mock-up)

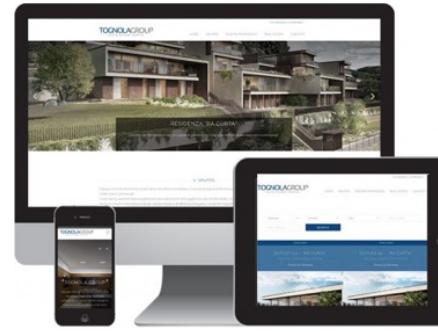
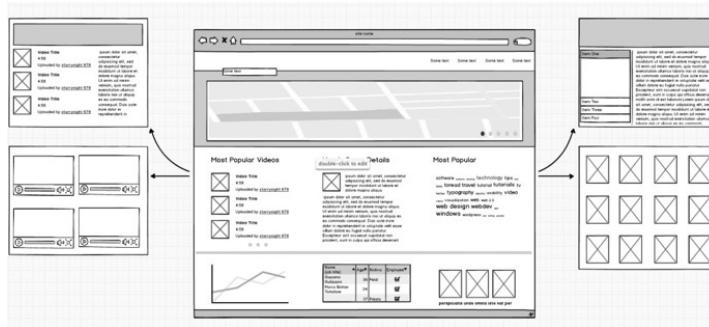
Jeff Hawkins, l'inventore del Palm Pilot, il primo PDA di successo, inizialmente tenne con sè un modellino in legno dello strumento, ovviamente non funzionante, fingendo di tanto in tanto di inserirvi o di leggervi delle informazioni



Che fine fa il prototipo?

- Il prototipo è un sistema preliminare
 - Spesso non utilizzabile, lento, troppo esteso, difficile da utilizzare
- Meglio ricominciare da capo
- Rischi:
 - Il cliente ha per le mani una versione del software che sembra essere quella finale
 - Gli sviluppatori scendono a compromessi per sviluppare rapidamente il prototipo
 - Qualità scadente

Wireframe vs Mock-up



- **Wireframe:** il focus è nella **user experience design** (trovare la migliore user experience).
 - Crea una bozza grafica fatta su un foglio di carta fatto a matita oppure utilizzando software specifici (es. Balsamiq)
 - Rappresenta la base dell'idea (spesso in bianco e nero)
- **Mock-up:** rappresenta la fase successiva in cui gli elementi del design entrano in gioco (colore, caratteri, immagini, etc) – si hanno più dettagli per dimostrare il **look and feel** (quello che sarà il prodotto).

Modello a spirale

Abbina la natura iterativa delle prototipizzazioni e gli aspetti controllati e sistematici del modello a cascata

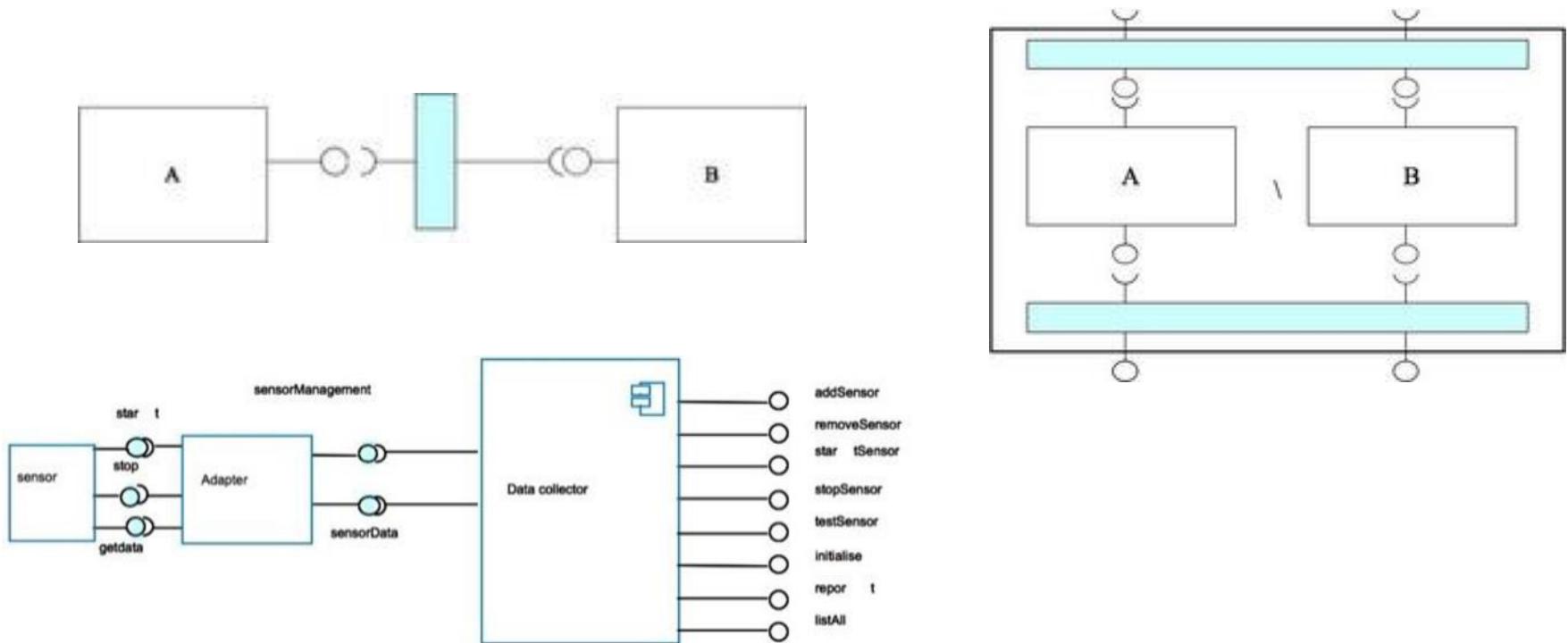


Model-Driven Development

- Modello, rappresentazione del sistema
 - Una forma particolare di prototipo del sistema
- Creazione di un modello di alto livello che evolve con la progettazione e lo sviluppo
 - Modello come guida per lo sviluppo
 - Strumenti per la generazione automatica del codice e dei test case
- Model-Driven Architecture (OMG)
 - Uso della notazione UML

Sviluppo a componenti

- Uso di componenti software con funzionalità mirate con interfacce ben definite



Composizione dei componenti

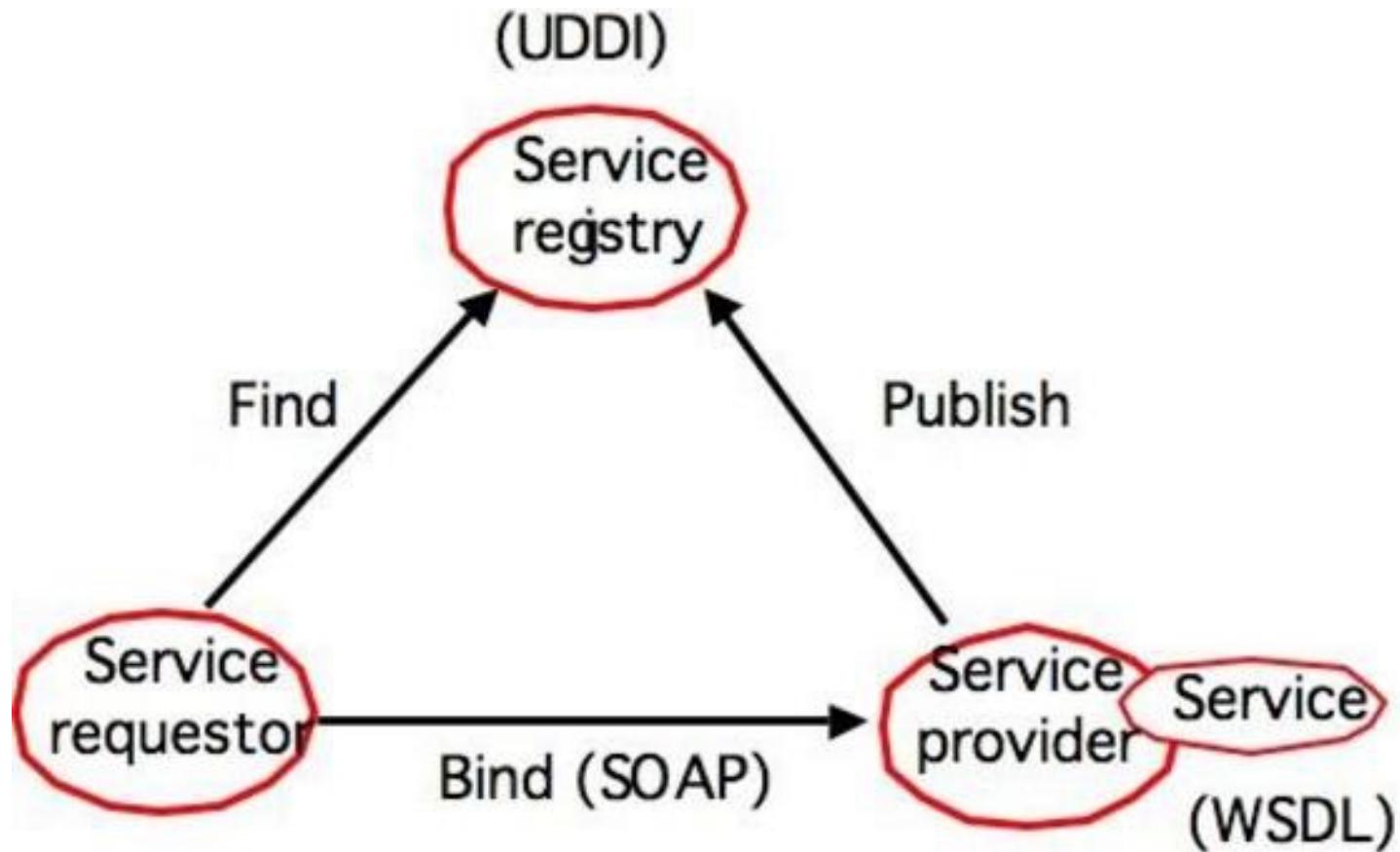
- I componenti più semplici risultano essere i più riusabili ma non riescono a risolvere problemi complessi
- Una composizione di componenti può raggiungere requisiti più specifici e complessi
 - è un processo che consente di ottenere un sistema attraverso l'assemblaggio di componenti
 - Spesso è necessario scrivere del codice integratore (glue code) per:
 - Rendere compatibili i formati degli input e degli output dei vari componenti
 - Coordinare i vari componenti

Valutazione dei componenti (esempio)

- Nel 1996, il razzo Ariane 5 (ESA) si distrusse durante il primo test di volo poichè il sistema ne perse il controllo dopo 37 secondi
 - un componente riusato dal software di Navigazione Inerziale sviluppato per Ariane 4
 - una eccezione di overflow non gestita, la quale però non poteva verificarsi nell'Ariane 4, a causa della ridotta potenza rispetto al 5



Architetture Orientate ai Servizi (SOA)



Metodologie Agile

- Per **metodologia agile** (o leggera) o **metodo agile** si intende un particolare metodo per lo sviluppo del software che coinvolge quanto più possibile il committente, ottenendo in tal modo un'elevata reattività alle sue richieste
- Esistono un certo numero metodologie Agile
 - Agile alliance, organizzazione no-profit creata allo scopo di diffonderle
 - <http://www.agilealliance.org>

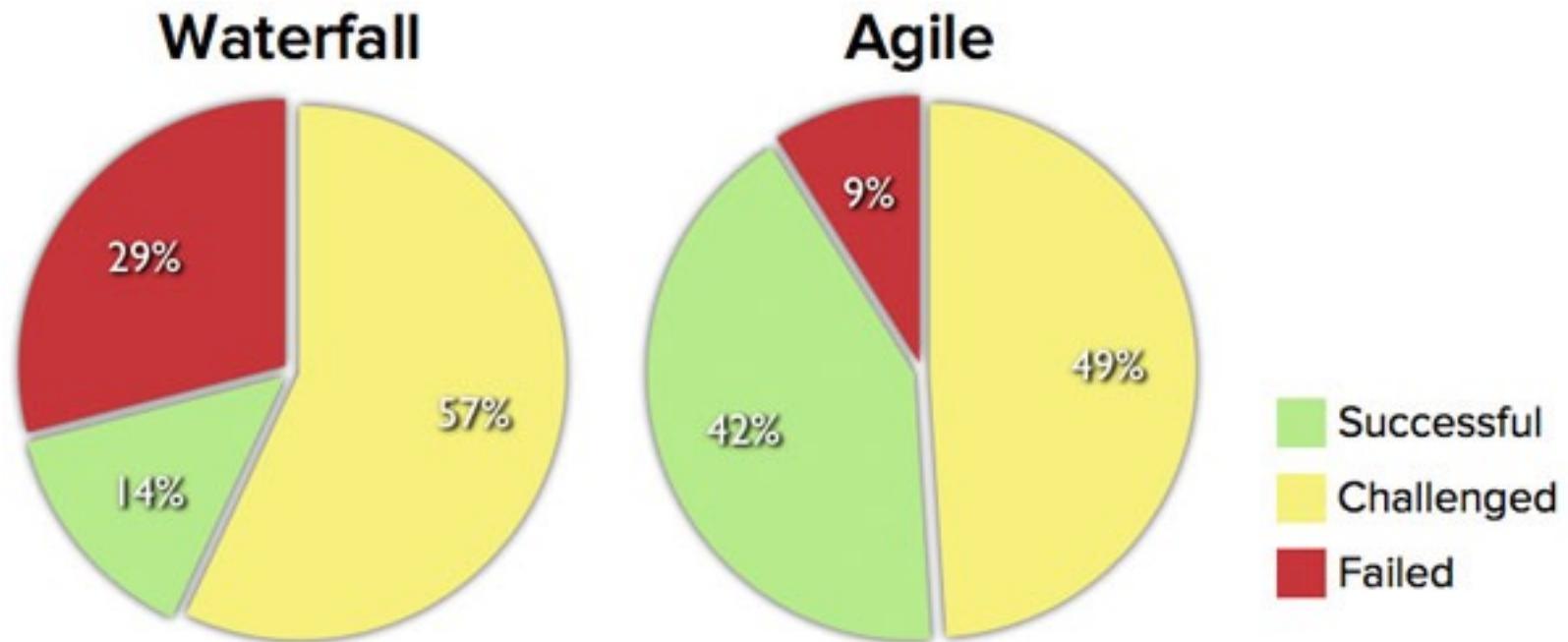




#113 - "AGILE DEVELOPMENT, EXAPLAINED" - BY SALVATORE IOVENE, FEB. 21ST 2009

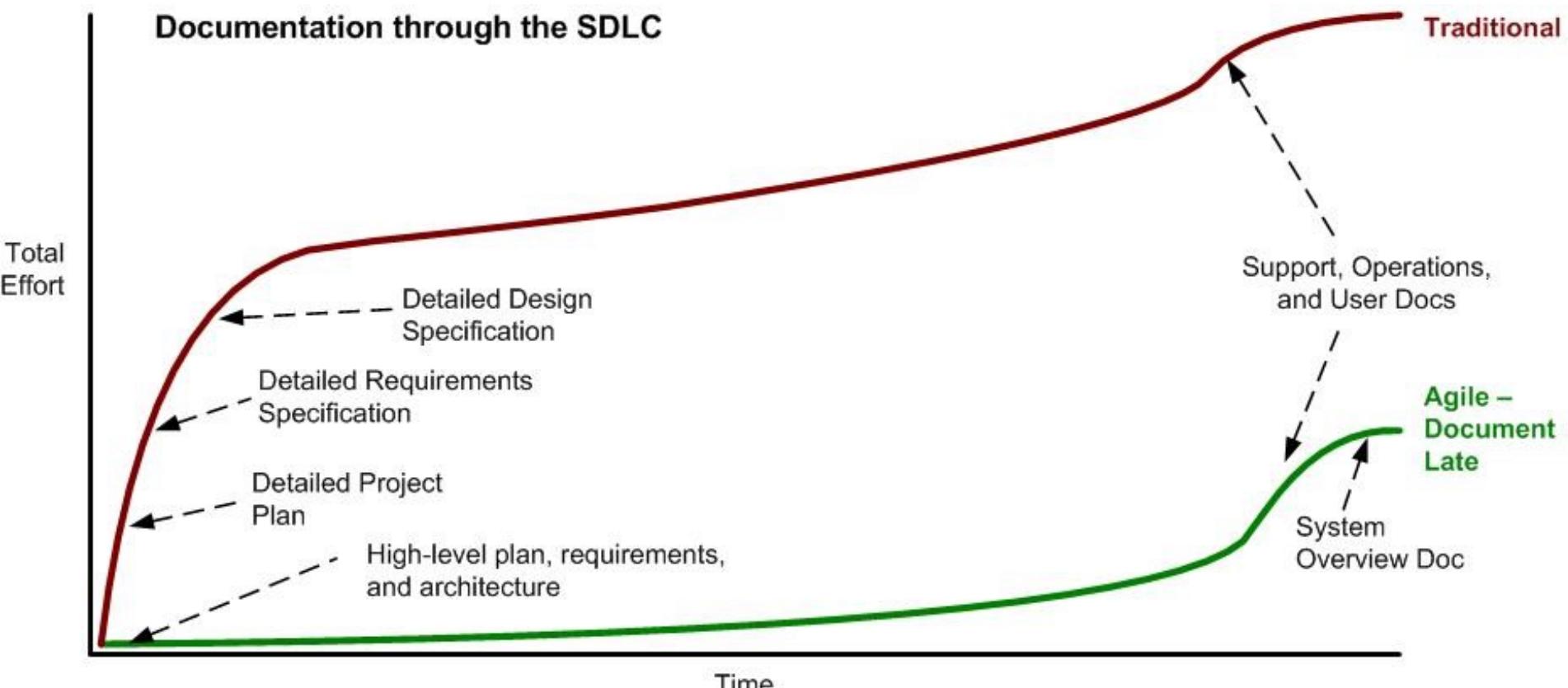
[HTTP://WWW.GEEKHEROCOMIC.COM/](http://www.geekherocomic.com/)

Agile vs Waterfall



Source: The CHAOS Manifesto, The Standish Group, 2012.

Documentazione



Copyright 2006-2014 Scott W. Ambler

Processi Agili

- Guidati dalla descrizione del cliente di che cosa gli serve: scenario
 - una descrizione di possibili interazione tra utenti e sistema (descrizione del funzionamento)
- Sviluppano software in maniera iterativa con forte enfasi sulle attività di costruzione
 - molteplici *incrementi software*
 - Adattamento ai cambiamenti

Agile vs IS

Non si deve pensare erroneamente che l'agilità dia la licenza di improvvisare le soluzioni. E' necessario un processo ed è fondamentale la disciplina

Non si tratta di scegliere fra agilità e ingegneria del software (classica) ma di definire un approccio agile all'ingegneria del software

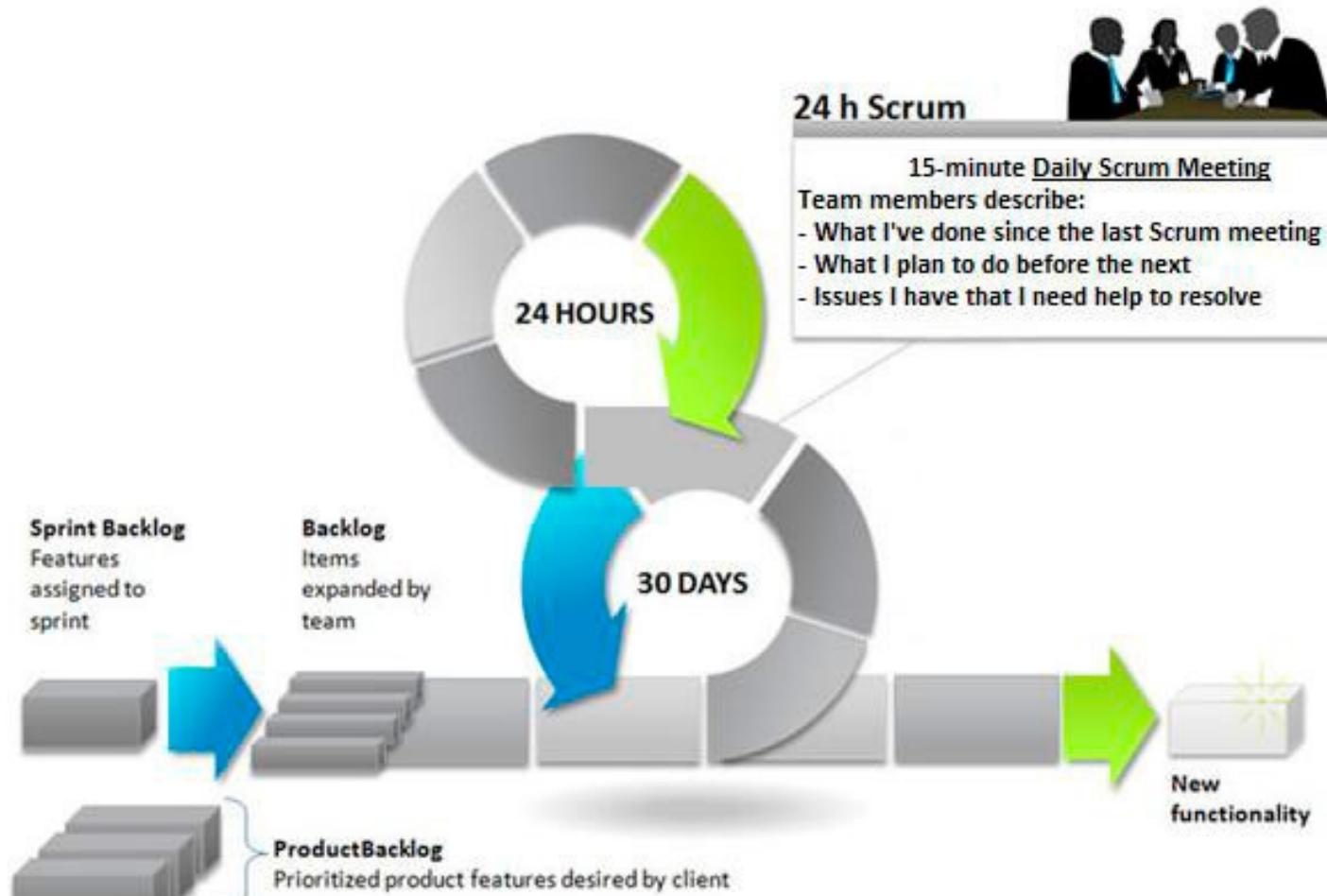
Molti concetti della metodologia agile non sono altro che adattamenti dei migliori concetti di ingegneria del software

Il modello Scrum (mischia)

- Insieme di **pratiche e regole** che permettono di ridurre l'**overhead** organizzativo
 - Piccoli team di lavoro organizzati
 - Produzione di frequenti incrementi software
 - Costante attività di testing e documentazione
 - Divide il progetto in blocchi rapidi di lavoro (**sprint**) alla fine dei quali viene consegnata una versione del sistema al cliente, indica come definire i dettagli del lavoro da fare nell'immediato futuro (**backlog**) per averne una definizione estesa, organizza riunioni giornaliere del team di sviluppo (**daily scrum**) per verificare cosa si è fatto e cosa si farà

Scrum

- Attività strutturali
 - Requisiti, analisi, design, evoluzione e consegna



Seminario 30 settembre

Kanban Simulation

Dr. York Rössler



Extreme Programming (XP)

- Kent Beck, Ward Cunningham e Ron Jeffries (1996)
- Approccio Object-Oriented per lo sviluppo
- **4 valori e 12 pratiche**
 - I 4 valori su cui si fonda questa metodologia sono:
semplicità, comunicazione, testing e coraggio
 - Le 12 "prassi consolidate" possono essere raggruppate in quattro aree fondamentali: **feedback a scala fine, processo continuo, comprensione condivisa, benessere dei programmatore**

Extreme Programming (XP)

- **Continuous integration**
 - L'integrazione delle modifiche nel sistema viene **fatta frequentemente**, in modo da limitare i conflitti che il nuovo codice (e il codice di test) dovesse dare
 - Procedura strettamente sequenziale: solo uno sviluppatore alla volta integra le sue modifiche con il sistema, testa e rilascia il codice
- **Refactoring o design improvement**
 - refactoring del codice cambiando l'architettura, in modo da renderlo più semplice e generico
- **Small releases**
 - consegna del software avviene tramite frequenti rilasci di funzionalità che creano del valore concreto

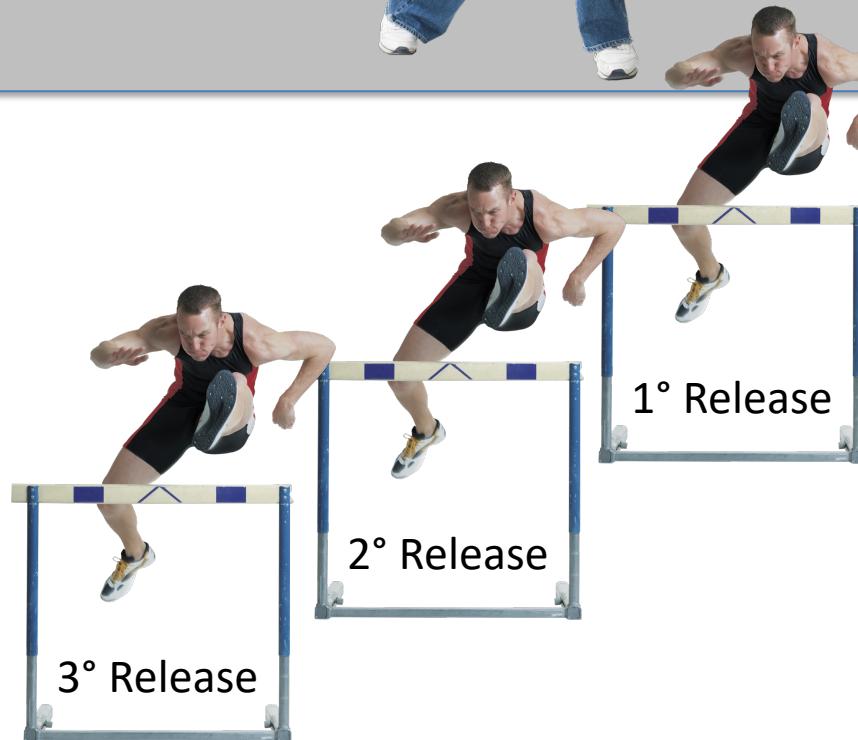
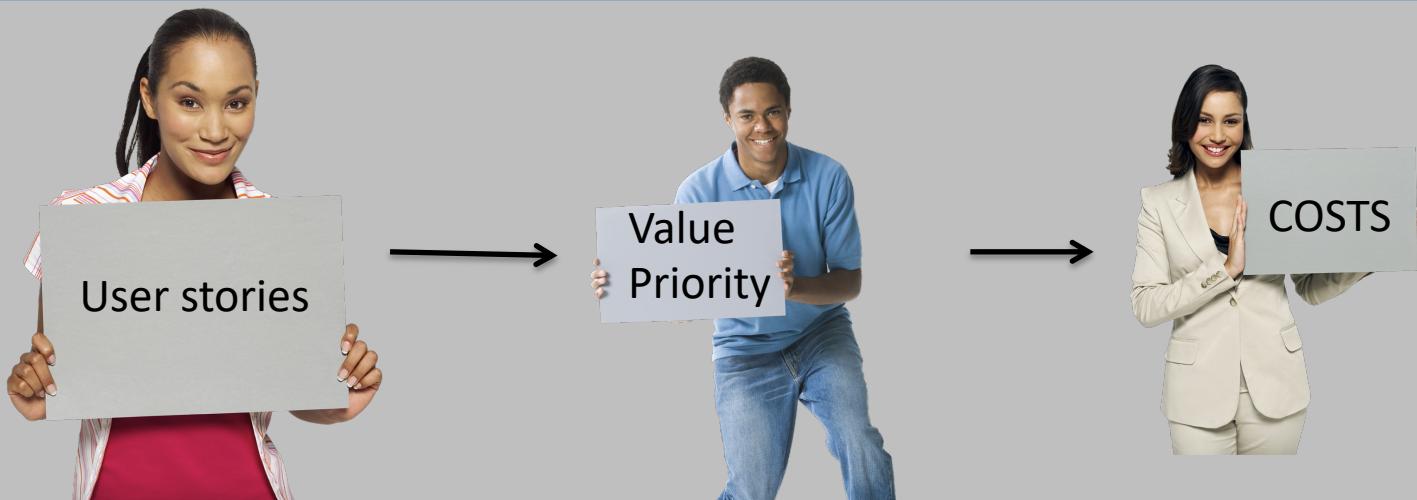
XP: Planning (1)

- Creazione di un insieme di **user story**
 - Descrivono le caratteristiche e le funzionalità del software (descritta dal cliente – scheda indice)
 - Il cliente assegna un valore (priorità) a ciascuna user story
 - I membri del team XP valutano ciascun user story e le assegnano un costo (settimane di sviluppo)
 - Se richiede più di tre settimane viene chiesto al cliente di dividerla in casi più semplici
 - Nuove user story possono essere scritte in qualsiasi momento
 - I clienti ed il team XP collaborano per decidere il modo in cui raggruppare le user story per la release successiva
 - Accordo di massima (user story, data di consegna etc)
 - Il team XP decide se: (1) implementare immediatamente tutte le user story; (2) implementare prima quelle a valore più alto; (3) implementare prima le user story più rischiose

User Story Examples

- A user wants access to the system, so they find a system administrator, who enters in the user's First Name, Last Name, Middle Initial, E-Mail Address, Username (unique), and Phone Number.
- **Risk: Low Cost:** 2 points

Front of Card	Back of Card
<p>1B</p> <p>As a student I want to purchase a parking pass so that I can drive to school</p> <p>Priority: AAA Should Estimate: 4</p>	<p><u>Confirmations:</u></p> <p>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.</p>



XP: Programmazione

- Sviluppo di Unit Test
 - Metteranno alla prova ognuna delle user story
 - Il programmatore si potrà quindi concentrare su ciò che deve essere implementato per passare tali test – non viene aggiunto nulla di superfluo
 - Pair programming (programmazione a coppie)
 - Due persone (con ruoli distinti) che collaborano alla stessa workstation per sviluppare il codice
 - Garantisce qualità e soluzione rapida a problemi
 - Il loro lavoro è poi integrato con quello degli altri (team di integrazione) – integrazione continua
 - Evita problemi di compatibilità e di interfacciamento

XP: Testing

- Utilizzo di strumenti di supporto per l'automazione degli unit test
 - Integration e validation test eseguiti quotidianamente (testing suite universale)
 - Indicazione continua dei progressi e individuazione più efficace delle problematiche
 - Acceptance test (o customer test) sono semplificati e si concentrano sulle funzioni e le caratteristiche globali del sistema

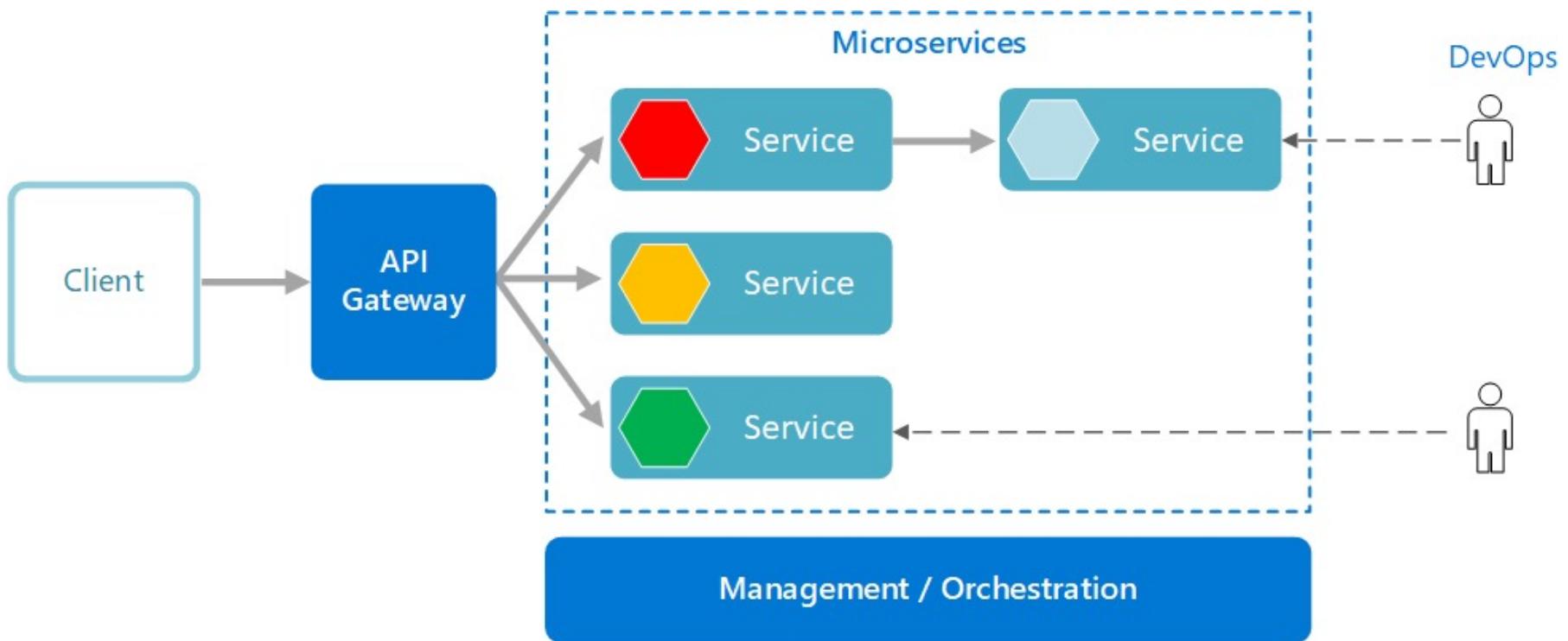
DevOps

- In realtà strutturate i componenti software vengono installati in infrastrutture grandi e complesse
- Bisogna coordinare il lavoro di chi scrive il software (**Development**) con quello di chi gestisce l'infrastruttura (**Operations**).
- Il termine **DevOps** (development+operations) indica una metodologia che consente di gestire in modo rapido ed efficiente questa interazione.
- Si basa su:
 - Virtualizzazione dei datacenter (o cloud)
 - Strumenti di automazione (IaC)
 - Metodologie di lavoro uniformi e condivise fra sviluppatori e sistemisti (di tipo agile).

Modularizzazione e microservizi

- La modularizzazione è essenziale nello sviluppo software
- L'architettura che si sta più diffondendo è quella a microservizi (**Microservice Architecture o MA**)
 - E' adatta per lo sviluppo di **grandi applicazioni** che hanno la necessità di scalare ed evolversi velocemente sfruttando soprattutto il **cloud**
 - Un'evoluzione «light» delle architetture SOA (Service Oriented Application)
 - Le applicazioni che sono costituite da un certo numero di servizi indipendenti e distribuiti, ciascuno incentrato su un particolare aspetto, che comunicano tra loro per realizzare servizi più complessi.

Esempio MA





Seminario 8 ottobre

Sviluppo Agile – user stories

Fincons Group è uno dei **principali player internazionali** che operano nella consulenza di business e nella system integration a supporto della **trasformazione tecnologica e digitale** di aziende leader nei loro settori di mercato.