

ALM and DevOps



Roberto Ajolfi

Senior Consultant @icubedsrl

Roberto.ajolfi@icubed.it



Software Lifecycle

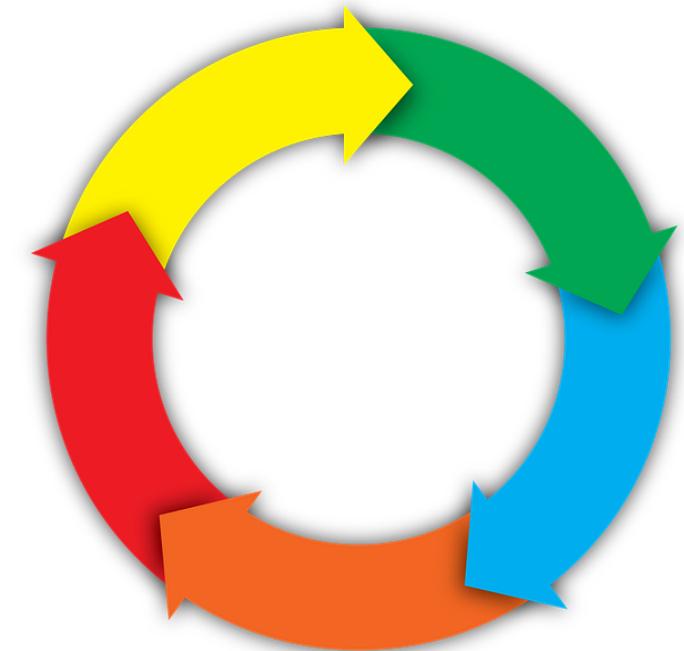
E' definito come «una serie di fasi che caratterizzano il corso dell'esistenza di un prodotto software».

E' il processo che definisce il ciclo di vita di un sistema informativo, dalla sua nascita, alla dismissione in quanto sistema obsoleto.

E' composto da diverse fasi, ciascuna importante quanto le altre, e quindi meritevoli di una considerazione che **spesso viene meno**; soprattutto dal punto di vista del tempo a loro dedicato.

Si parte con il concetto di «Conception», seguono numerose fasi operative «intermedie», per arrivare alla fase finale che solitamente è detta «Death» o «Retirement» del sistema.

La definizione delle fasi intermedie è la parte del Lifecycle di un sistema complesso, che cambia a seconda del «**Software Development Process Model**» utilizzato.



Analogia per la costruzione di un software

Possiamo paragonare la realizzazione di un Sistema Informativo complesso alla costruzione di una casa

Molte sono le fasi in comune tra le due attività.

A pensarci bene, la costruzione di un software è a tutti gli effetti un «processo manifatturiero» che coinvolge:

- diversi differenti fasi operative e di pianificazione
- diversi differenti figure professionali con competenze diverse, e tra loro complementari



Costruzione: le fasi del processo (1)

Conception:

- "Voglio costruire una casa dove vivere e crescere una famiglia»

Requirements Analysis:

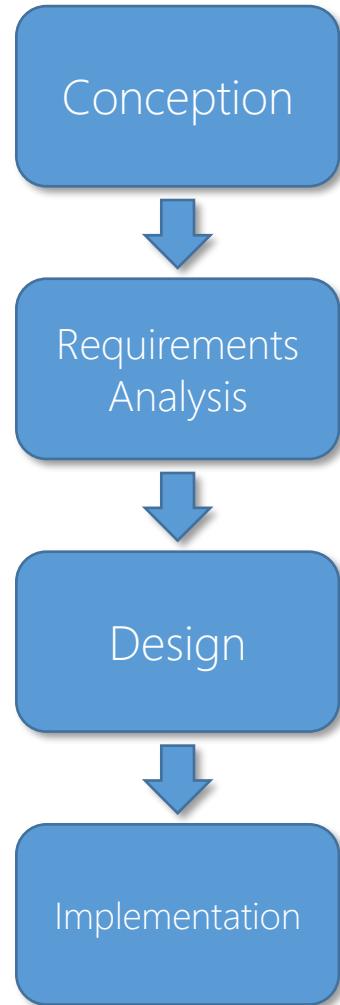
- «Che tipo di casa voglio?»
- «Quante stanze da letto e quanti bagni?»
- «Ci sarà una cantina o un garage?»
- «Quale sarà la metratura del giardino?»

Design:

- Scelta dell'architetto
- Disegno della struttura
- Selezione dell'impresa di costruzione

Implementation:

- Posa delle fondamenta
- Costruzione del tetto
- Costruzione dei muri esterni ed interni
- Impianti idraulici e elettrici



Costruzione: le fasi del processo (2)

Testing & Integration:

- Ispezioni sulla struttura
- Certificazione dell'impianto elettrico e idraulico
- Certificazione dell'abitabilità

Acceptance & Installation:

- Verifica da parte del proprietario
- Pianificazione del trasloco

Operation & Maintenance:

- Trasloco effettivo della famiglia
- Attività da parte del costruttore per sistemare i problemi di costruzione che possono essere occorsi durante il processo

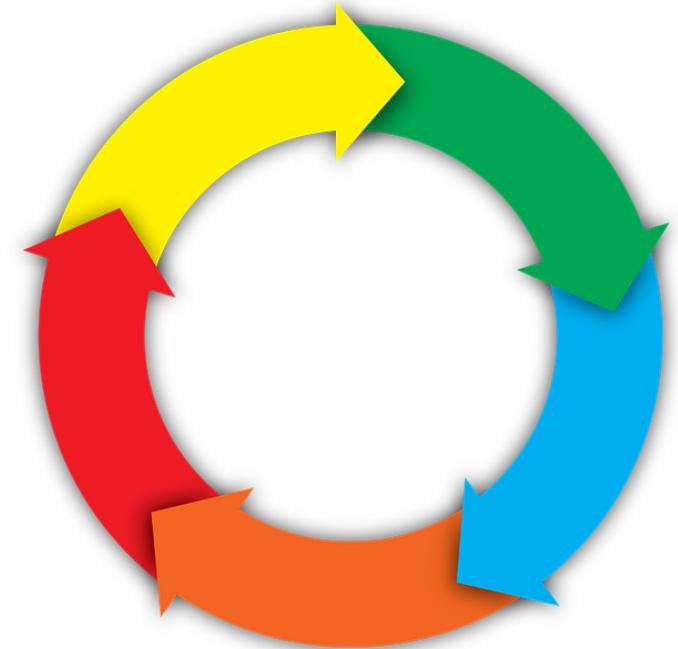
Retirement:

- Dopo molti anni di utilizzo, rinnovamento della costruzione...
- ...oppure demolizione della casa per avviare una nuova costruzione



Perchè innovare il nostro processo

Ci stiamo spostando da un approccio allo sviluppo del software di tipo ingegneristico, ad un approccio più manifatturiero, dove è necessario minimizzare le possibilità di ritardi oppure imprevisti.



Application LifeCycle Management

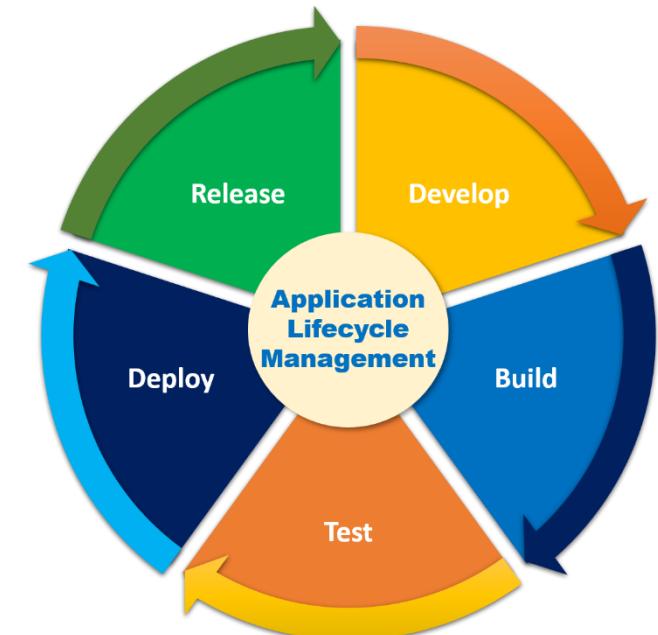
Ottimizzazione e gestione del ciclo di vita del software

E' il coordinamento delle attività del «Ciclo di Vita» dello sviluppo software:

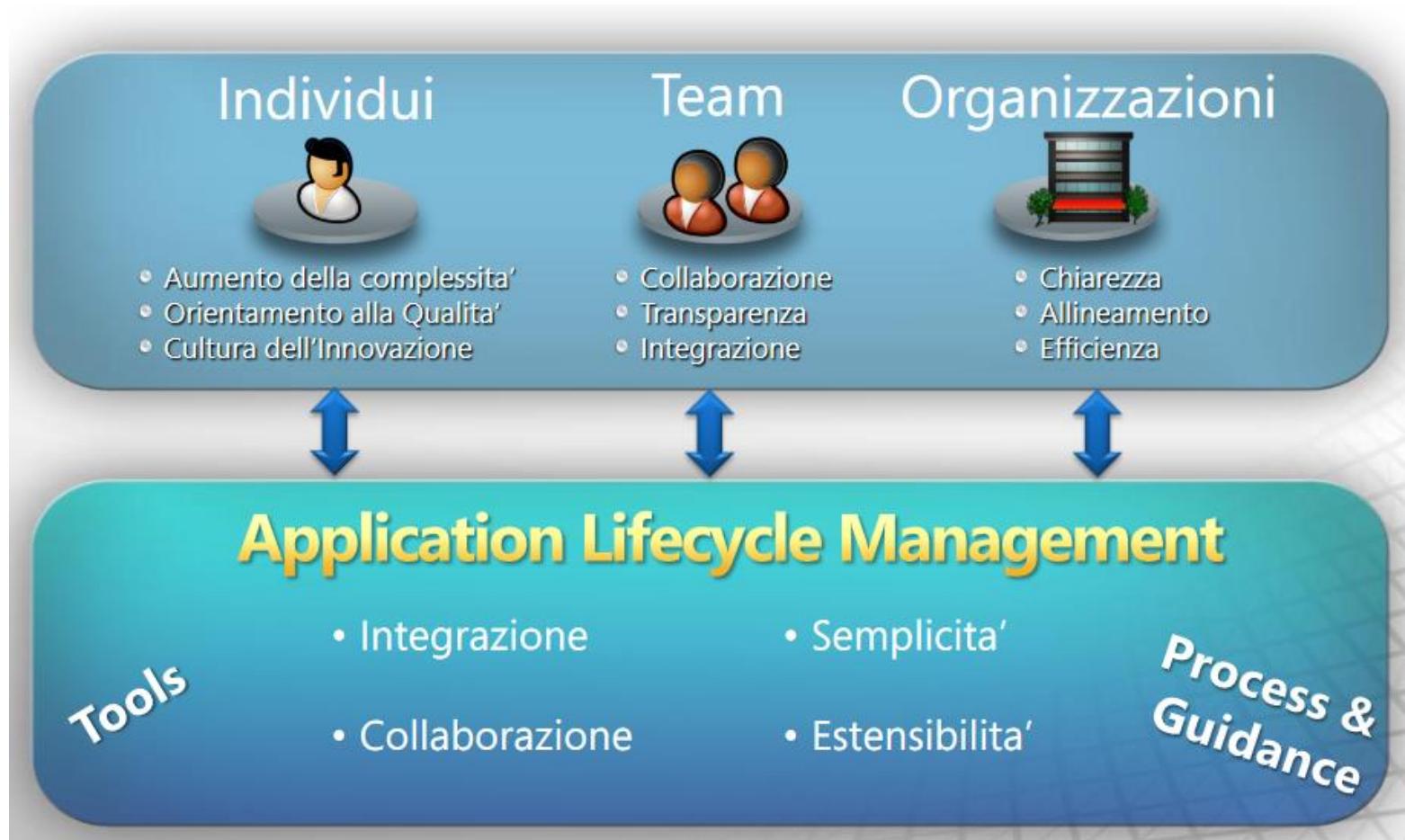
- Gestione dei requisiti
- Creazione di modelli
- Sviluppo del codice
- Build del software
- Test di prodotti

Il coordinamento delle attivita' è possibile attraverso:

- Un processo che le guidi in ogni sua fase
- Gestione delle relazioni tra gli attori e gli artefatti prodotti
- Real Time reporting per verificare l'andamento del lavoro



Application LifeCycle Management



Tante sono le parti coinvolte nel processo.

E' fondamentale che ciascuna faccia del suo meglio per arrivare all'obiettivo finale

I 6 ruoli nel team di sviluppo software (1)

Quando si approccia lo sviluppo di un sistema, sono 6 i ruoli (o le figure) che normalmente compongono il team:

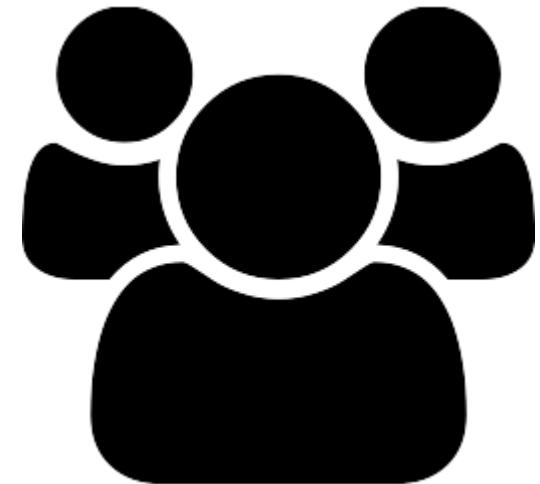
«**Product owner**» – il proprietario del prodotto finale. Può essere egli stesso l'utente utilizzatore (nella maggior parte dei casi non lo è) ma è certamente la persona che decide «cosa vuole» e svolge il ruolo principale perché è quello che trasmette in primis i concetti funzionali e beneficerà maggiormente dal prodotto realizzato



I 6 ruoli nel team di sviluppo software (1)

Quando si approccia lo sviluppo di un sistema, sono 6 i ruoli (o le figure) che normalmente compongono il team:

«**Business Analysts**» – si pongono come figura «di mezzo» tra gli owner del prodotto e il team tecnico. Hanno un ruolo fondamentale perché devono comprendere i bisogni del business e tradurre – per quanto possibile – tali bisogni in un linguaggio comprensibile ad un team di tecnici che spesso non conoscono affatto il business



I 6 ruoli nel team di sviluppo software (1)

Quando si approccia lo sviluppo di un sistema, sono 6 i ruoli (o le figure) che normalmente compongono il team:

«**Software/solution Architect**» - progetta i sistemi, identifica il modo in cui gli sviluppatori dovrebbero costruire il sistema e si pone come guida per tutti gli aspetti tecnici relativi al progetto. Il suo lavoro è importante perché serve a realizzare – almeno concettualmente – lo schema architettonico della soluzione, predisponendo il terreno su cui tutto l'apparato sarà costruito. Un cattivo lavoro fatto dall'architetto, spesso e volentieri, sfocia nel fallimento del progetto o comunque in aumenti di costi o ritardi sulla consegna



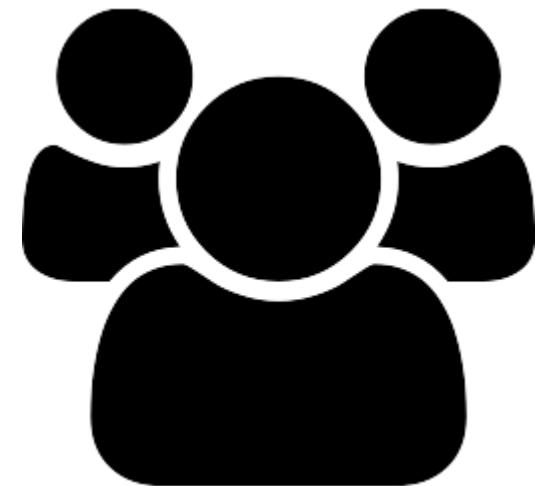
I 6 ruoli nel team di sviluppo software (2)

«Sviluppatori» – Ereditano i disegni funzionali e tecnici redatti dagli analisi e dall’architect, spesso aiutano loro stessi ad arricchire tali documenti, quindi li trasformano da semplice concetto - vivo solo su carta - in modelli software eseguibili e in soluzione funzionante. Inutile dire che è il ruolo più importante e percepibile, perché è l’unica figura della catena il cui lavoro è veramente tangibile ed evidente a tutti.



I 6 ruoli nel team di sviluppo software (2)

«Team QA» (Quality Assurance) – Detto anche team di «Testing», si assicura che il software scritto dal team di sviluppo corrisponda ai requisiti definiti in fase di analisi. Si assicura che il sistema risponda sia in termini di logica funzionale, sia in termini di performance e di sicurezza. Interagisce con gli sviluppatori in ogni momento, iterando il processo di costruzione/riparazione del sistema attraverso continui raffinamenti (rework) fino alla soddisfazione piena dei requisiti



I 6 ruoli nel team di sviluppo software (2)

«Team Operations» – Prende il software finito (e testato) e predisponde gli ambienti di esecuzione temporanea (Staging) e finali (Production) occupandosi della configurazione di quelli che sono i requisiti tecnici di esecuzione, e manutenendo la soluzione nel corso del tempo in termini di fruibilità da parte degli utenti finali



Il ciclo di vita del software

Quando **manca** un modello del ciclo di vita
(sia per la parte di coding che quella di fix)



Conseguenze

Non è possibile pianificare

Sviluppo caotico

Impossibile controllare
tempi e costi

Impossibile garantire la
qualità di ciò che si produce

Il ciclo di vita del software

Perchè avere un modello di ciclo di vita?

- Consente di definire i processi di sviluppo in maniera dettagliata
- Consente di pianificare/ottimizzare i tempi di sviluppo
- Consente di definire in maniera dettagliata le risorse necessarie
- Consente il lavoro delle singole risorse
- Consente di definire e controllare i costi e la qualità del prodotto realizzato

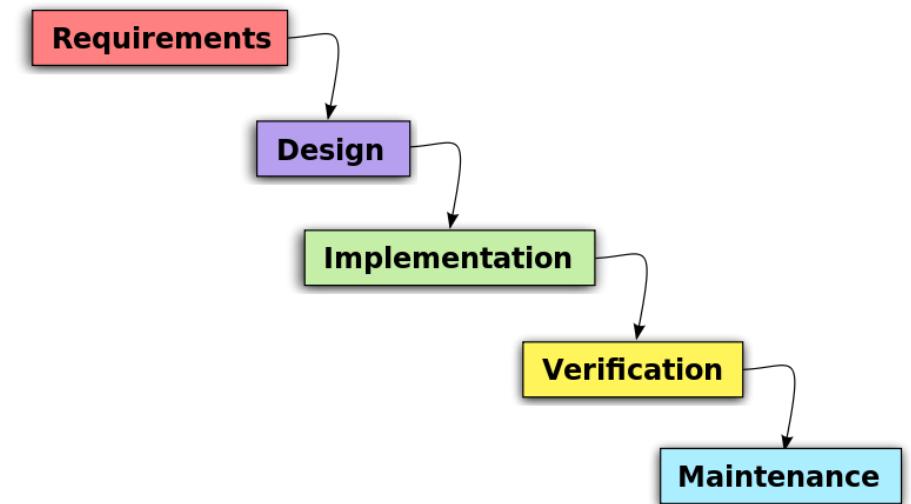
Il modello di processo «Waterfall»

La sequenza di costruzione del sistema è condotto in modo iterativo e incrementale, come formulato dal processo.

In alcuni approcci allo sviluppo del software - conosciuti collettivamente come modelli «waterfall» (a cascata) - i confini tra ciascuna fase sono destinati a essere abbastanza rigidi e sequenziali.

Il termine "cascata" è stato coniato per tali metodologie per significare che il progresso avanza sequenzialmente, in **una sola direzione**.

Una volta che l'analisi è stata completata il progetto inizia, ed è raro - e considerato fonte di errore – richiedere una modifica del modello di analisi o **cambiare i requisiti** per un problema di codifica.



«Waterfall»: vantaggi e svantaggi (1)

La sua definizione è da attribuirsi a Winston W. Royce, un computer scientist americano, in una pubblicazione del 1970.

Si tratta del modello di processo di sviluppo software più vecchio e largamente utilizzato.

E' particolarmente utile in situazioni in cui è fondamentale avere una stima anticipata del lavoro (e dei costi) dell'implementazione, ancora prima di iniziare le attività operative.

E' stato il modello "principe" che ha permesso per anni ad aziende di consulenza di proporre l'implementazione "chiavi in mano" di un prodotto software ad un particolare committente:

- Semplice da implementare e comprendere
- Utilizzato per anni e quindi ben conosciuto
- Semplice da gestire (perché sequenziale)
- L'allocazione delle risorse è ben definita
- E' l'ideale per lo sviluppo di piccoli processi



«Waterfall»: vantaggi e svantaggi (2)

Gli svantaggi della sua adozione sono maggiori dei vantaggi:

Sono necessari dei requisiti chiari e ben definiti fin dalle prime fasi

- Non sempre i requisiti sono chiari all'inizio (men che meno a tutte le persone)
- Alcune tematiche funzionali e tecniche possono cambiare i requisiti

Non è possibile avere un feedback durante la fase di implementazione

- I primi commenti da parte degli Stakeholders arrivano nella fase di Testing
- Non c'è possibilità di «rassicurare» Stakeholders sul fatto che le attività stanno procedendo nel migliore dei modi, e non ci saranno sorprese alla delivery

I problemi emergono quando ormai è troppo tardi

- Problemi tecnici, di codifica, che necessitano un fix prima della consegna
- Problemi di comprensione delle funzionalità, che richiedono una riscrittura
- Gli impatti di queste problematiche possono essere a volte devastanti

E' molto complesso stimare la durata delle implementazioni

- Non è possibile stimare la «velocity» del team

Non è possibile in alcun modo «parallelizzare» le attività

Esiste un utilizzo inefficiente delle risorse del team

- Alcune persone possono essere «scariche», altre troppo cariche di attività

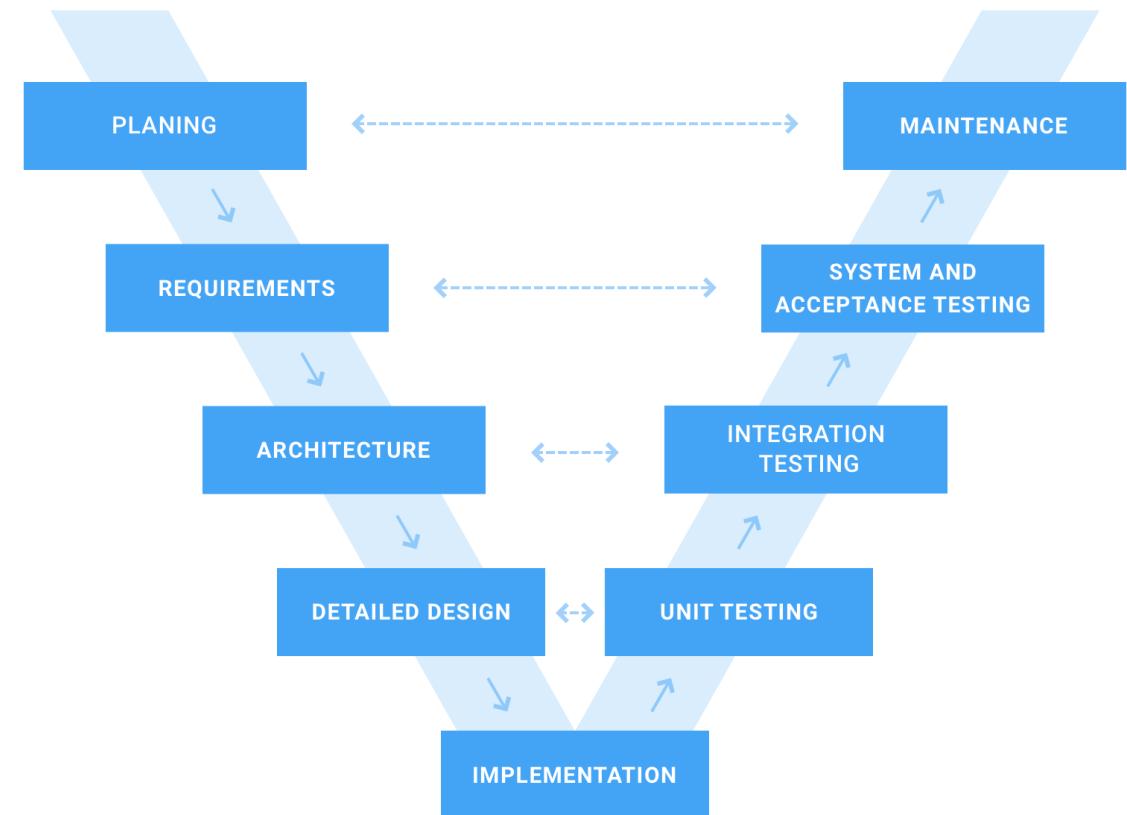


Una variante: il «V» Model

E' una alternativa al modello «Waterfall» e dello stesso incarna la stessa logica sequenziale (con tutto quello che ne consegue).

Presta tuttavia il fianco in misura minore agli errori che possono occorrere durante la fase implementativa:

- Esiste una fase iniziale di pianificazione detta «Business Case» o «Planning» che serve ad avere maggior focus sulla parte funzionale.
- La raccolta dei requisiti è seguita da due fasi di Design che si dividono in «Design dell'Architettura» e «Design di Dettaglio»
- Ciascuna fase di design trova una corrispondenza nel secondo ramo dello schema a «V» con una relativa fase di Testing, che dovrebbe permettere aumentare la qualità dell'artefatto realizzato nella fase di «Implementation»
- Nonostante gli sforzi, questo modello soffre di molti degli svantaggi del suo predecessore, e si mostra quindi poco adatto a **sistemi complessi** dove il numero e la complessità dei requisiti è elevato e quindi difficilmente controllabile.

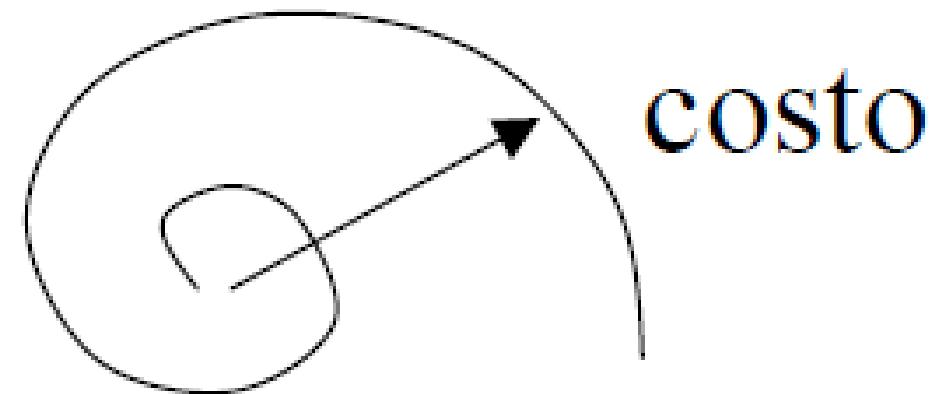


Il ciclo di vita del software

Un primo modello... «innovativo»!

Il modello a spirale!

- I processi vengono visti come un ciclo nel quale si iterano le attività di analisi, progetto, realizzazione/test e valutazione per successivi cicli
- Ad ogni ciclo, il costo (raggio della spirale) aumenta



Il ciclo di vita del software

Prodotti innovativi: qual'è il loro ciclo di vita?

- Incrementabilità diventa **fondamentale**
- Rilasci rapidi
- Fidelizzazione del cliente
- Spesso i requisiti non esistono
- Si creano multiple versioni di prova



Il ciclo di vita del software

Alcune considerazioni a carattere generale

- Molti progetti tendono a fallire!
- Spesso sono i progetti grandi a fallire rispetto a quelli piccoli
- Progetti grandi falliti, implicano spreco di risorse investite
- Spesso meno del 50% delle features richieste vengono prodotte

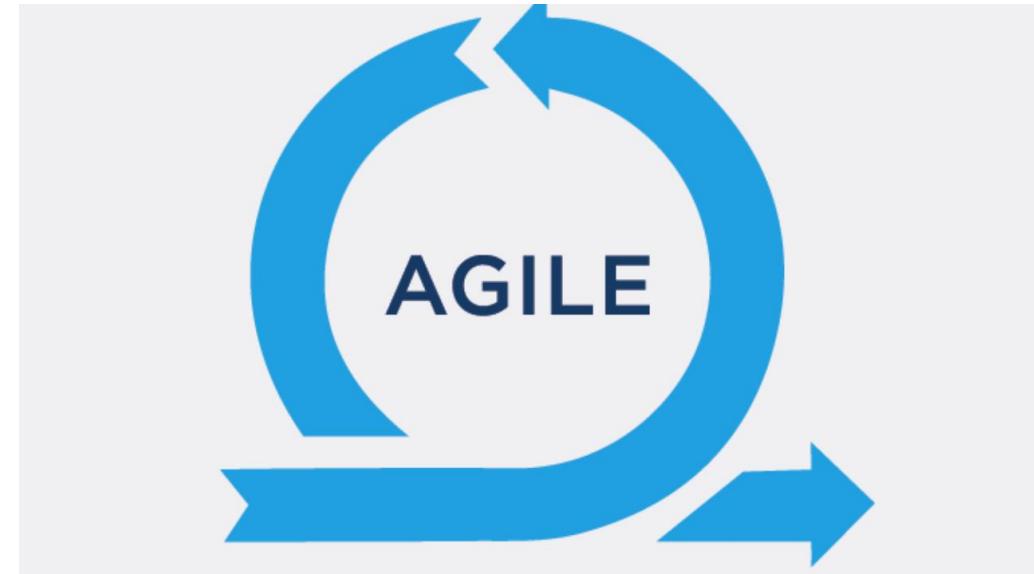


Il ciclo di vita del software

I modelli di sviluppo tradizionali hanno un alto overhead!

- L'idea... sviluppare in maniera iterativa...
- Rilasci rapidi...
- Continui...

Alla fine degli anni 90... nascono i modelli Agile!



Il ciclo di vita del software

Le metodologie storiche hanno un approccio di tipo PREDITTIVO

- Viene definito un percorso
- Viene seguito rigorosamente

Le metodologie agile hanno un approccio ADATTATIVO

- Rispondono al cambiamento



Il ciclo di vita del software

Il modello Agile si contrappone a:

- Nessuna metodologia di modello
(`Cowboy coding' ☺)
- Modello a cascata (Waterfall e V-Model): tradizionalista
- Modello iterativo: basato sul modello a cascata ma con rilasci più frequenti, fornendo avanzamenti incrementali o prototipi del sistema



Metodologia «Agile»

La proposta di modelli iterativi è variegata e ciascuno di essi presenta vantaggi e svantaggi. Ma è la **flessibilità** degli stessi che li sta facendo preferire a soluzioni più rigide e meno adatte ad un processo di progettazione moderno.

La metodologia «agile» descrive una serie di principi per lo sviluppo del software in base alle quali i requisiti e le soluzioni si evolvono attraverso lo sforzo di collaborazione di organizzazioni intra-funzionali di autoorganizzazione.

Promuove la **pianificazione adattabile**, lo sviluppo evolutivo, la consegna anticipata e il miglioramento continuo, e incoraggia una risposta rapida e flessibile al cambiamento.

Il termine «Agile» è stato adottato dagli autori del «Manifesto per lo sviluppo di software agile» («Agile Manifest») pubblicato nel 2001 da Kent Beck, Robert C. Martin, Martin Fowler e altri.



Manifesto Agile

Riassume in maniera semplificata i principi su cui si basa la metodologia Agile, mostrando quanto si possa arrivare ad un prodotto di qualità dando valore al dialogo e collaborazione tra le persone.

Rimarca come un software funzionante è più importante di qualsiasi documentazione.

Quanto è importante fondare il processo sulla collaborazione con il cliente finale e la sua esperienza

In ultimo - probabilmente la cosa più importante – enfatizza come è fondamentale essere in grado di «rispondere efficacemente al cambiamento», indirizzando le chieste funzionali dell'ultimo minuto senza per forza sconvolgere il processo di costruzione del sistema.

The Agile Manifesto

Individuals and interactions	over	Processes and Tools
Working Product	over	Comprehensive Documentation
Customer Collaboration	over	Contract Negotiation
Responding to change	over	Following a plan

That is, while there is value in the items on the right, we value the items on the left more.

www.agilemanifesto.org

I Principi su cui si basa Agile (1)

L'obiettivo è soddisfare il cliente consegnando continuamente versioni funzionanti del sistema

Accettare i cambiamenti in corsa

Consegnare software frequentemente, ogni due settimane o ogni due mesi (meglio 2 weeks)

Deve esistere una collaborazione tra il team tecnico e chi conosce il business

É fondamentale mantenere i partecipanti motivati

Preferire una comunicazione faccia a faccia rispetto la scrittura di documenti



I Principi su cui si basa Agile (2)

La misurazione del progresso di un progetto si misura con dell'avanzamento delle funzioni del sistema

Il processo Agile richiede che il progetto sia sostenibile dal team:
evitare lavoro overtime!

Dare molta importanza all'eccellenza tecnica, sempre

Mantenere il design il più semplice possibile

Le migliori architetture derivano da team auto organizzati piuttosto
che supervisionati

Ad intervalli regolari il team si ferma per valutare la qualità del suo
lavoro e cerca di porre rimedio agli errori commessi



Strumenti e pratiche «agili»

Una metodologia flessibile come quella descritta richiede delle pratiche ben specifiche, che permettano la sua implementazione:

- Cercare di realizzare piccoli team di persone, tutte «alla pari»
- Organizzare meeting frequenti e periodici con il committente
- Mantenere il progetto «code-centrico»
- Documentazione: solo su richiesta e se necessaria
- Raccolta dei requisiti sempre ad «alto livello»
- Favorire la collaborazione del committente con il team
- Basare lo sviluppo sul concetto che il «refactor» è inevitabile
- Applicare uno sviluppo orientato ai test di «unità» e «accettazione»
- Automatizzare il test, evitando di applicarlo «on demand»



Il ciclo di vita del software

Alcuni passi fondamentali della metodologia Agile

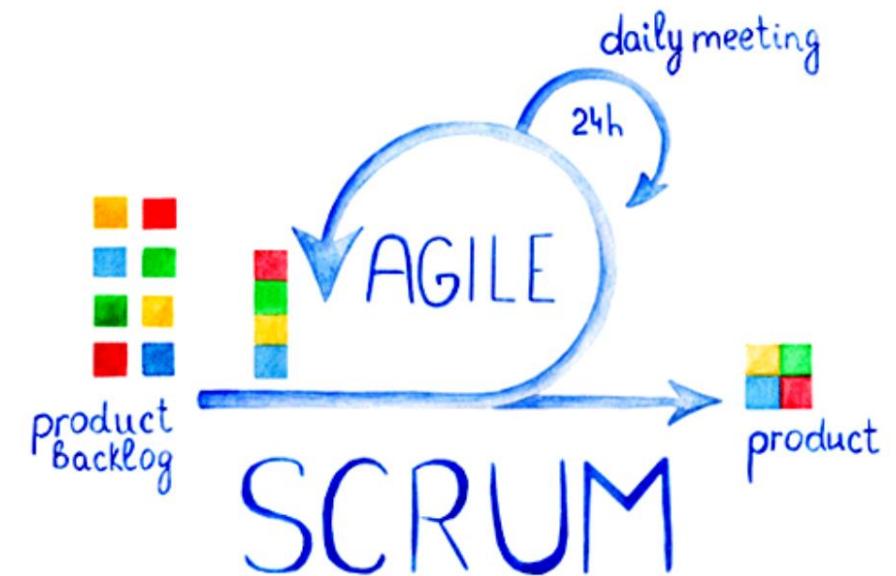
- Il lavoro è svolto da persone. Se il team è coeso, si produce meglio!
- Una visione più ampia di un problema, lo rende facilmente risolvibile
- Più persone conoscono il problema, più facilmente sarà risolvibile



Il ciclo di vita del software

Ogni metodo descrive un diverso processo, ma condividono gli stessi principi

- SCRUM (Sutherland)
- Extreme Programming (Ken Beck 1999)
- Lean Software Development (Poppendieck)
- Feature Driven Development (De Luca & Coad)
- Crystal (Cockburn)
- DSDM (Dynamic System Development Method)



Il ciclo di vita del software

I metodi Agile – Alcuni Principi

- Coinvolgimento del cliente
- Consegnare Incrementale
- Persone, non processi
- Accettare i cambiamenti
- Mantenere la semplicità



Il ciclo di vita del software

Ma il modello? Agile non vuole dire «senza modello»

La modellazione deve esistere... è solo Agile!

Lo scopo dei modelli e della modellazione è supportare

- la comprensione
- la comunicazione

Non modellare tutto!

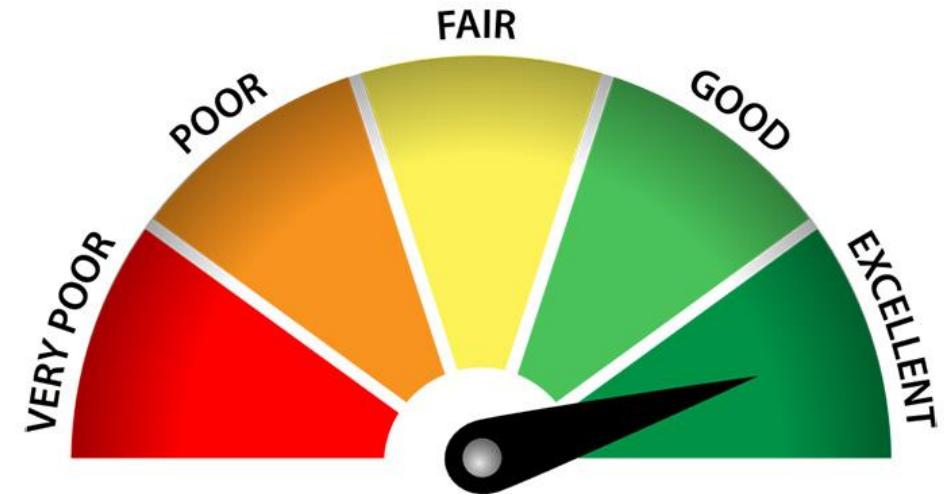
- usare gli strumenti più semplici possibili;
- non modellare da soli;
- creare modelli in parallelo;
- sapere che tutti i modelli sono incompleti e imprecisi.



Il ciclo di vita del software

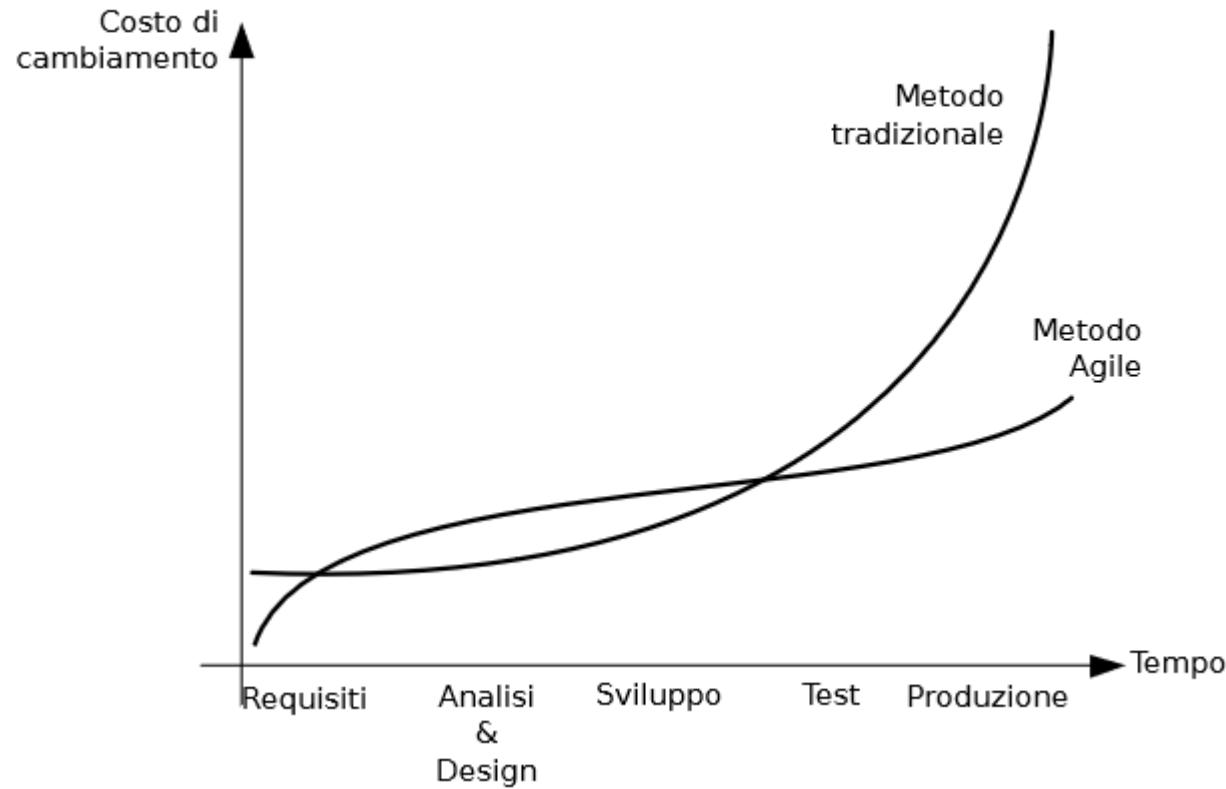
I metodi Agile

- Tempi di consegna facilmente rispettati
- Rapida risposta dell'utente
- Rischi attenuati
- Cicli più rapidi = maggiore controllo su quanto prodotto
- Alta produttività
- Alta qualità
- Maggiore possibilità di successo per il progetto (clienti soddisfatti)



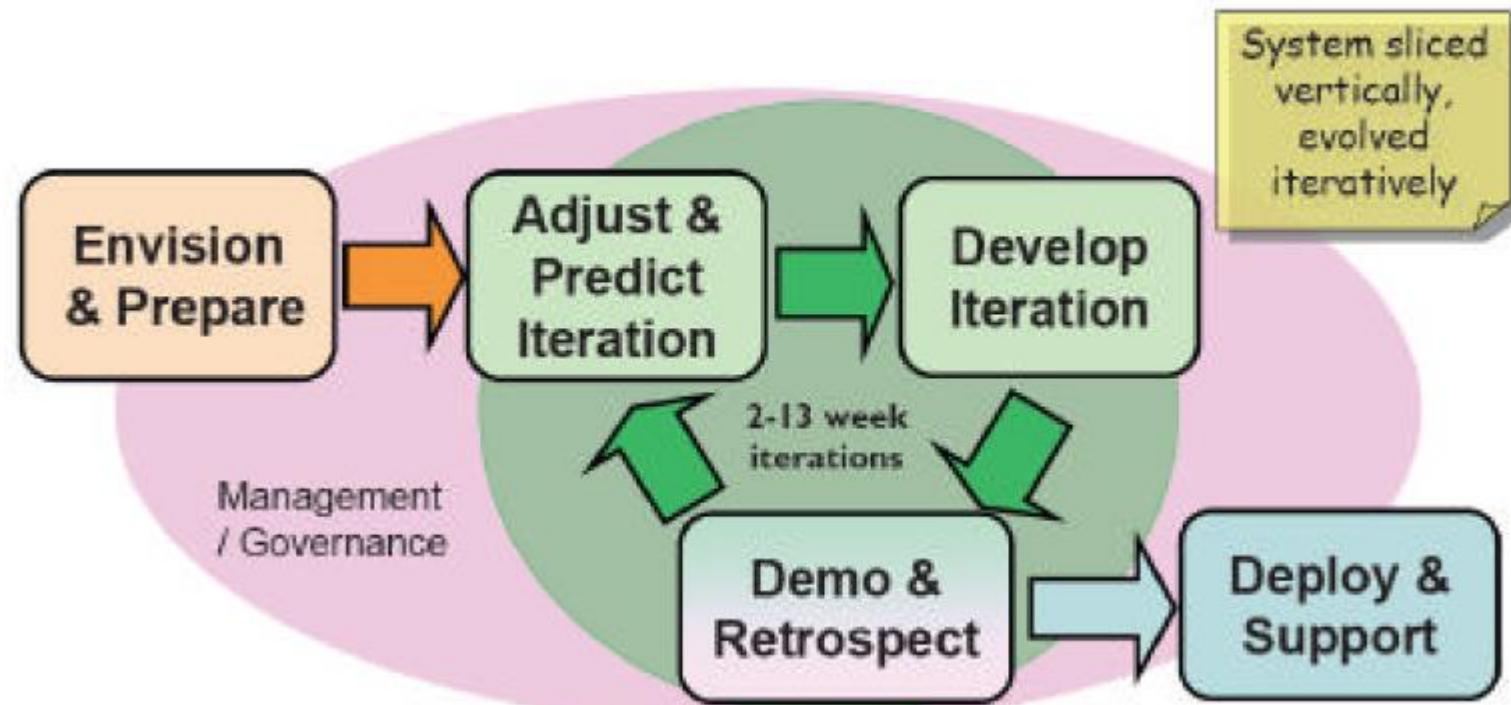
Il ciclo di vita del software

Confronto tra costo al cambiamento



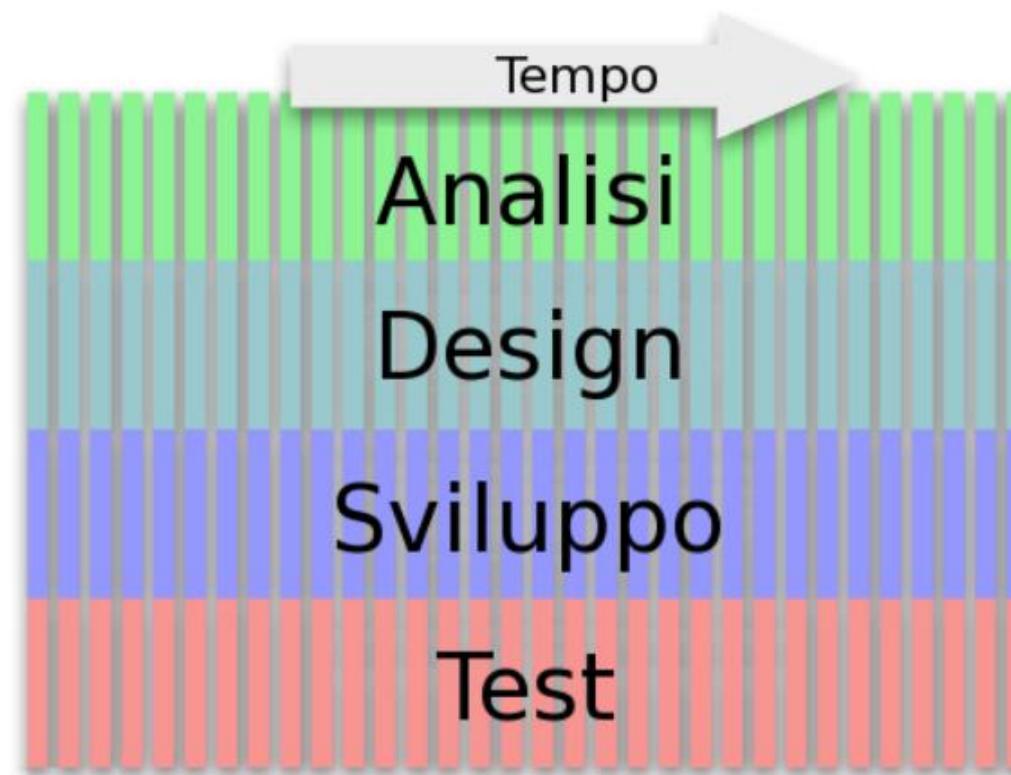
Il ciclo di vita del software

Un esempio di modello Agile



Il ciclo di vita del software

Fasi di sviluppo Agile



Scrum

- E' un framework agile, all'interno del quale possiamo utilizzare vari processi e varie tecniche
- Scrum non ha delle regole rigide: ogni azienda, dunque, ha la piena libertà di adattare la metodologia come meglio crede

Non attaccatevi troppo ad una qualsiasi arma o ad una singola scuola di combattimento

Miyamoto Musashi

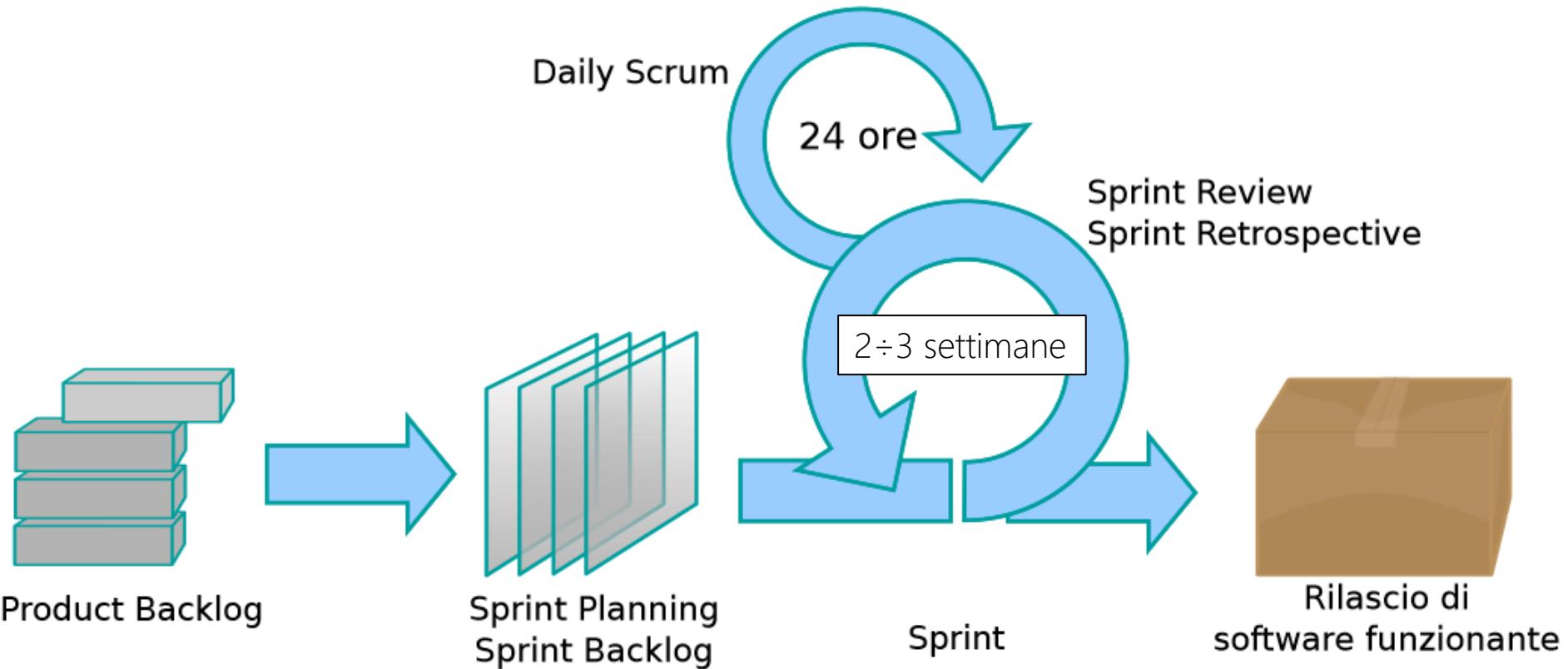


Scrum

- Il termine Scrum deriva dal rugby ed indica «il pacchetto di mischia»
- Tutti i protagonisti devono lavorare insieme per raggiungere un unico obiettivo
- Nel rugby, lo scopo è di spingere tutti nella stessa direzione per conquistare la palla; nell'informatica, fine ultimo è quello di sviluppare software più efficiente e di migliore qualità.
- Particolarmente indicato per situazioni progettuali caotiche, con obiettivi e requisiti in costante cambiamento, e quindi molto imprevedibili.

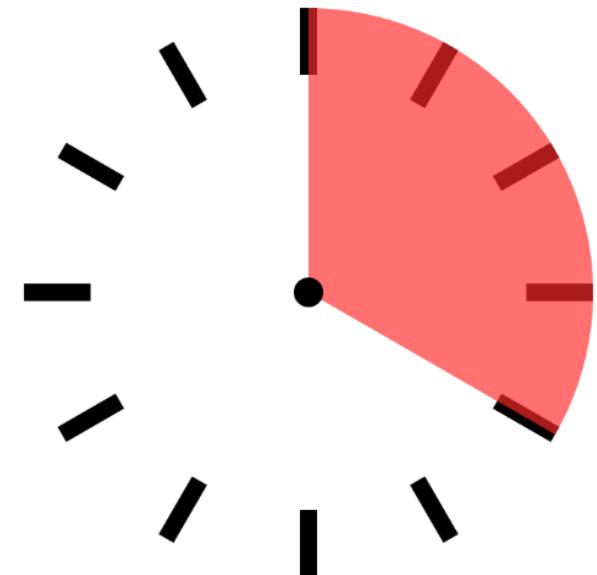


Scrum



Scrum: i principi

- Timeboxed
 - progetti affrontati in modo iterativo ed incrementale
 - software funzionante con sessioni di durata definita
- Quattro ragioni fondamentali per il Timeboxed
 - Permette di concentrarsi su ciò che è più determinante, evitando l'overengineering.
 - Serve come indicatore di tempo effettivo da dedicare ad un lavoro, evitando le perdite di tempo ed aumentando l'efficienza.
 - È uno strumento efficace contro la distrazione
 - Permette di anticipare del lavoro durante i momenti "liberi" tra un impegno e l'altro



Scrum: i principi

- Sprint
 - Rappresentano sessioni di durata definita
 - Tipicamente, hanno una durata che oscilla tra le 2 e le 4 settimane, ma questo dipende dalle scelte progettuali
 - Comprendono tutte le attività di sviluppo (analisi, progettazione, realizzazione, test, integrazione)
 - Solo il team interviene nello sviluppo, il cliente le considera come black-box
 - ogni giorno si tiene traccia del completamento dei task
 - le priorità, una volta decise e inserite nello sprint planning, nessuno ha la possibilità di cambiarle all'interno dello sprint stesso

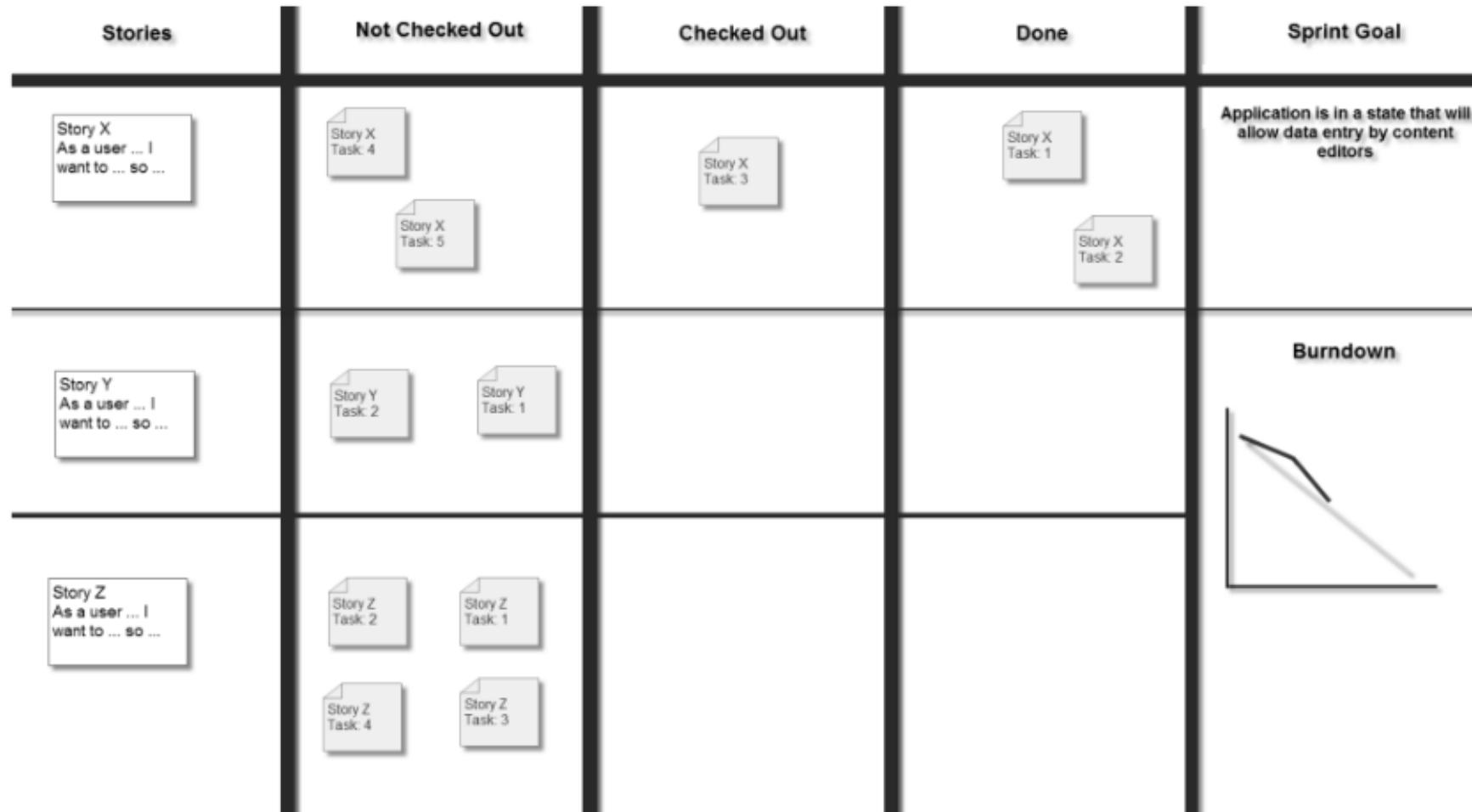


Scrum: i principi

- Team Capacity
 - Misura il tempo disponibile da dedicare allo sprint da parte dei membri del team
 - Può variare sulla base del tempo che i membri del team passano in vacanza, in malattia, che devono dedicare ad attività fuori dello scrum, etc.
 - Viene utilizzata per determinare il numero di attività del product backlog da prendere in carico in un determinate sprint
- Team Velocity
 - Si basa sul lavoro effettivamente completato negli sprint precedenti (magari misurato tramite gli story points)
 - Viene utilizzata (insieme alla Capacity) per determinare il numero di attività del product backlog da prendere in carico in un determinato sprint



Scrum: utilizzo di Task Board



Scrum: i ruoli

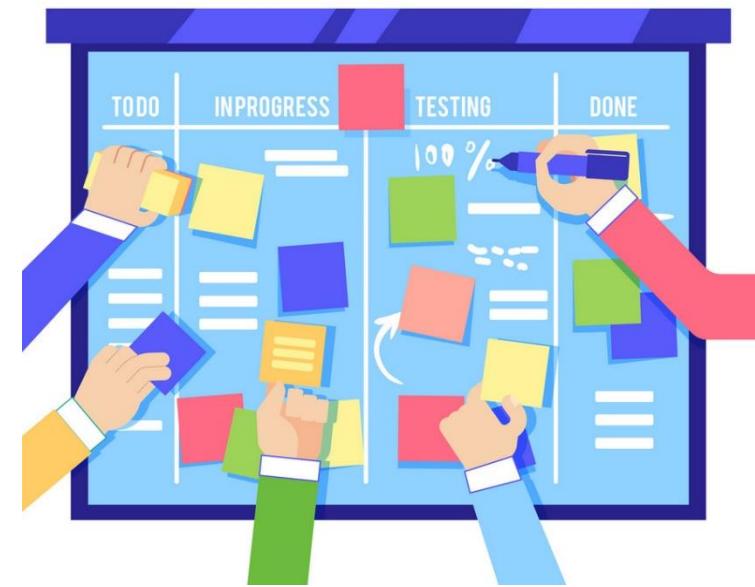
- **Product Owner:** è il proprietario del prodotto ed è colui che dirige il progetto dal punto di vista del business, oltre a presentare la visione delle funzionalità al team degli sviluppatori
 - Gestisce il Product Backlog
 - Quantificare l'impegno delle funzionalità
- **Scrum Master:** è il sovrintendente all'applicazione del processo di sviluppo del progetto
 - protegge il team dalle distrazioni esterne al progetto
 - incoraggia e incentiva la comunicazione all'interno del team per semplificare la risoluzione dei problemi

Scrum: i ruoli

- **Scrum Team:** insieme delle persone con competenze specialistiche diverse in grado di tradurre le richieste del Product Owner in un prodotto potenzialmente rilasciabile entro la fine dello Sprint.
 - Tipicamente un gruppo che varia tra le quattro e le otto persone
 - Valuta ciò che deve esser fatto e si impegna pubblicamente a realizzarne una parte, creando un elenco di tasks
- **Stakeholders**
 - Sono tutti coloro che possono inoltrare richieste relative all'evoluzione o alla modifica del prodotto

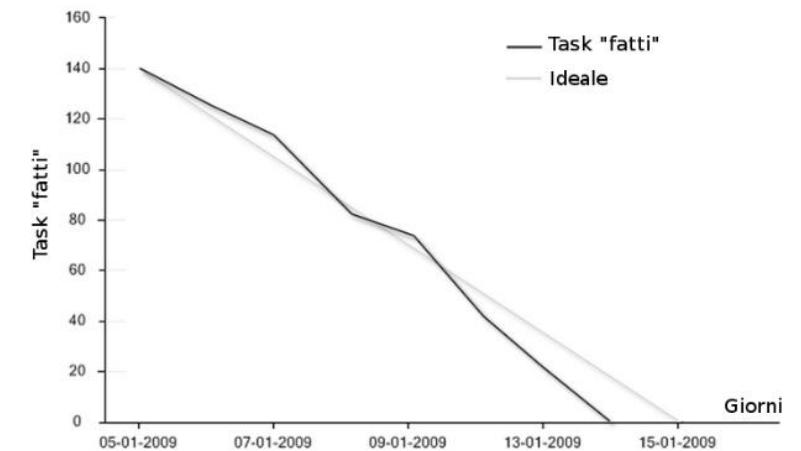
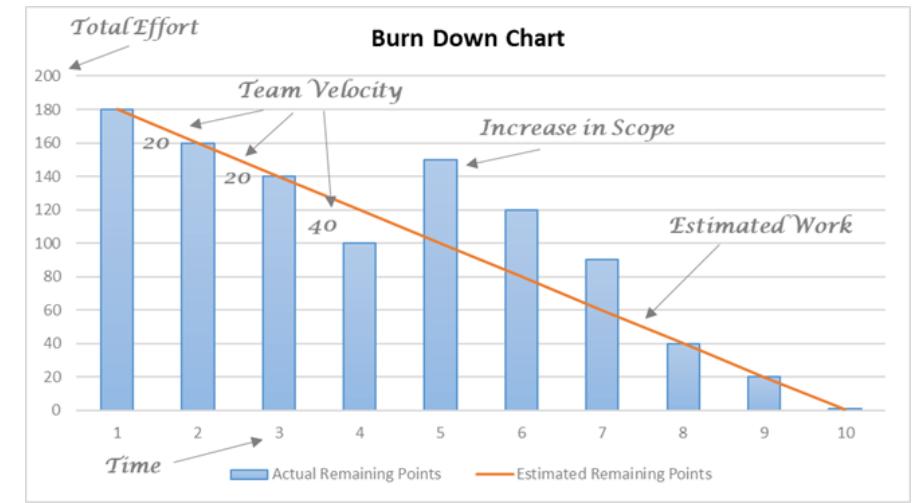
Scrum: Backlogs

- I Backlogs sono gli elenchi delle funzionalità che sono emerse e che servono a costruire il prodotto
 - *Product Backlog*: elenco di tutte le funzionalità del prodotto, ordinato per priorità.
 - Gestito dal Product Owner (o da qualche delegato)
 - Non è necessario avere già all'inizio un elenco completo di requisiti
 - Le funzionalità del Product Backlog vengono espresse sotto forma di user stories
 - *Sprint Backlog*: è la lista delle funzionalità (prese dal Product Backlog) che devono essere implementate in uno sprint. Viene popolato durante lo Sprint Planning e le attività (user stories) prese in carico vengono suddivise in task

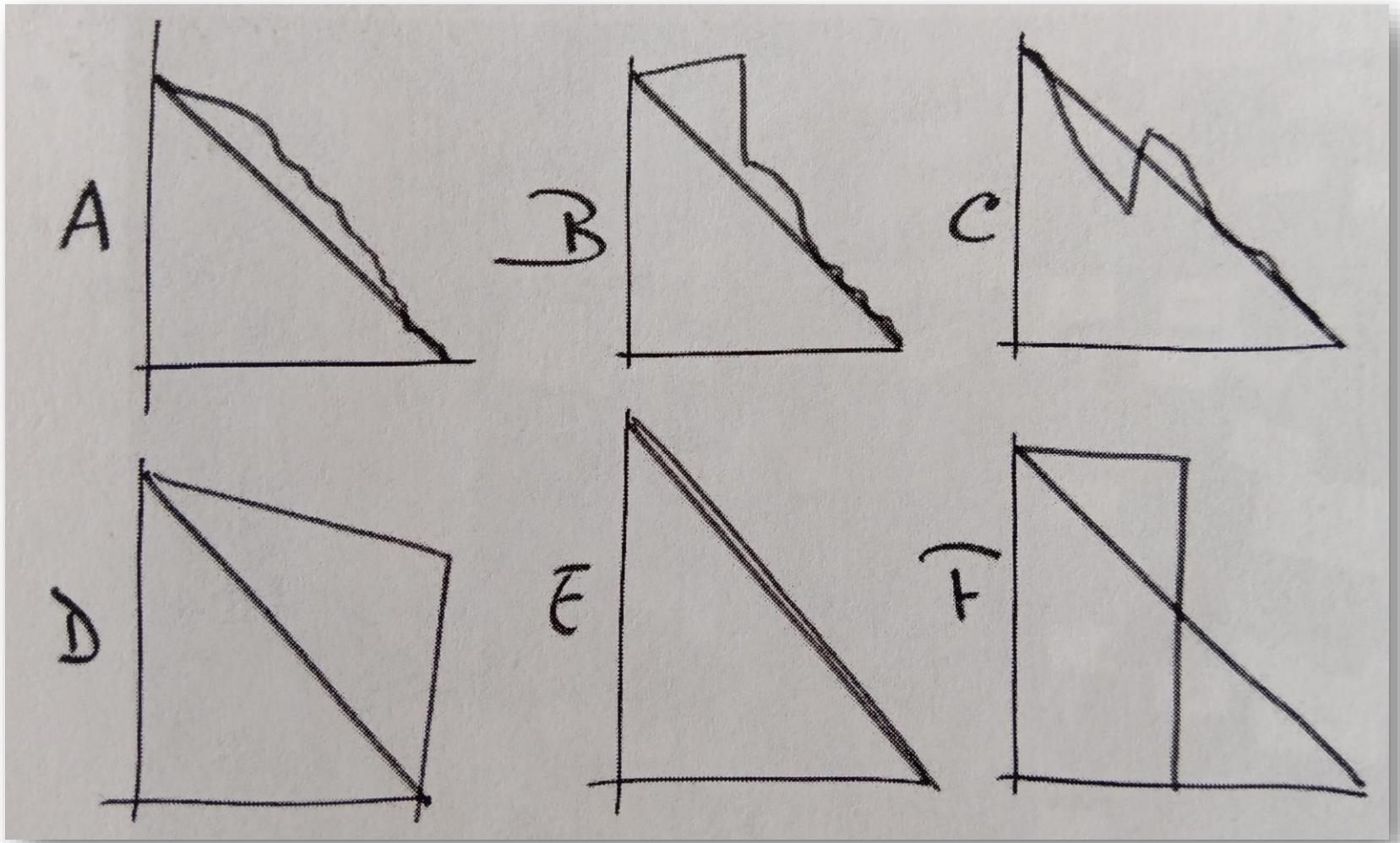


Scrum: Burndown Chart, rispettare le scadenze

- Misura la quantità di elementi rimanenti di uno Sprint Backlog nel corso di uno Sprint
- Mostra se gli elementi dello sprint backlog sono stati fatti
- Sull'asse delle ordinate vengono messi il numero degli elementi da sviluppare e sulle ascisse i giorni nello sprint corrente
- Il team aggiorna il grafico quotidianamente.
- Deve essere aggiornabile e modificabile in modo semplice e non complicato



Scrum: interpretazione di un Burndown Chart



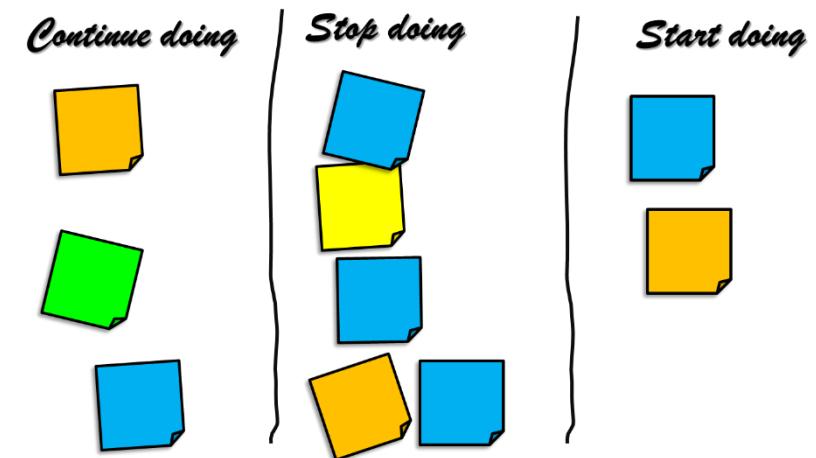
Scrum: attività

- Daily Scrum:
 - riunione quotidiana di 15 minuti circa tra lo Scrum Master e tutti i partecipanti, da svolgere rigorosamente in piedi per evitare che duri di più
 - Viene effettuata sempre nello stesso luogo e alla stessa ora, ogni giorno
 - Identifica solamente i "colli di bottiglia", cioè le difficoltà riscontrate, e non la loro risoluzione
- Sprint Planning:
 - è una riunione, fatta con il proprietario del prodotto e gli altri stakeholders, per la pianificazione dello sprint
 - Viene deciso ciò che sarà fatto nello Sprint,
 - Viene deciso il come il Team andrà ad implementare la funzionalità in modo da formare un incremento del prodotto durante lo Sprint

Scrum: Sprint Review e Sprint Retrospective

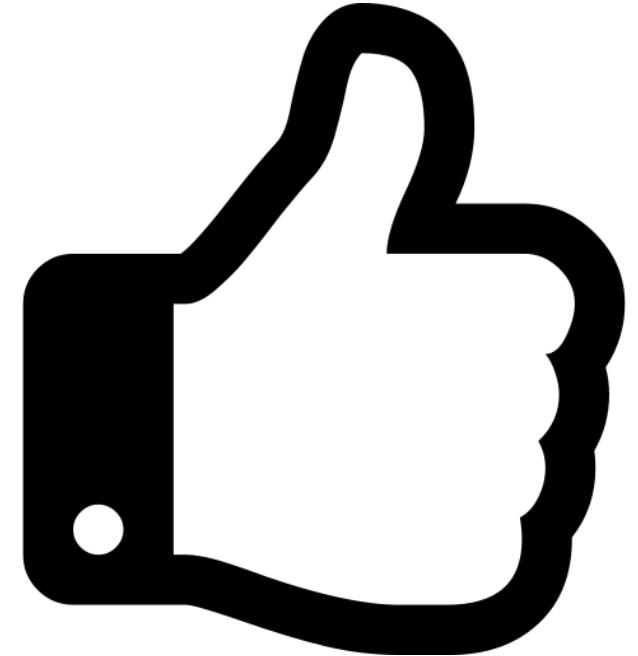
Sono delle riunioni, fatte con il proprietario del prodotto e gli altri stakeholders

- Review: esporre gli elementi che sono stati sviluppati e verificare il lavoro (funzionalità rilasciate o modificate) dello sprint appena concluso (demo)
- Retrospective: fare un esame delle criticità riscontrate nello sprint appena concluso, in modo da aggiornare il processo lavorativo per aumentare l'efficienza e la produttività



Scrum: vantaggi

- Consente di lavorare in modo produttivo anche in situazioni particolarmente caotiche e confuse
- È fortemente orientato verso risultati concreti e verso la gestione dei cambiamenti, piuttosto che alla pianificazione precisa
- Porta ad un forte coinvolgimento del committente e ad una assunzione di responsabilità collettiva da parte del team
- Favorisce la creazione di team coesi



Scrum: svantaggi

- È un processo che definisce esclusivamente pratiche di project management
- Non fornisce indicazioni su come condurre altre discipline fondamentali (gestione requisiti, analisi e disegno, test, gestione configurazione, ecc.) ed è quindi opportuno integrarlo con altri approcci (ad esempio eXtreme Programming) per colmare le parti mancanti.

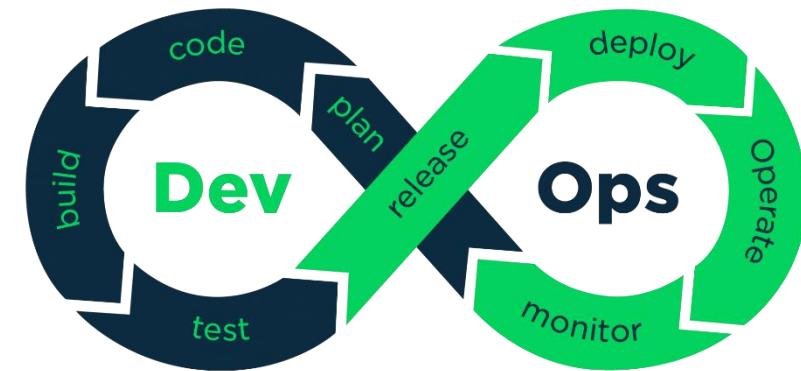


Che cos'è DevOps?

DevOps è un termine che indica una serie di pratiche atte ad enfatizzare la collaborazione e comunicazione dei team che si occupano di sviluppo (Dev) e di operations (Ops).

Il processo del software delivery e del cambiamento infrastrutturale deve essere automatizzato.

Mira a stabilire una cultura in un ambiente in cui le fasi di build, test e release avvengono ad un passo molto più veloce del normale e con più affidabilità.



Source: <https://en.wikipedia.org/wiki/DevOps>

Pratiche di DevOps

- Infrastructure as Code (IaC)
- Continuous Integration
- Automated Testing
- Continuous Deployment
- Release Management
- App Performance Monitoring
- Load Testing & Auto-Scale
- Availability Monitoring
- Change/Configuration Management
- Automated Environment De-Provisioning
- Self Service Environments
- Automated Recovery
- Feature Toggles
- Hypothesis Driven Development



DevOps: i benefici

- Si applica ad ogni app, qualsiasi codice e qualsiasi piattaforma
- Funziona in ambito cloud, on-premise o ibrido
- Velocizza i rilasci
- Produce un prodotto migliore nel tempo
- Nessun problema di deployment
 - Se ci sono problemi, sicuramente è colpa del team
- Tempi di recovery quasi azzerati da possibili errori

Getting Started

- Usa un sistema di Source Control
- Non aspettatevi un cambio immediato
- Preparate i test
- Tenete delle alternative sempre pronte
- Tool vecchi e nuovi possono lavorare assieme



Tutti devono adottare DevOps



Se lo adotta solo una parte del team, non ci sono benefici

DevOps tools

Si possono usare tool diversi per raggiungere lo stesso scopo

Jira, TFS, Azure DevOps ...

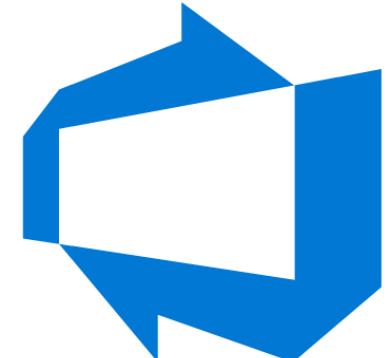
Azure DevOps Server (was Team Foundation Server)

Pro: Funziona on-premise

Contro: deve essere manutenuto

Azure DevOps Services

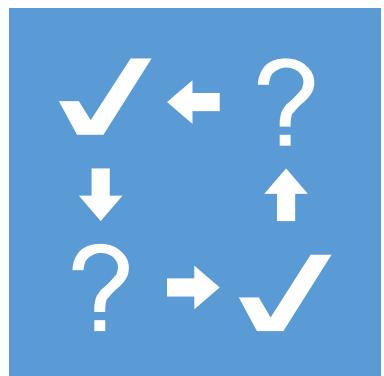
Costantemente aggiornato (grazie a DevOps) ogni 3 settimane



Azure DevOps



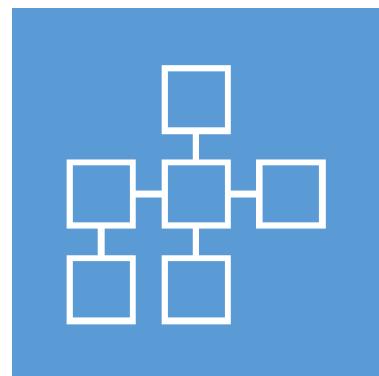
Azure DevOps per l'enterprise



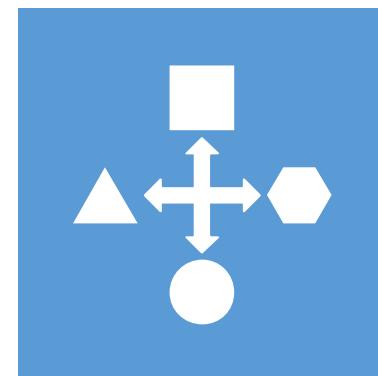
Compliance



Data
Sovereignty



Active Directory



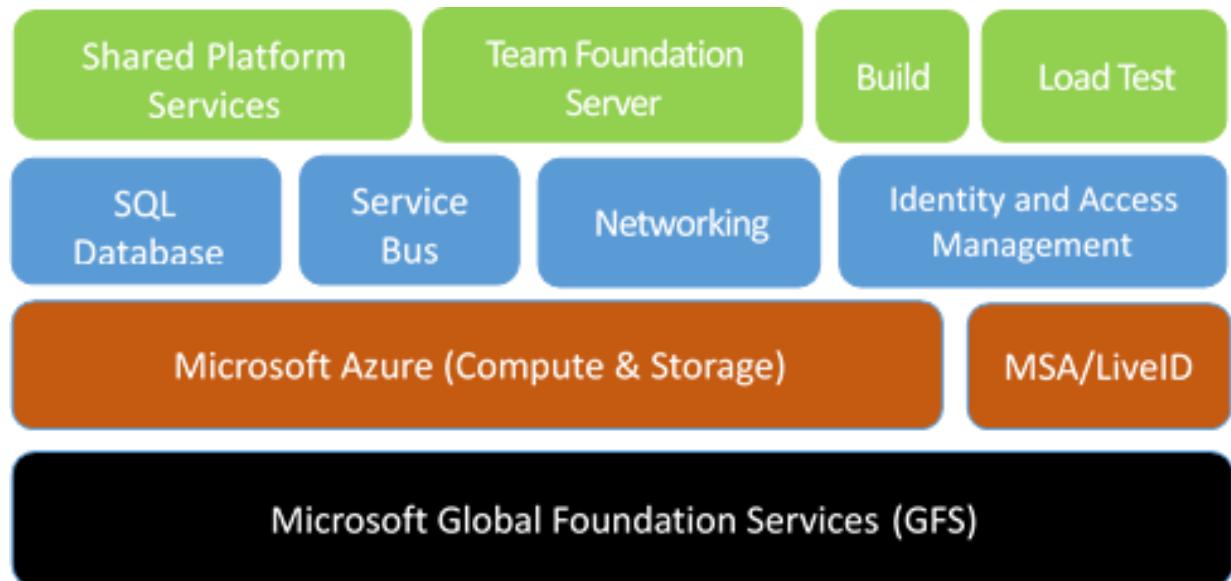
Process
Customization



Reporting

Disponibilità e sicurezza dei dati

- Azure DevOps è costruito in Microsoft Azure
- Geo-replica garantita dei dati in DC opposti all'interno della stessa regione
- Daily backup dell'intero sistema
- Protezione ad attacchi DDoS
 - Grazie allo stesso firewall di Microsoft Azure
- 24x7 operations team
 - Live-site con incidenti aggiornato in real-time



Il ruolo degli agents

Per fare le build ed i deploy è necessario avere una macchina che faccia del lavoro

- Può essere una macchina virtuale, una macchina on-premise o su Azure
- Le build e le release possono essere fatte in parallelo
 - Solo una pipeline è gratuita, le altre >15\$/mese
- Agent hosted o privati
- Sistema operativo a scelta



Azure Pipelines

Private agents

Ci posso installare tutto il software che voglio

- La gestione di VM, SO e software è a carico nostro

Serve un Personal Access Token

- Generato dal portale di Azure DevOps

Download dell'agent

- Scaricabile direttamente dalla propria organization

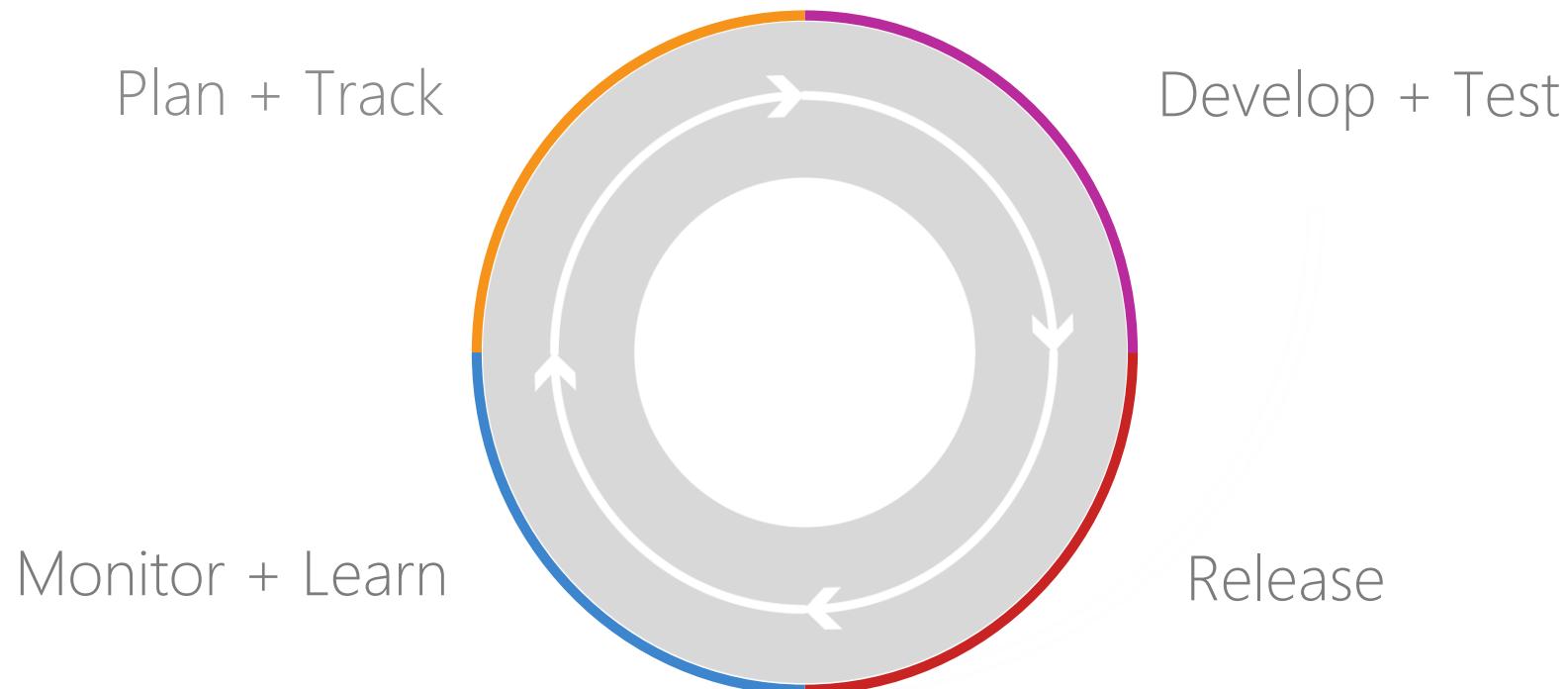
Configurazione dell'agent

- PowerShell -> .\config

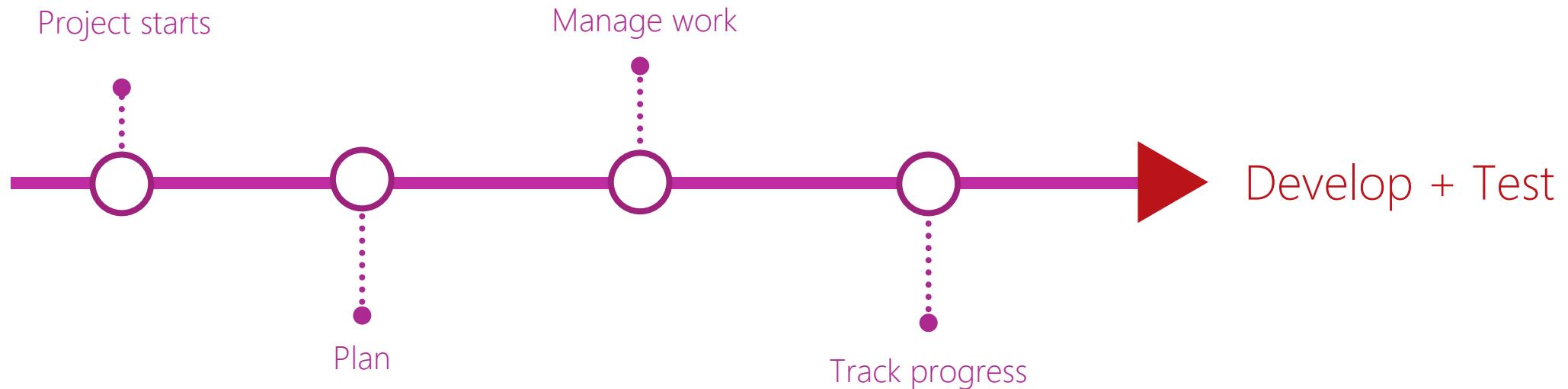


Azure Pipelines

Project lifecycle



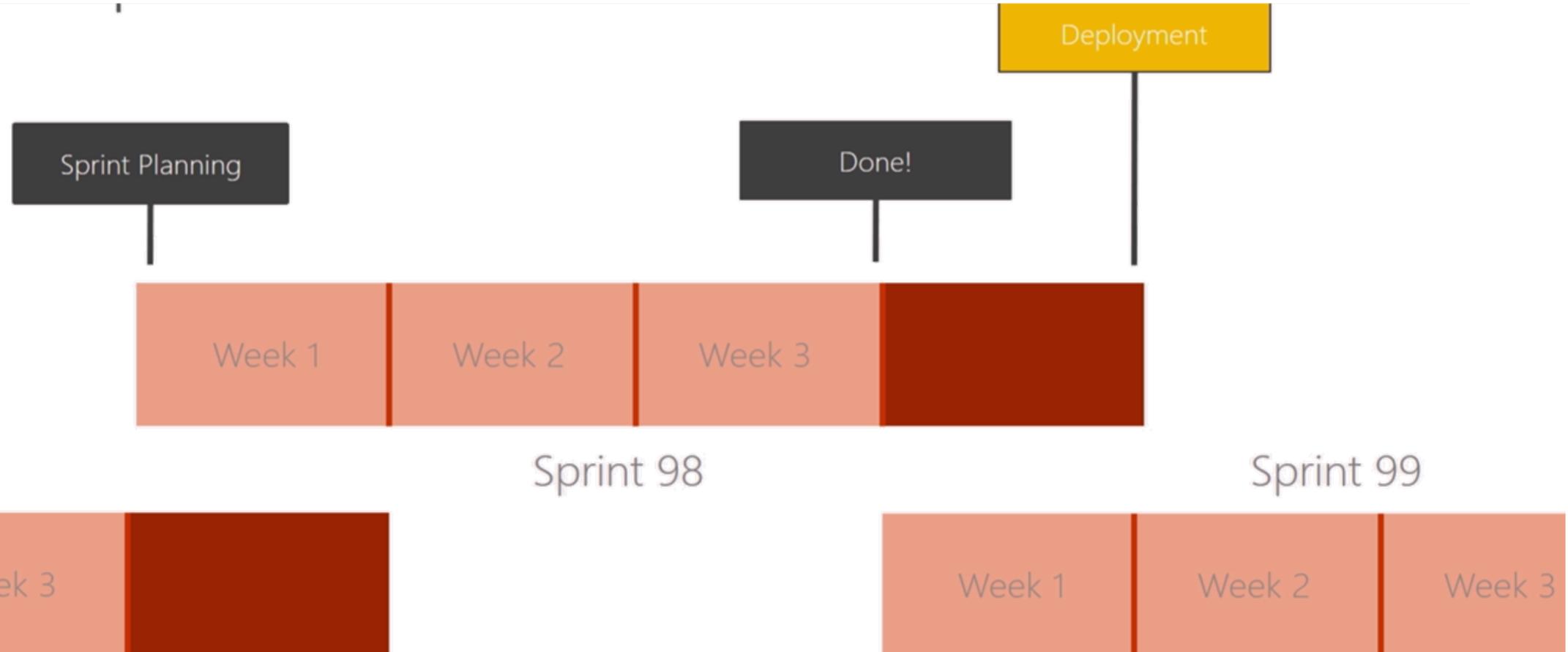
Planning phase



- Dashboards e grafici
- Work Items
- Processi personalizzabili
- Backlog & Kanban Board personalizzabili



Agile planning



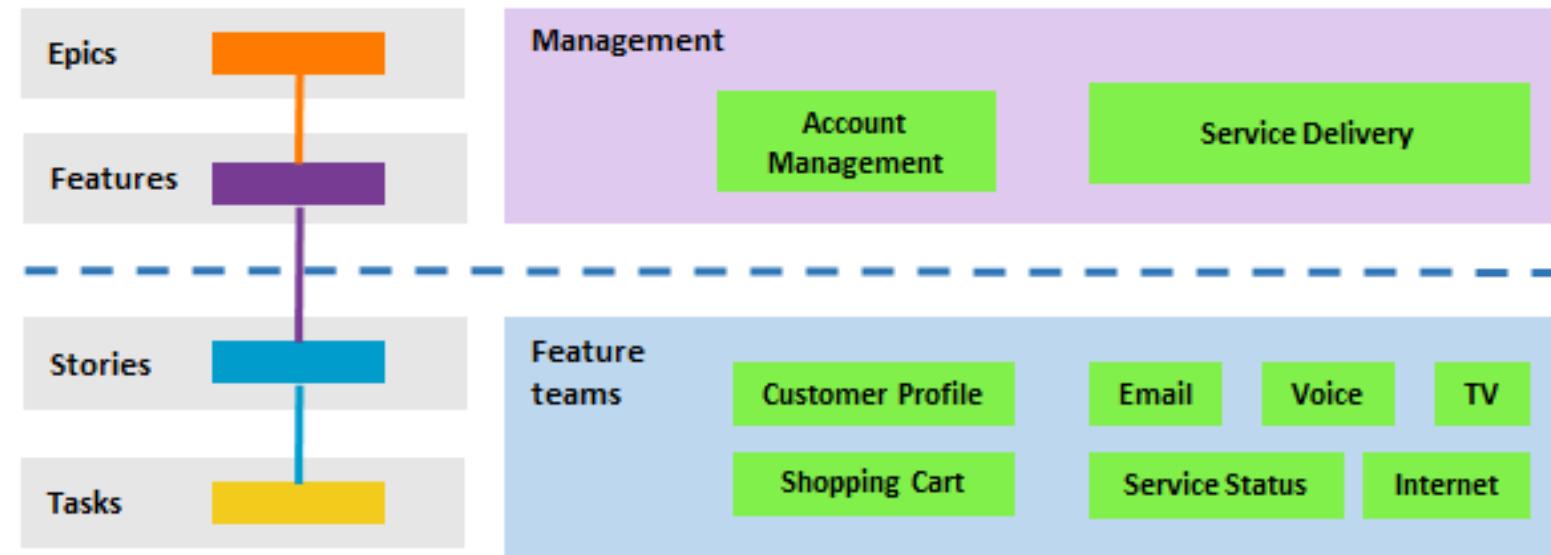
Definizione di lavoro

User Story e Task

Dedicated to developers who need granular access to information

Epic e feature

Dedicated to managers (CTO/CIO/PM) who need to see high-level progress



Work items management

Backlogs Queries Plans*

Epics Features Backlog items

Backlog Board Forecast Off Parents Hide In progress items Show Mapping Off

New | Create query | Column options |

Filter

Current Sprint 1 Product Backlog 2

Ord...	Work Item Type	Title	State	Effort	Value Area	Iteration Path	Area Path	Tags
1	Product Back...	Create appointment			Business	MyHealthClinic	MyHealthClinic\Web	
2	Product Back...	Sign-in with O365		3	Business	MyHealthClinic\Sprint 1	MyHealthClinic	
3	Product Back...	Create New Appointment		0	Business	MyHealthClinic	MyHealthClinic	
+	4	Product Back...	Backend Services		0	Business	MyHealthClinic	MyHealthClinic
5	Product Back...	Appointment detail info		0	Business	MyHealthClinic	MyHealthClinic	Appointment C...
6	Product Back...	Personal Patient Information		0	Business	MyHealthClinic	MyHealthClinic	Compliance Pa...
7	Product Back...	Appointment Integration with Native Calendar		0	Business	MyHealthClinic	MyHealthClinic	Appointment C...
8	Product Back...	Authentication		0	Business	MyHealthClinic	MyHealthClinic	
9	Product Back...	Summary information		1	Business	MyHealthClinic\Sprint 4	MyHealthClinic	
10	Product Back...	Appointment detail info		3	Business	MyHealthClinic\Sprint 3	MyHealthClinic	Appointment F...
11	Product Back...	Update visit status		0	Business	MyHealthClinic	MyHealthClinic	Doctor Notifica...
12	Product Back...	Manage Doctors		0	Business	MyHealthClinic	MyHealthClinic	Clinic Doctor
13	Product Back...	Patients evolution		4	Business	MyHealthClinic\Sprint 3	MyHealthClinic	
14	Product Back...	My medical treatment		2	Business	MyHealthClinic\Sprint 3	MyHealthClinic	
15	Product Back...	Total income & expenses summary		8	Business	MyHealthClinic\Sprint 2	MyHealthClinic	Compliance Pe...
16	Product Back...	Summary patient information		1	Business	MyHealthClinic\Sprint 3	MyHealthClinic	Compliance Pa...

Work items management

Visual Studio Online / Fabrikam Fiber / Team A Aaron Bjork |

HOME CODE WORK BUILD TEST SEARCH * Search work items

Backlogs Queries Backlog **Board** Settings

Team A Stories

New	Spec ①	In Progress	Sign Off	Deployed
<p>+ New item</p> <p>1047 User can update their profile and link to their gravatar profile Christina Kelly 1</p> <p>1040 Profile editing directly from a mobile device - supports avatar changing Christina Kelly 5</p> <p>1075 Email links directly from an item in your cart - always visible and</p>	<p>1038 User can update their profile to include an avatar Noah Munger 5</p> <p>966 Password reset for anyone that forgot their password Aaron Bjork 5</p> <p>1034 Allows users to choose a font size suitable to their needs.</p>	<p>964 Responsive css for better rendering across different device sizes. Christina Kelly 3</p> <p>969 Responsive design causing some menus to flicker on resize Aaron Bjork</p> <p>1039 Users can update their profile with language and timezone Lowell Steel 3</p>	<p>1086 Add new items Christina Kelly 3</p> <p>1037 Update profile with notification settings Christina Kelly 8</p> <p>1078 Profile errors happening for users with stale items in their cart. Lowell Steel</p>	<p>962 RSS subscription - one click from the home page Aaron Bjork 8</p> <p>1046 A user can choose their landing page after login Lowell Steel</p> <p>961 RSS feed of all project activity - displayed on the project home page. Aaron Bjork 13</p>

Personalizzazione della kanban

Ogni campo è personalizzabile

The screenshot shows the 'Fields' configuration page in Jira. On the left, a sidebar menu lists categories: Cards, Fields (selected), Styles, Tag colors, Annotations, Tests, Board, Columns, Swimlanes, Card reordering, Charts, Cumulative flow, General, Backlogs, Working days, and Working with bugs. The main panel is titled 'Fields' with the sub-instruction: 'Show the important information to your team. Fields are editable directly on the card.' It displays two tabs: 'Product Backlog Item' (selected) and 'Bug'. Under 'Core fields', several checkboxes are checked: 'Show ID', 'Show Assigned To as:' (set to 'Avatar and full name (default)'), 'Show Effort', and 'Show Tags'. Under 'Additional fields', there is a '+ Field' button followed by two dropdown menus: 'Priority' and 'Area Path', each with a red 'X' icon indicating they are currently disabled or removed. At the bottom, a section titled 'Show empty fields' contains a checkbox: 'Check if you want to display fields, even when they are empty.'

Personalizzazione della kanban

Ogni campo è personalizzabile

Lo stile è fondamentale per visualizzare le criticità del progetto

Styles

Styling rules make the cards with important information stand out. When a work item matches more than one rule, the first rule is used.

+ Styling rule

Rule Name	Preview	Enabled
High Effort	Title	<input checked="" type="checkbox"/>

Rule name
Name

Styling
Select your style choices.

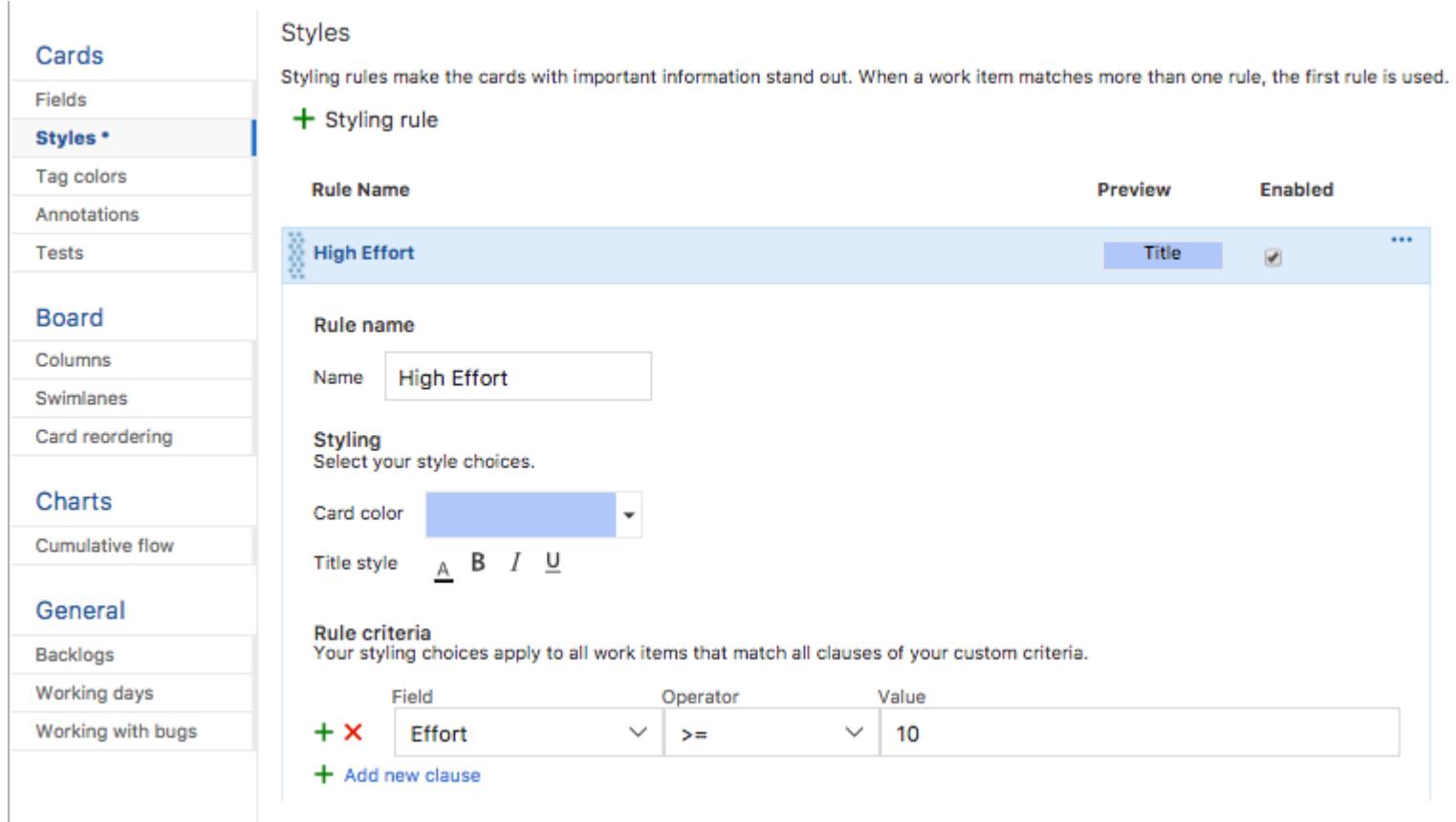
Card color

Title style A B I U

Rule criteria
Your styling choices apply to all work items that match all clauses of your custom criteria.

Field	Operator	Value
Effort	>=	10

+ Add new clause



Personalizzazione della kanban

Ogni campo è personalizzabile

Lo stile è fondamentale per visualizzare le criticità del progetto

Si possono modificare le singole colonne dell'intero processo

The screenshot shows the 'Columns' configuration page for a Kanban board. On the left, a sidebar lists various customization options under 'Cards' (Fields, Styles, Tag colors, Annotations, Tests) and 'Board' (Columns*, Swimlanes, Card reordering). The 'Columns*' option is selected. The main area displays the current column structure: Backlog, Analyze, Develop, Test, and Done. The 'Analyze' column is currently selected. A 'Column name' field shows 'Name' and 'Analyze'. Below it, a 'Work in progress limit' field is set to '5'. There is also a checked checkbox for 'Split column into doing and done'. Under 'State mapping', 'Bug' is mapped to 'Approved' and 'Product Backlog Item' is also mapped to 'Approved'. At the bottom, there is a section for defining 'Definition of done'.

Disponibilità di lavoro

Fondamentale per tenere traccia di tempistiche

Siamo sicuri di fare il delivery dello sprint in tempo?

I membri del team sono troppo allocati?

Quanto tempo può lavorare un membro del team?

Per giorno, settimana, sprint...

Ci sono delle vacanze, ferie?

Vanno gestite poiché non ci sarà sviluppo



Disponibilità di lavoro

Screenshot of a software interface showing work capacity and availability.

The main navigation bar includes Backlogs, Queries, Plans*, Features, Backlog items, Current, and Sprint 1. The Capacity tab is selected.

Project details: Web Sprint 1, April 10 - April 28, 10 work days remaining.

Work details section shows a summary of work hours:

Team	Work
(34 of 136 h)	34 of 136 h

Work By: Activity:

Activity	Work
Development	26 of 86 h
Testing	8 of 50 h

Work By: Assigned To:

User	Work
BS Brian Spann	12 of 36 h
JO Jeff Ogorek	8 of 50 h
NP Nick Patterson	14 of 50 h

Capacity configuration dialog:

Add new capacity user

User	Days Off	Activity	Capacity Per Day
Jeff Ogorek	0 days	Testing	5
Nick Patterson	0 days	Development	5
Brian Spann	0 days	Development	4

Team Days Off: These days off apply to the whole team.

Days off for: Brian Spann dialog:

Start Date: 4/24/2017, End Date: 4/24/2017, Net Days Off: 1, Total: 1

OK Cancel

La dashboard

My Health Clinic

Vision

- To Develop world class health service for our customer;

Mission

- Provide quality health services and facilities for the community; To be the preferred choice of service providers for our customers; Create a supportive team environment for patients, employees, and all staff.

Sprint 4

October 3 - October 18

My Health Clinic2 T...

Sprint 4
October 3 - October 18

1 day remaining

4 work items not started

All Items by Work Item Type

Work Item Type	Count
Task	38
Product Bac...	35
Test Case	9
Feature	4
Test Suite	3
Test Plan	2
Shared Para...	2
Bug	1

Welcome

Get started using Visual Studio Team Services to make the most of your team dashboard.

Manage Work
Add work to your board

Collaborate on code
Add code to your repository

Continuously integrate
Automate your builds

Visualize progress
Learn how to add charts

New Work Item

Enter title

Bug

Create

Work

Backlog
Board
Task board
Queries

Open Items (94)

ID	Work Item...	Title	Assigned To	State
20635	Feature	Security	Sachin Hri...	New
20636	Product Ba...	Authentication	Prafulla N...	Done
20637	Task	Authentication	Sachin Hri...	Done
20638	Product Ba...	Sign-in with O365	Sachin Hri...	New
20639	Task	Sign-in with O365	Sachin Hri...	Done
20640	Product Ba...	SSO Active directory authenticat...	Craig Cam...	Done
20641	Feature	Settings	Sachin Hri...	New
20642	Feature	Appointmets	Sachin Hri...	New

[View query](#)

Open Items by Work Item Type

Work Item Type	Count
Task	34
Product Bac...	16
Test Case	9
Feature	4
Test Suite	3
Test Plan	2
Shared Para...	2
Bug	1

Open Items by State

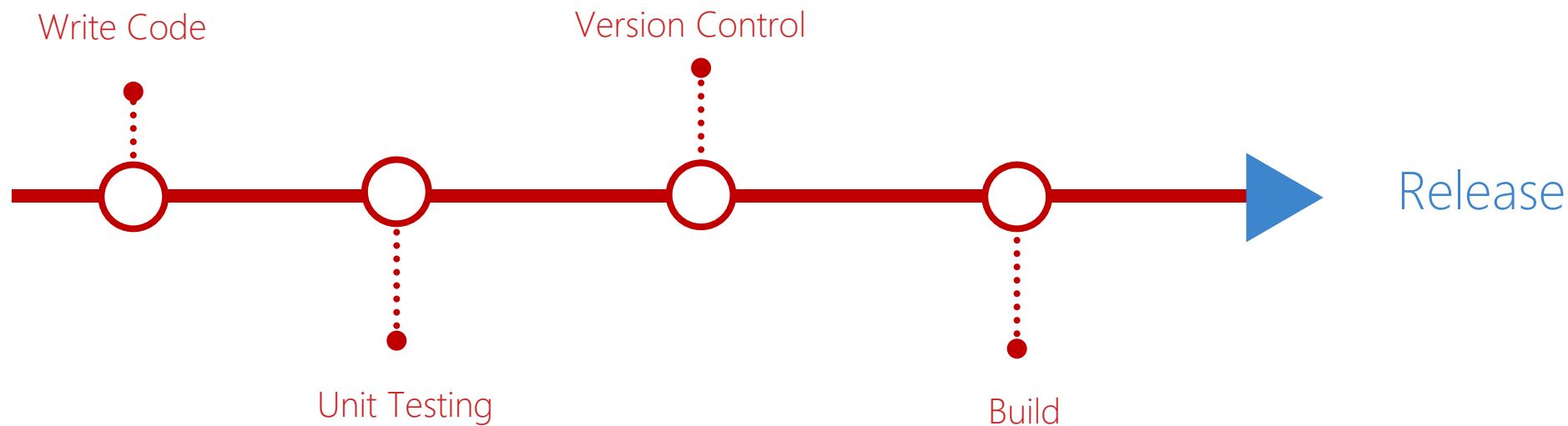
Date	Done	New	Committed	Design	In Progress	To Do	Active	Approved
Oct 11	~50	~10	~5	~5	~5	~5	~5	~5
Oct 12	~55	~10	~5	~5	~5	~5	~5	~5
Oct 13	~58	~10	~5	~5	~5	~5	~5	~5
Oct 14	~60	~10	~5	~5	~5	~5	~5	~5
Oct 15	~62	~10	~5	~5	~5	~5	~5	~5
Oct 16	~64	~10	~5	~5	~5	~5	~5	~5
Oct 17	~66	~10	~5	~5	~5	~5	~5	~5

Other Links

Request feedback
Configure schedule and iterations
Configure work areas

Team Members

Developing the product...



- Supporto per TFVC e GIT
- Integrazione con Visual Studio
- Continuous Integration per le builds



Gestione del codice

Obiettivo: Lavorare simultaneamente sullo stesso gruppo di file mantenendo il controllo dell'evoluzione delle modifiche che vengono apportate

Più sviluppatori lavorano in parallelo sullo stesso software

Definire procedure per tenere traccia e controllare i cambiamenti

- dei file
- del codice sorgente
- della documentazione



Gestione del codice

Mettere a disposizione un repository, di cui ogni sviluppatore può ottenere una copia di lavoro

Problema: se due sviluppatori tentano di modificare lo stesso file contemporaneamente, in assenza di un metodo di gestione degli accessi, essi possono sovrascrivere o perdere le modifiche effettuate



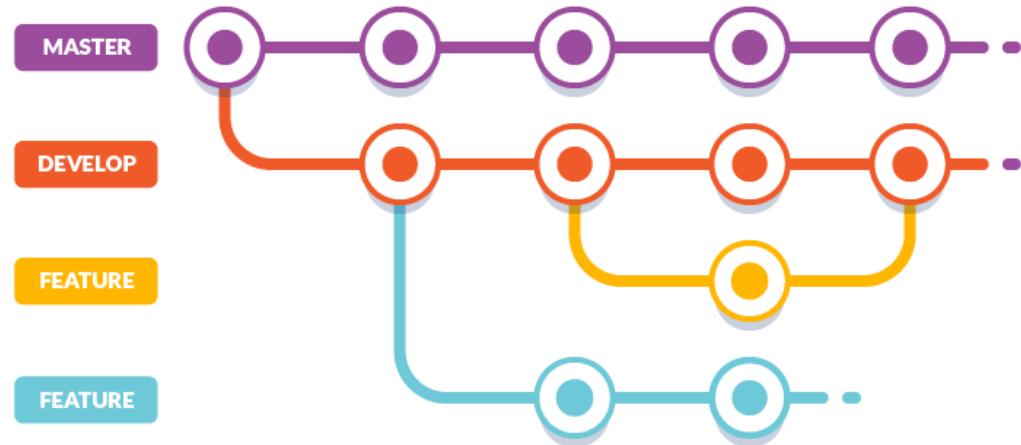
GIT Flow

GitFlow è un modello di branching per Git

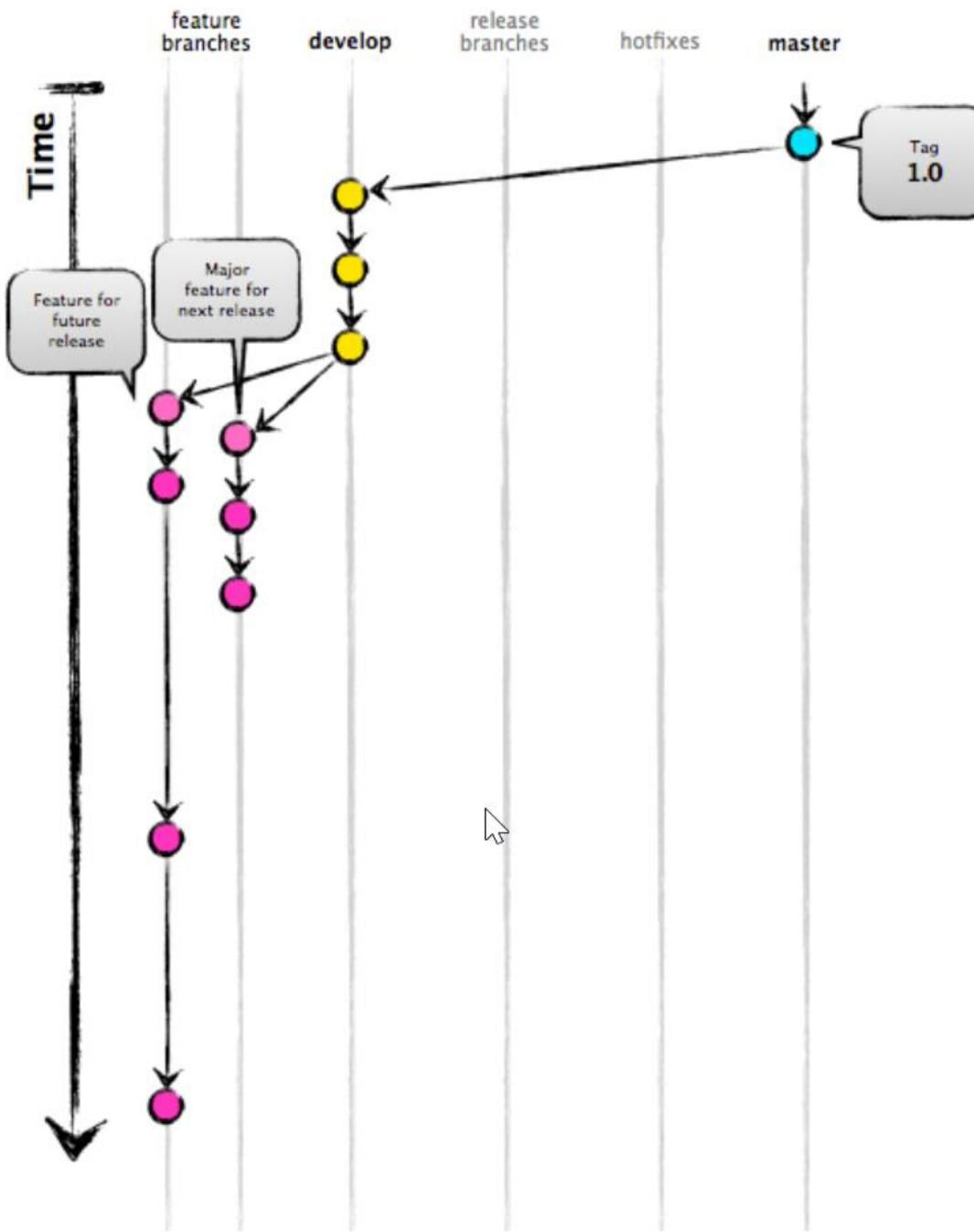
- È molto adatto alla collaborazione e al ridimensionamento del team di sviluppo

Vantaggi chiave

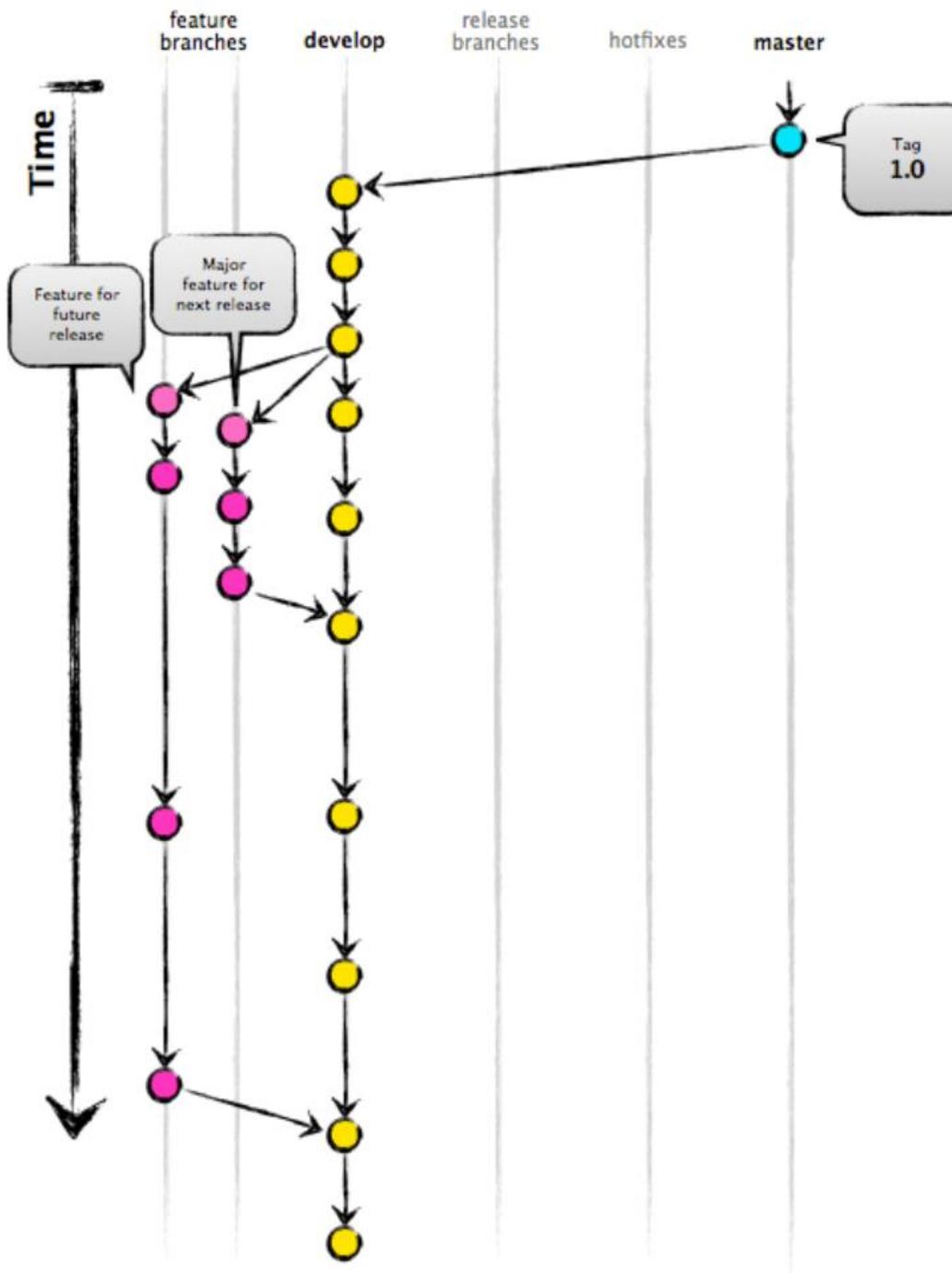
- Sviluppo parallelo
- Collaborazione
- Rilasciare l'area di gestione temporanea
- Supporto per le correzioni di emergenza



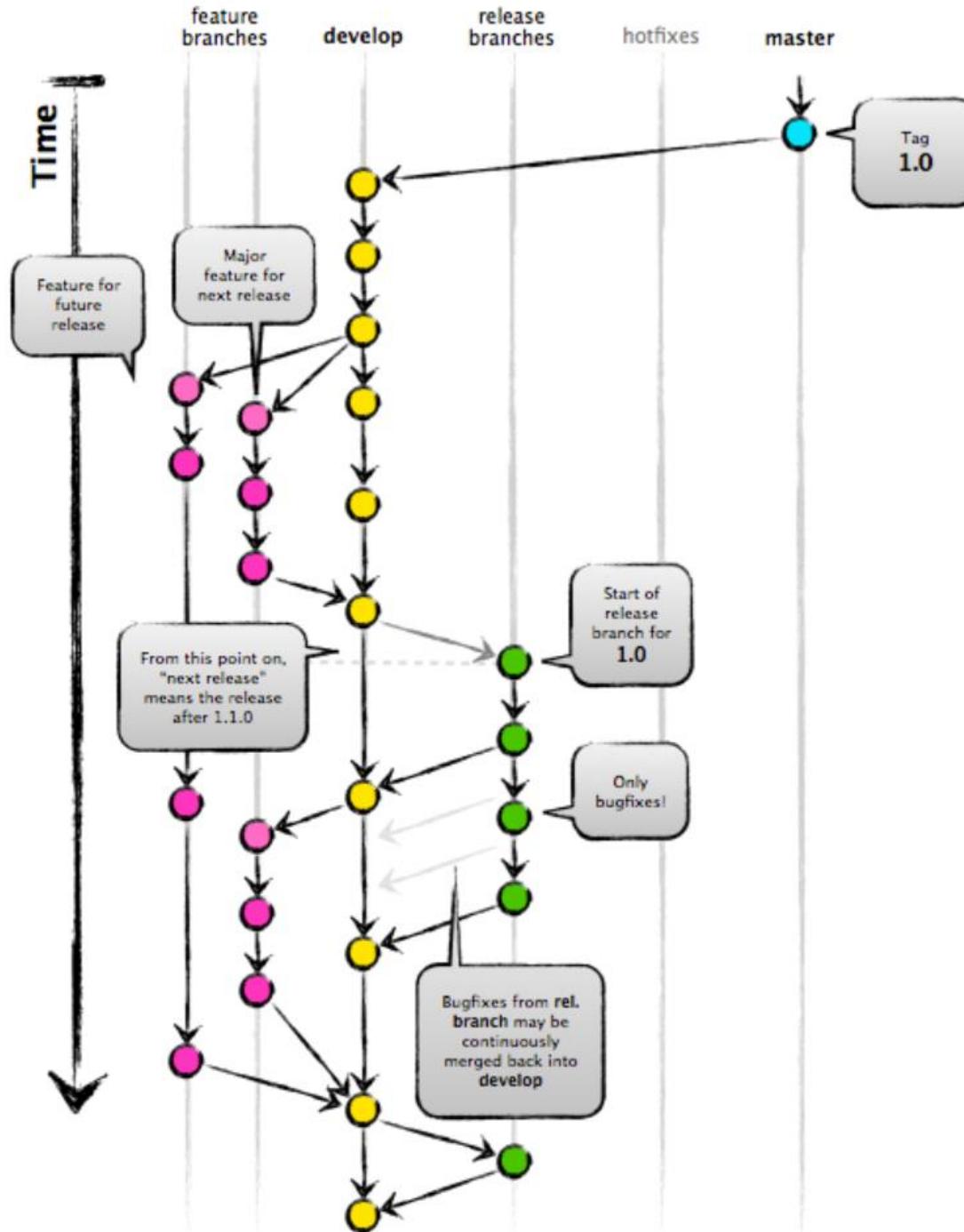
Git Flow



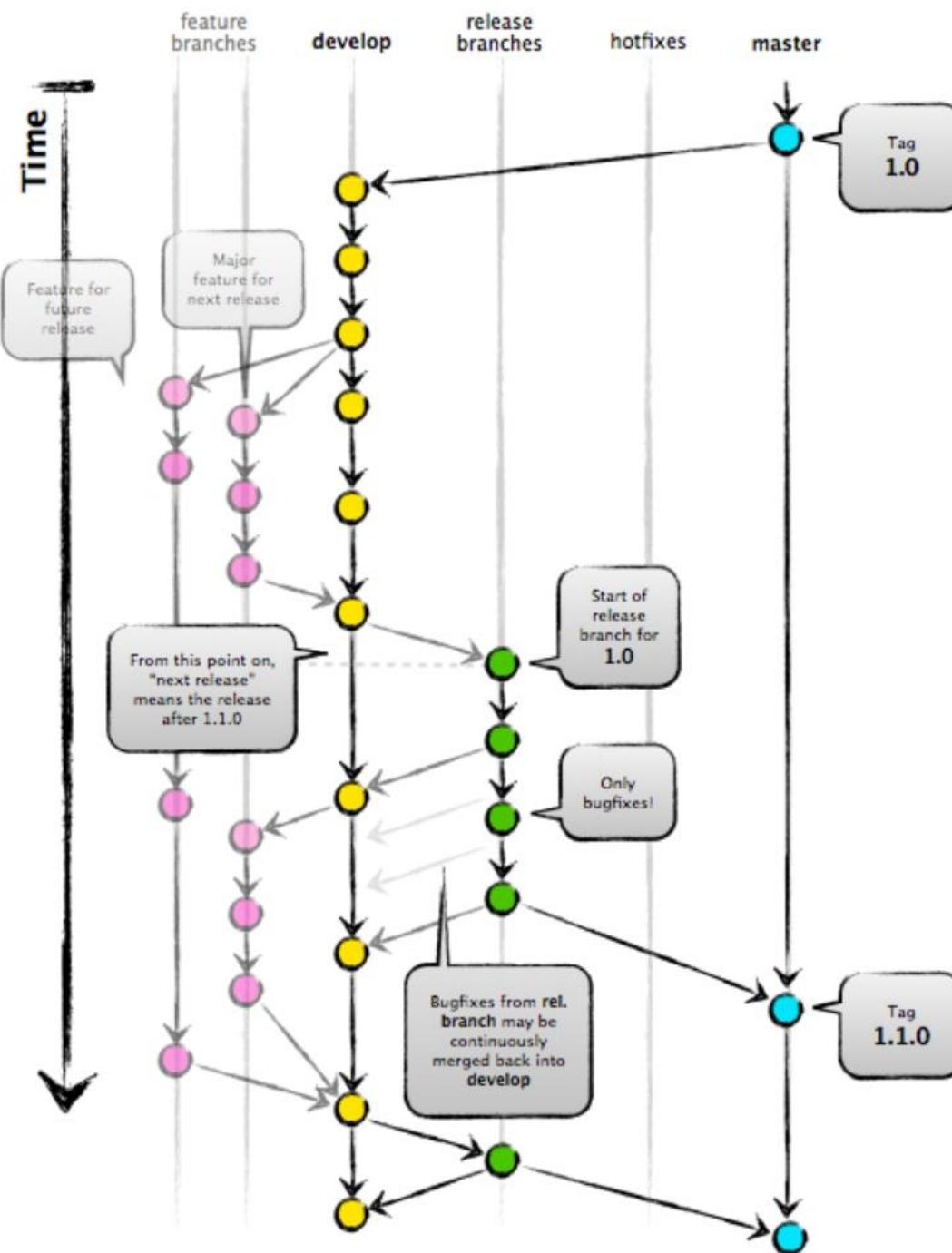
Git Flow



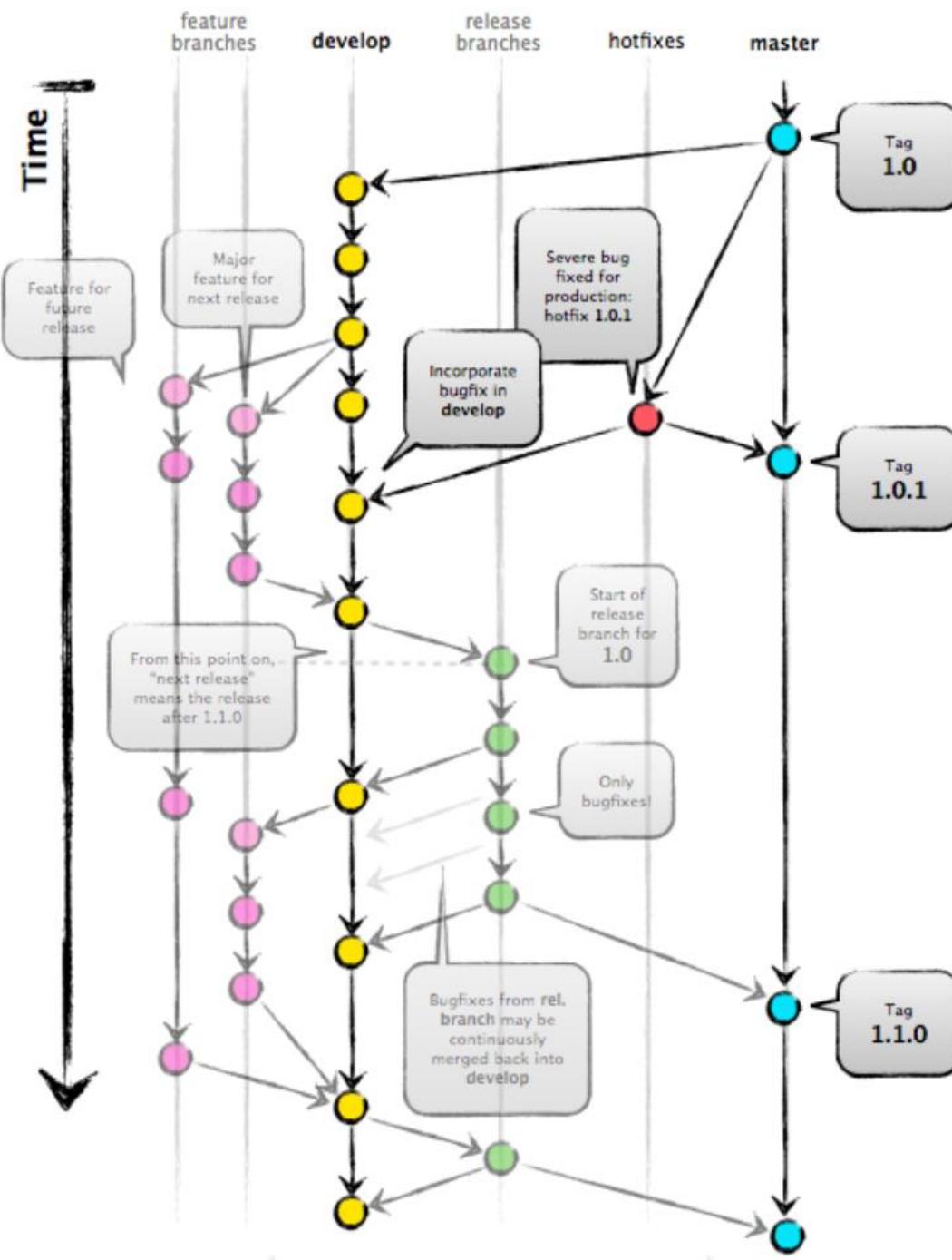
Git Flow



Git Flow



Git Flow

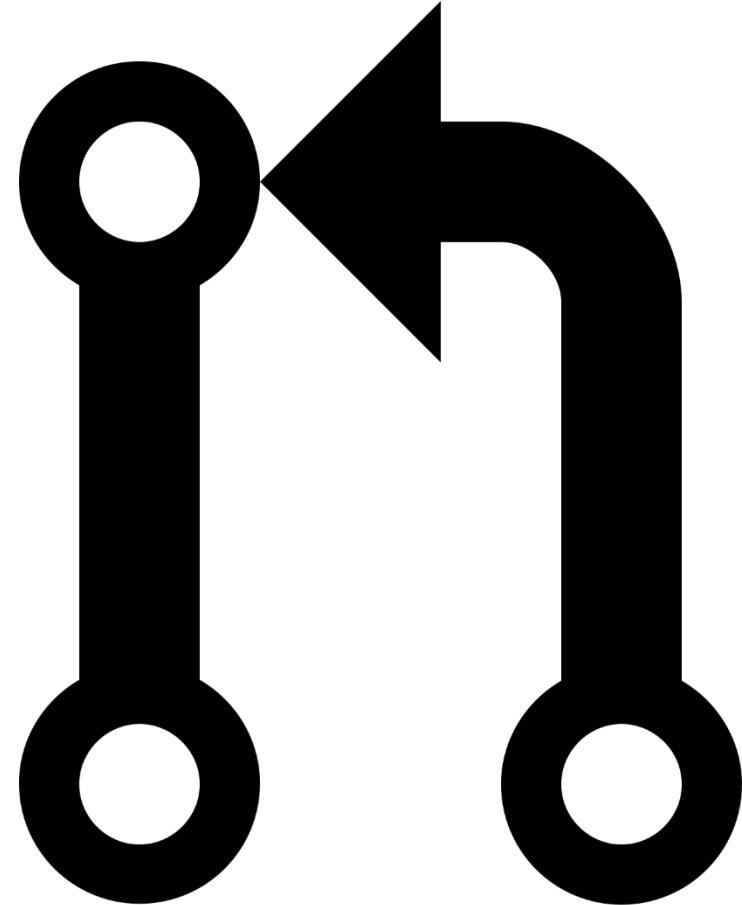


Pull Request

Le Pull Request consentono di comunicare agli altri le modifiche inviate a un ramo in un repository su GitHub

Una volta aperta una richiesta pull, è possibile

- discutere e rivedere le potenziali modifiche con i collaboratori
- aggiungere commit di follow-up prima che le modifiche vengano unite nel ramo di base
- Gestire tramite le policy di branch un processo di approvazione («proteggere il branch»)



Pull Request

2 COMPLETED Get method for Ticket with Notes

RA Roberto Ajolfi → develop into master

All resolved RA Delete source branch ...

Overview Files Updates Commits

Roberto Ajolfi completed the pull request on 1/10/2020 2:24 PM. Cherry-pick Revert

702f2adf Merged PR 2: Get method for Ticket with Notes...

Description

Get method for Ticket with Notes

Show everything ▾

Add a comment...

RA Roberto Ajolfi completed the pull request 1/10/2020

✓ Approved by RA Roberto Ajolfi 1/10/2020

RA Roberto Ajolfi joined as a reviewer 1/10/2020

C# TicketEFController.cs /TicketingAPI/Controllers/TicketEFController.cs 1/10/2020

```
74 +         .Include(t => t.Priority)
75 +         .Include(t => t.State)
76 +         .Include(t => t.Notes)
77 +         .FirstOrDefault(t => t.Id == id);
```

Policies

Required

- ✓ 1 reviewer approved
- ✓ All comments resolved

Work Items

No related work items

Reviewers

RA Roberto Ajolfi Approved

Labels

Add label

Azure DevOps Repos (GIT)

The screenshot shows the Azure DevOps Repos (GIT) interface. The left sidebar navigation includes: Overview, Boards, **Repos** (selected), Files, Commits, Pushes, Branches, and Tags. The main content area displays the 'Ticketing Demo' repository under 'ajr1272 / Ticketing Demo / Repos / Files /'. The repository details show it's a 'master' branch with a status of 'succeeded' and a 'Clone' button. The 'Files' section lists the contents of the master branch, including a folder structure and several files: TicketingAPI, TicketingAPI.Test, .gitignore, .gitattributes, and TicketingAPI.sln. A detailed table below shows the commit history for each item.

Name ↑	Last change	Commits
TicketingAPI	Jan 10	ce5397fb Get method for Ticket with Notes...
TicketingAPI.Test	Oct 4, 2019	0e845eb2 Test project commented out. Ro...
.gitattributes	May 13, 2019	06e3e724 Add .gitignore and .gitattributes....
.gitignore	May 13, 2019	06e3e724 Add .gitignore and .gitattributes....
TicketingAPI.sln	Oct 4, 2019	0e845eb2 Test project commented out. Ro...

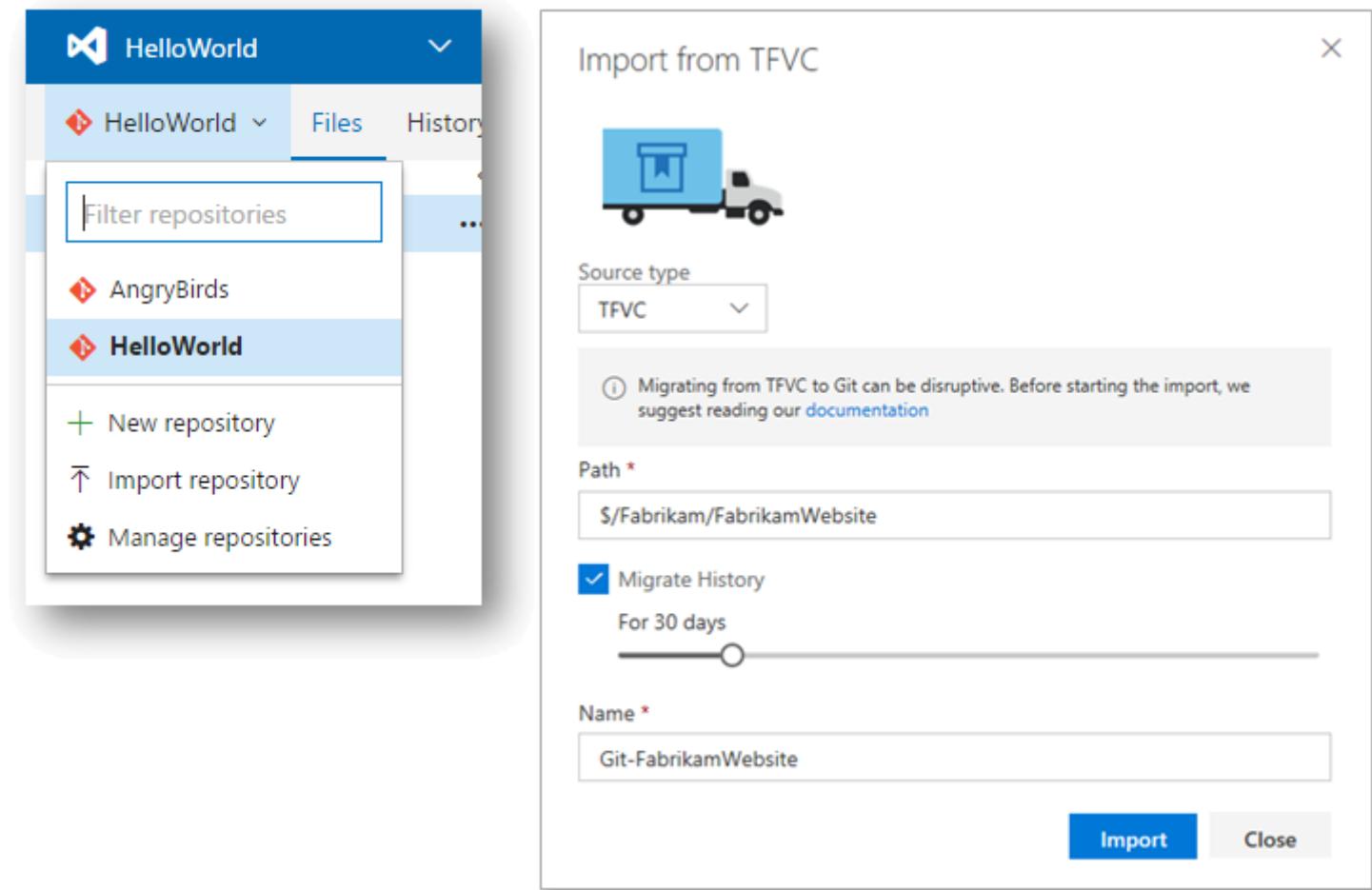
Migrazione da TFVC/TFS

It is a disruptive change that requires careful planning

- Tool e processi
- Binaries ed eseguibili
- Training del team

Si può migrare la history degli ultimi 180 giorni, ma è sconsigliato per via delle differenze tra GIT e TFVC

Max 1Gb di repository su un solo branch



Processo di build

Serie di passaggi (Task) da eseguire prima del rilascio

- Restore dei pacchetti (dipendenze), build del progetto, tests...

Funziona con pipeline pre-esistenti

- Jenkins, Octopus, Ansible...

Automatizzazione tutto il possibile

- Si può automatizzare tutto tramite PowerShell, Bash o similari...



Processo di build

The screenshot shows the Azure DevOps interface for managing pipelines. The left sidebar lists various project components: Overview, Boards, Repos, Pipelines (selected), Pipelines, Environments, Releases, Library, Task groups, Deployment groups, Test Plans, and Artifacts. The main area displays the YAML configuration for a pipeline named "Build ConsoleHelpers". The pipeline is triggered by a branch named "develop". It runs on an Ubuntu latest VM pool. The variables section specifies a build configuration of "Release". The steps section contains a restore task for dependencies, setting the command to "restore" and the projects to all ".csproj" files. It also includes a feed selection for "select" and a specific feed named "ajr1272". The final step is a dotnet build command with the configuration set to \$(buildConfiguration). The display name for this step is "Compilazione per Pull Request di ConsoleHelpers".

```
trigger:
- develop

pool:
  vmImage: 'ubuntu-latest'

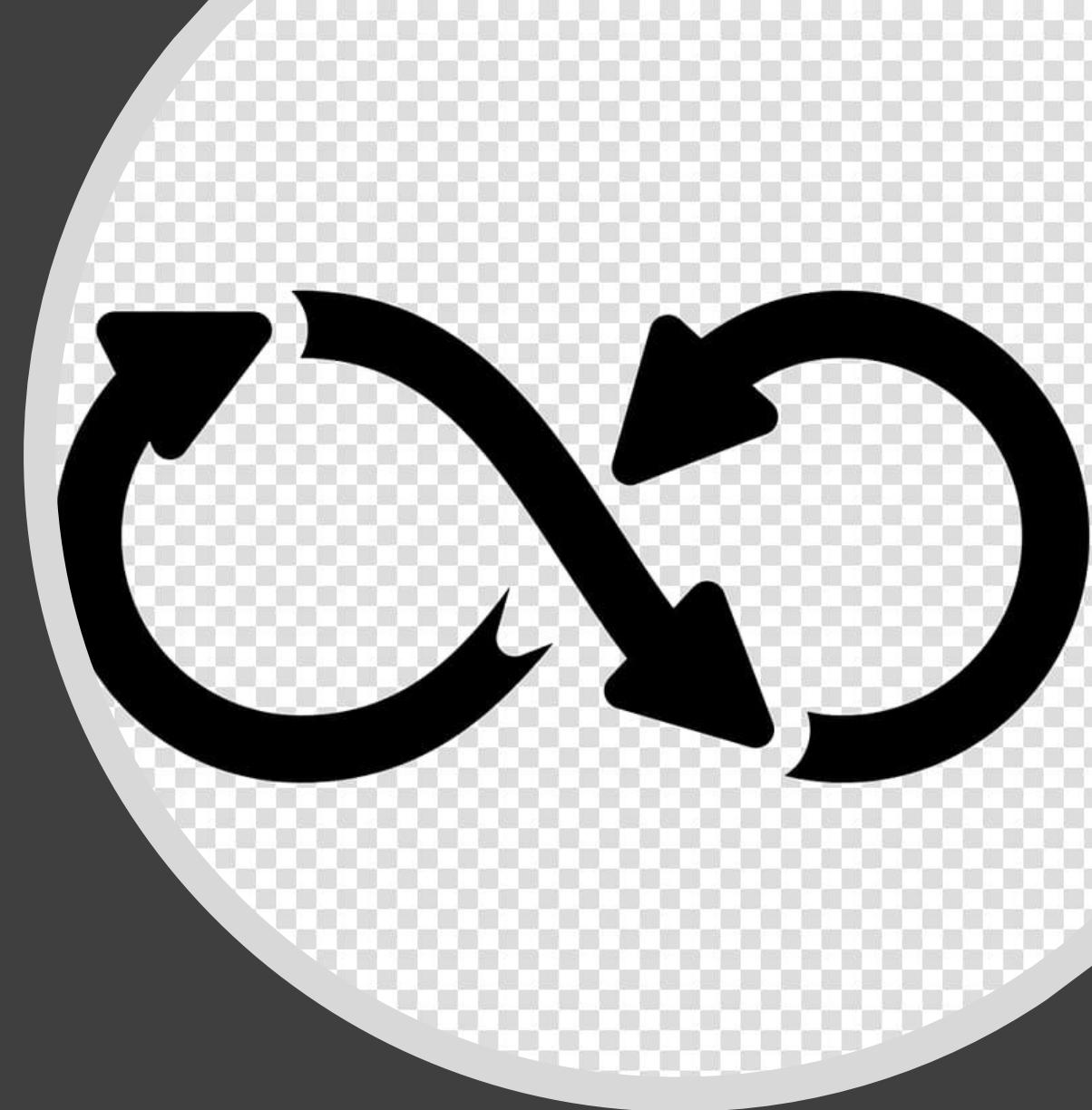
variables:
  buildConfiguration: 'Release'

steps:
  - task: DotNetCoreCLI@2
    displayName: 'Restore Dependencies'
    inputs:
      command: restore
      projects: '**/*.csproj'
      feedsToUse: 'select'
      vstsFeed: 'ajr1272'
      includeNuGetOrg: true
  - script: |
    cd ConsoleHelpers
    dotnet build --configuration $(buildConfiguration)
    displayName: Compilazione per Pull Request di ConsoleHelpers
```

Continuous Integration

È una pratica che si applica in contesti in cui lo sviluppo del software avviene attraverso un sistema di versioning

- Consiste nell'allineamento continuo (molte volte al giorno) dagli ambienti di lavoro verso un ambiente condiviso
- Proposto nel contesto dell' extreme programming (XP)
- Complementare al Test Driven Development
- L'integrazione continua è un processo che consiste nell'integrare presto e spesso, limitando le possibilità dell'inferno dell'integrazione
- La pratica cerca di minimizzare il lavoro inutile e risparmiare tempo



Continuous Integration

- Recuperare una copia del codice corrente sulla quale lavorare
- Altri sviluppatori fanno delle modifiche sul repository, la copia smette di rappresentare il codice del repository
- Le modifiche non si riflettono unicamente sul codice, ma anche sulle librerie ed altre risorse che possono creare dipendenze e potenziali conflitti

Più è lungo il tempo durante il quale un branch di codice rimane scaricato, maggiore è il rischio di conflitti di integrazione quando il branch viene reintegrato nel repository

Continuous Integration

- Quando gli sviluppatori vogliono sottomettere il codice, devono prima aggiornare le proprie copie locali del codice con i cambiamenti che sono stati fatti al repository dal momento in cui loro hanno aggiornato la copia locale
- Più modifiche sono state fatte e più lavoro è necessario fare prima di sottomettere le proprie modifiche
- In uno scenario estremamente negativo, gli sviluppatori potrebbero dover annullare le proprie modifiche e rifarle da capo

Continuous Integration - Principi

- La build è un processo che trasforma il codice sorgente in un artefatto.
- Deve essere possibile eseguire la compilazione e la creazione dei pacchetti da mettere sui server in modo completamente automatizzato, senza alcun intervento umano
- Ogni volta che il codice sorgente viene buildato ed impacchettato, è importante che vengano eseguiti dei test sul sorgente affinché la qualità del codice venga tenuta sotto controllo ed eventuali bug vengano scoperti il prima possibile

Continuous Integration - Principi

- È inutile «buildare» e testare il sorgente se gli sviluppatori sviluppano codice sui propri computer senza sincronizzarsi spesso col codice scritto da altri.
- È importante però che il codice sia compilabile ed abbia superato tutti i test.
- Ogni modifica al codice sorgente condiviso potrebbe generare dei bug e quindi compilare e testare subito dà la possibilità di intervenire immediatamente su eventuali bug.
- Se un commit di uno sviluppatore è tale da far fallire un test, bisogna intervenire immediatamente prima che si crei una sequenza di errori successivi a valanga.

Continuous Integration - Processo

- Mantieni un repository del codice sorgente
 - Tutti eseguono commit alla baseline tutti i giorni
 - Ogni commit fa partire una build
- Automatizza il build
 - Rendi il build auto-testante
 - Fai in modo che il build sia veloce
 - Ognuno può vedere i risultati dell'ultimo build
- Automatizza il deployment
 - Eseguire i test in un clone dell'ambiente di produzione
- Fai in modo che sia facile prendere le ultime versioni dei pacchetti

Continuous Integration - Principi

- Le build devono essere rapide!
- Utilizzare un ambiente di staging: clone dell'ambiente di produzione
- Tutti gli ambienti devono essere facilmente allineabili
- Il deploy deve essere immediatamente reso a disposizione per tutti i potenziali utilizzatori
- Predisposizione di Test automatici

Testing

- E' (*dovrebbe essere*) alle base dello sviluppo del software
- La qualità del software deve essere garantita da processi e modelli di validazione
- La qualità del software non può essere relegata al singolo programmatore
- Definire precise attività di test
- Definire una serie di momenti in cui effettuare il test



Tipologie di test

Unit Test

L'obiettivo è validare i singoli componenti: di esclusiva a competenza del programmatore

Behavioral Test

L'obiettivo è validare un «caso d'uso»

Integration Test

L'obiettivo è validare le interazioni di diverse componenti singolarmente consistenti.

Test di accettazione

Imposto dal cliente, verifica che il programma soddisfi le richieste del cliente stesso

Test di regressione

Verifica che non si siano introdotti errori in versioni successive



Testing: qualità del software



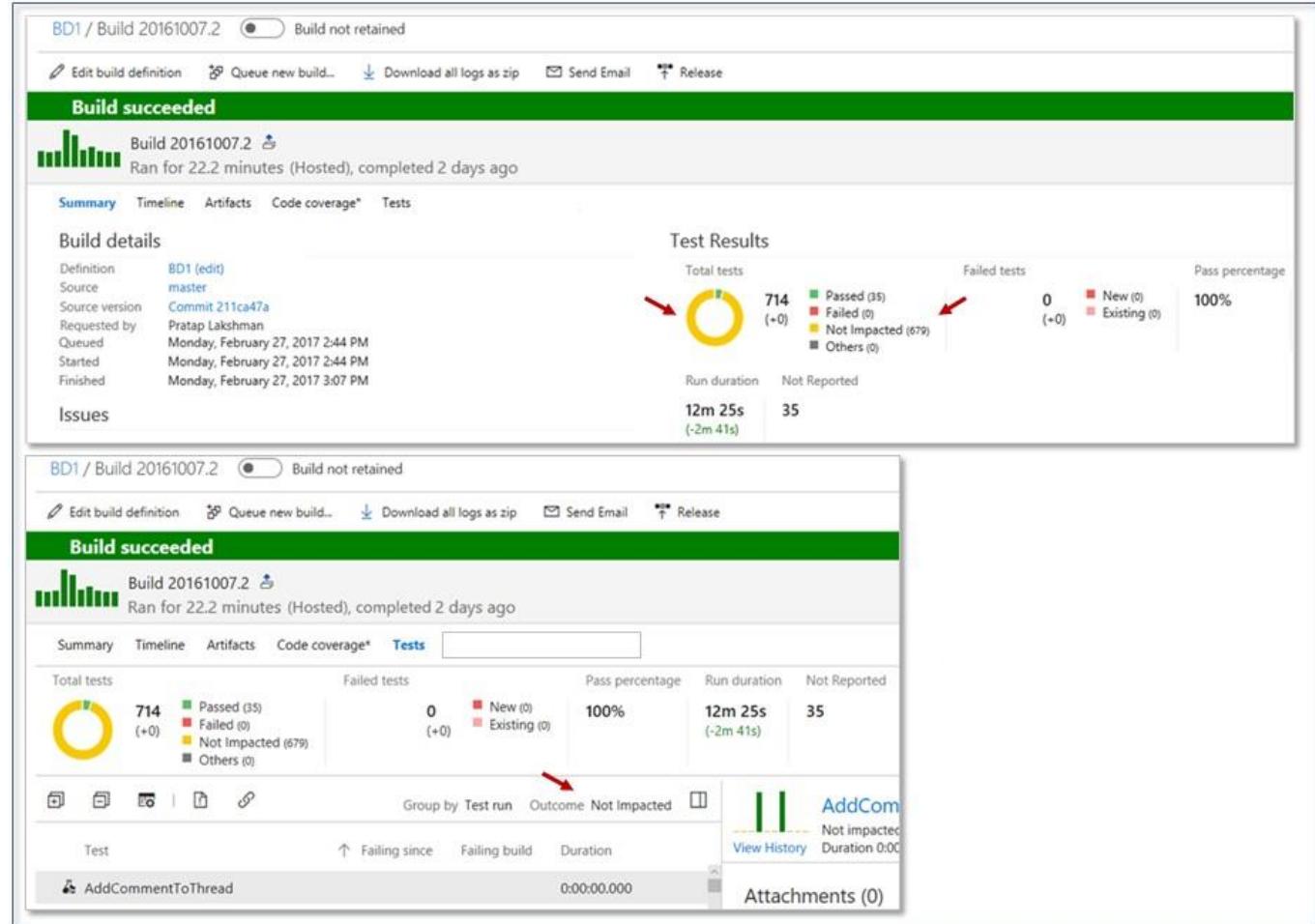
Test Plan

E' un documento che disciplina lo svolgimento dei test per un determinato sistema/applicazione

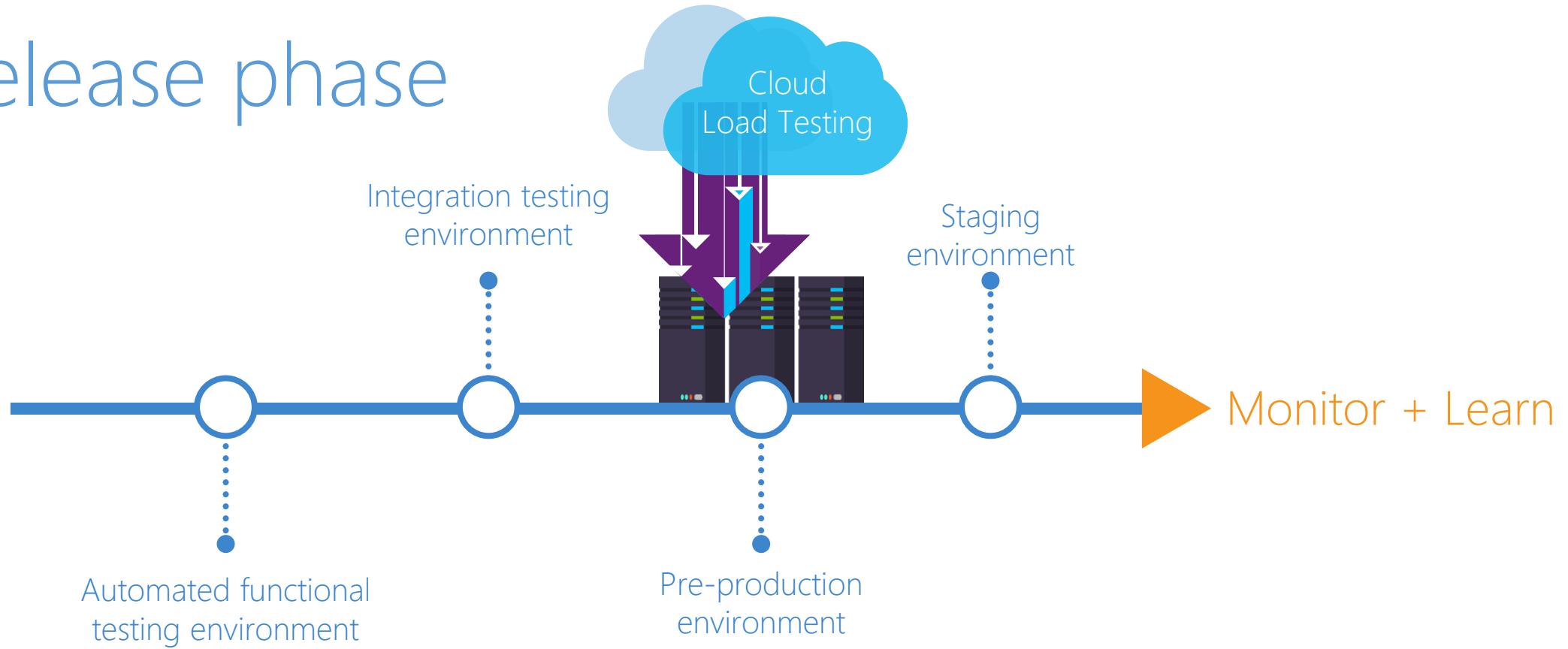
- la specifica dei moduli che devono essere testati
- il confine funzionale dell'applicazione
- la matrice di copertura dei test
- i criteri di sospensione e ripresa dei test
- i criteri di Pass/Fail
- i deliverables della fase di test
- le necessità di sviluppo di stub/drivers
- la descrizione dell'ambiente hardware/software
- eventuali tool utilizzati per la Test Automation

Azure DevOps Continuous Testing

Integrazione
dell'esecuzione dei Test
nelle pipeline di build
(esecuzione ad ogni
build)



Release phase



- Continuous Delivery
- Deployments in parallel
- Deploy in datacenter in cloud o on-premises



Continuous deployment

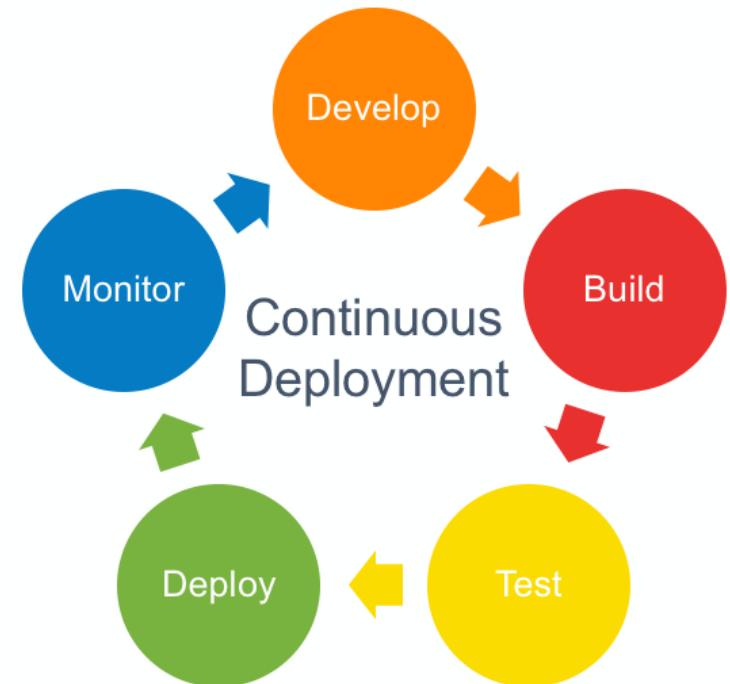
Si rilascia continuamente

Non è importante l'ambiente

- Può essere sviluppo, QA come produzione

E se le feature non sono complete?

- Feature Flags (o feature toggle)!
- Storage delle feature complete dentro un database
- Abilitazione tramite flag, anche su PROD



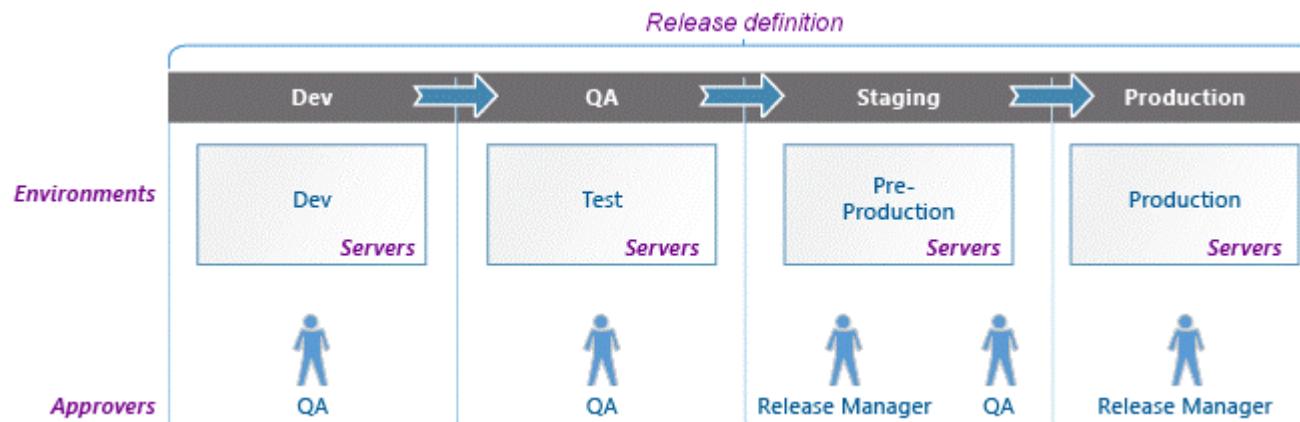
Release definition

Una release definition è una collezione di ambienti diversi

Un ambiente specifica come e dove deve essere effettuato il deployment

- Può essere un App Store, un servizio di Azure, un IIS on-premise...

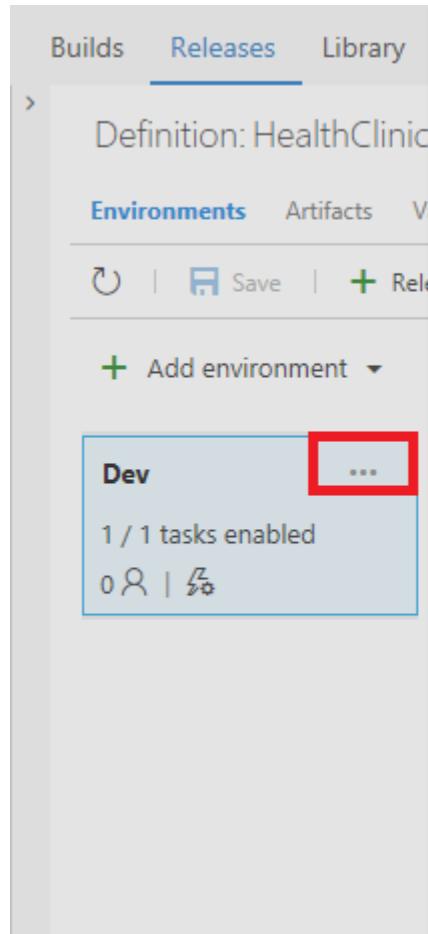
Il deployment è fatto tramite una serie di task



Deployment condizionale

Ogni deployment può essere soggetto a determinate condizioni:

- Deploy manuale
- Deploy schedulato ad orari specifici
- Deploy automatico dopo il deployment dell'ambiente precedente



Configure - 'Dev' environment

Approvals Variables **Deployment conditions** General

Trigger

Define the trigger that will start deployment to this environment.

No automated deployment

After release creation

Scheduled

After successful deployment to another environment

Options

Define behavior when multiple releases are waiting to be deployed on this environment. [\(1\)](#)

Allow multiple releases to be deployed at the same time

Allow only one active deployment at a time

Deployment approval

Ogni deployment deve essere autorizzato:

- Automaticamente
- Attraverso utenti specifici

Select the users who can approve or reject the deployments to this environment.

Pre-deployment approver

- Automatic
 Specific Users

craig

	Craig Campbell	craig109@outlook.com
---	----------------	----------------------

Post-deployment approver

- Showing 1 result
 Specific Users

A pre-deployment approval is pending for 'Dev' environment. [Approve or Reject](#)

Details		
Release-31		
Manually created by Srivatsa just now		
CI.Web.Master / 622 (Build) ↗ master		
Environments		
Environment	Actions	Deployment status
Dev	...	 NOT DEPLOYED
QA	...	NOT DEPLOYED
Issues		
No issues reported in this release.		

 Pre-deployment approval pending
on Srivatsa
just now (Reassign)

Type comments here...

Defer this deployment to 2/20/2017 (UTC) Coordinated Universal Time 6:30 PM

[Approve](#) [Reject](#)

Continuous Deployment

VodafoneT Dashboards Code Work Build & Release Test | ⚙

Builds Releases Library Task Groups Explorer

Search work items ⚙ MT

Magellan.Infrastructure.Deployment / Magellan.Infrastructure.Deployment-9

Summary Environments Artifacts Variables General Commits Work items Tests Logs History

Release Definitions

- All release definitions
- Magellan.Creation.CD
- Magellan.Enrichment.CD
- Magellan.Geocoding.CD
- Magellan.Infrastructure.Dep... ...
- Magellan.Ingestion.CD
- Magellan.Periodic.CD
- Magellan.Raw.CD

Search release definitions... +

Deploy Save Abandon

Add environment

Magellan.Infrastructure Deployment

Raw Periodic Ingestion +3

Magellan.Infrastruc... ✓ +1

Magellan.Infrastruc... ✓ +1

Magellan.Infrastruc... +1

Magellan.Infrastruc... +1

Magellan.Infrastruc... +1

View all releases for Magellan.Infrastructure.Deployment release definition

CI/CD automatiche

Continuous Delivery
Tools for Visual Studio
2019

Supporto per containers
Supporto per Azure
Service Fabrics

<http://aka.ms/CD4VS>

Configure Continuous Delivery

Microsoft [x]

Select a Repository Branch

Repository: `builddemoaspnetcorelinuxcdtest`

Branch: `master`

Select Target Azure Resources

Subscription: `VS [REDACTED]`

Host Type: `App Service (Linux)`

App Service: `BuildDemoASPNETCoreLinuxCDTest`

Container Registry: `AhmedBuildACR`

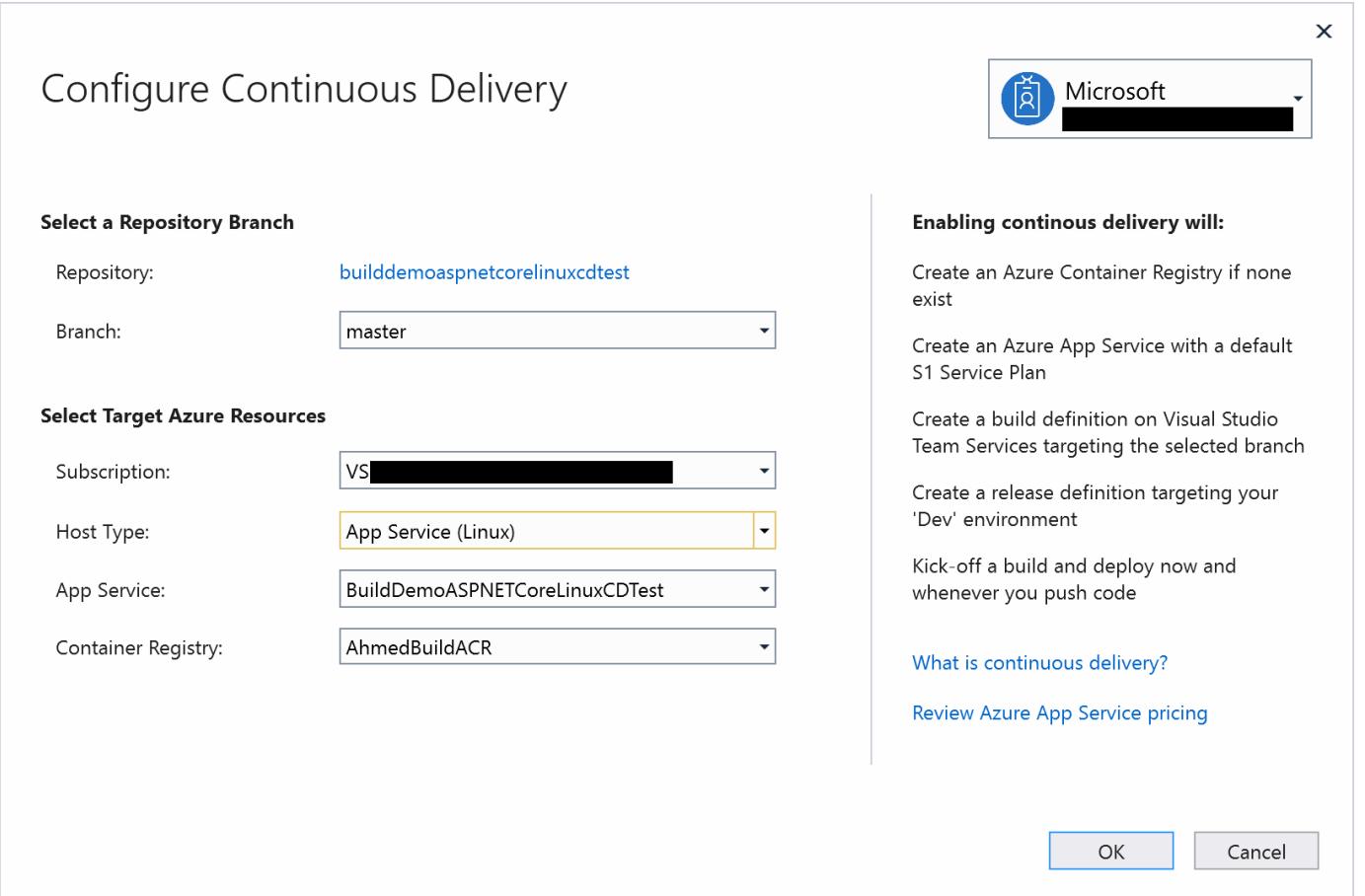
Enabling continuous delivery will:

- Create an Azure Container Registry if none exist
- Create an Azure App Service with a default S1 Service Plan
- Create a build definition on Visual Studio Team Services targeting the selected branch
- Create a release definition targeting your 'Dev' environment
- Kick-off a build and deploy now and whenever you push code

[What is continuous delivery?](#)

[Review Azure App Service pricing](#)

OK Cancel



Package Management

Perché?

- Si inizia da un team, che poi cresce...
- La codebase cresce...
- I membri del team iniziano a “pestarsi i piedi”

Componentizzazione

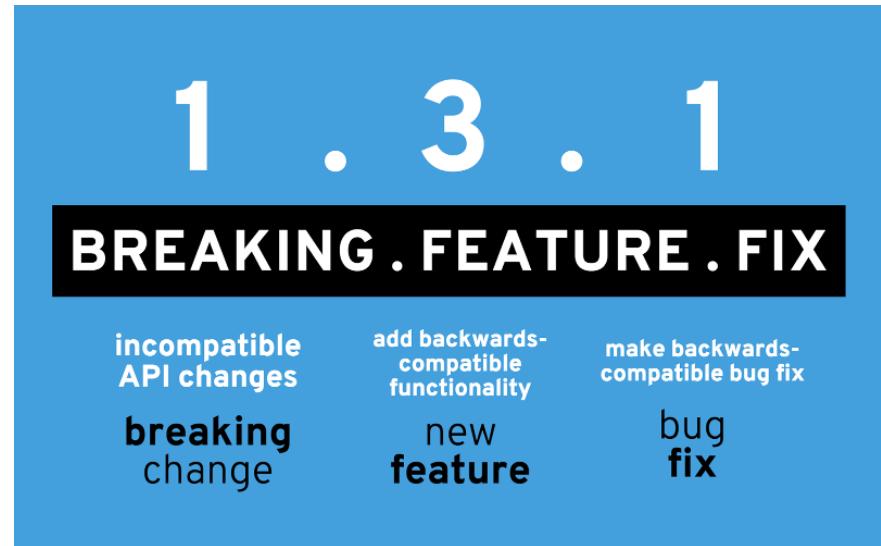
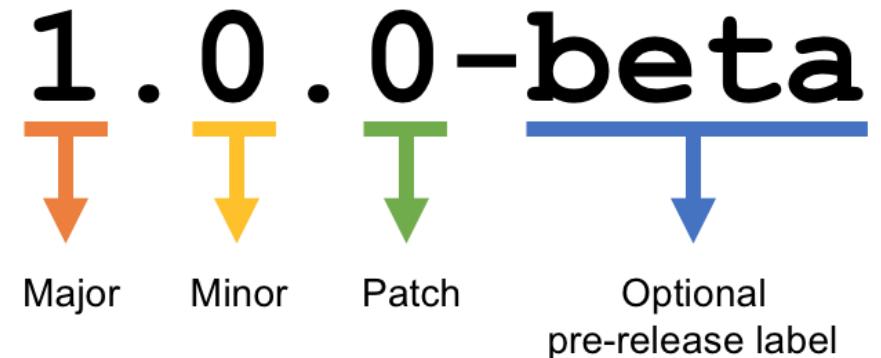
- Non è pensabile mettere “dll” sotto source control
- Si fa già oggi dentro Visual Studio, si chiamano reference...
- Meglio produrre pacchetti di NuGet anziché “dll”...



Semantic Versioning

«*Nel mondo della gestione del software esiste un posto terribile chiamato "inferno delle dipendenze". Più cresce il sistema, più crescono i pacchetti da integrare in esso e più è probabile ritrovarsi, un giorno, in questa valle di lacrime.*» - semver.org

Non è nulla di innovativo, ma una standardizzazione di un processo di versionamento che impone un po' di organizzazione in un sistema complesso come possono essere i microservices



Package Management

Package Management rende facile il discovery, l'installazione e la pubblicazione dei pacchetti/reference

Supportato sia in Azure DevOps (tramite extension dal marketplace) sia su TFS 2017 (built-in)

Package type	Components	Package Management Support
NuGet	.NET Assembly	Supportato
npm	Node.js	Supportato
Maven	POM e JAR	In arrivo a brevissimo...
Bower	Web front-end (es. HTML,CSS, Javascript)	Non supportato

Feed

I feed sono un modo per organizzare i pacchetti in modo logico e conveniente per gli sviluppatori

Producono un url che sarà l'endpoint per la ricerca dei pacchetti

Get packages using Visual Studio

Package source URL

[View All packages](#) 

https://enhancealm.pkgs.visualstudio.com/_packaging/ALMTraining/nuget/v3/index.json 

 Connect to feed

Create new feed 

Feed name

Description

Who can read

Only PackageManagement-NuGet Team members can view packages
 Everyone in your account can view packages

Who can contribute

PackageManagement-NuGet Team members can add packages
 Project Collection Build Service can add packages

Include upstream sources

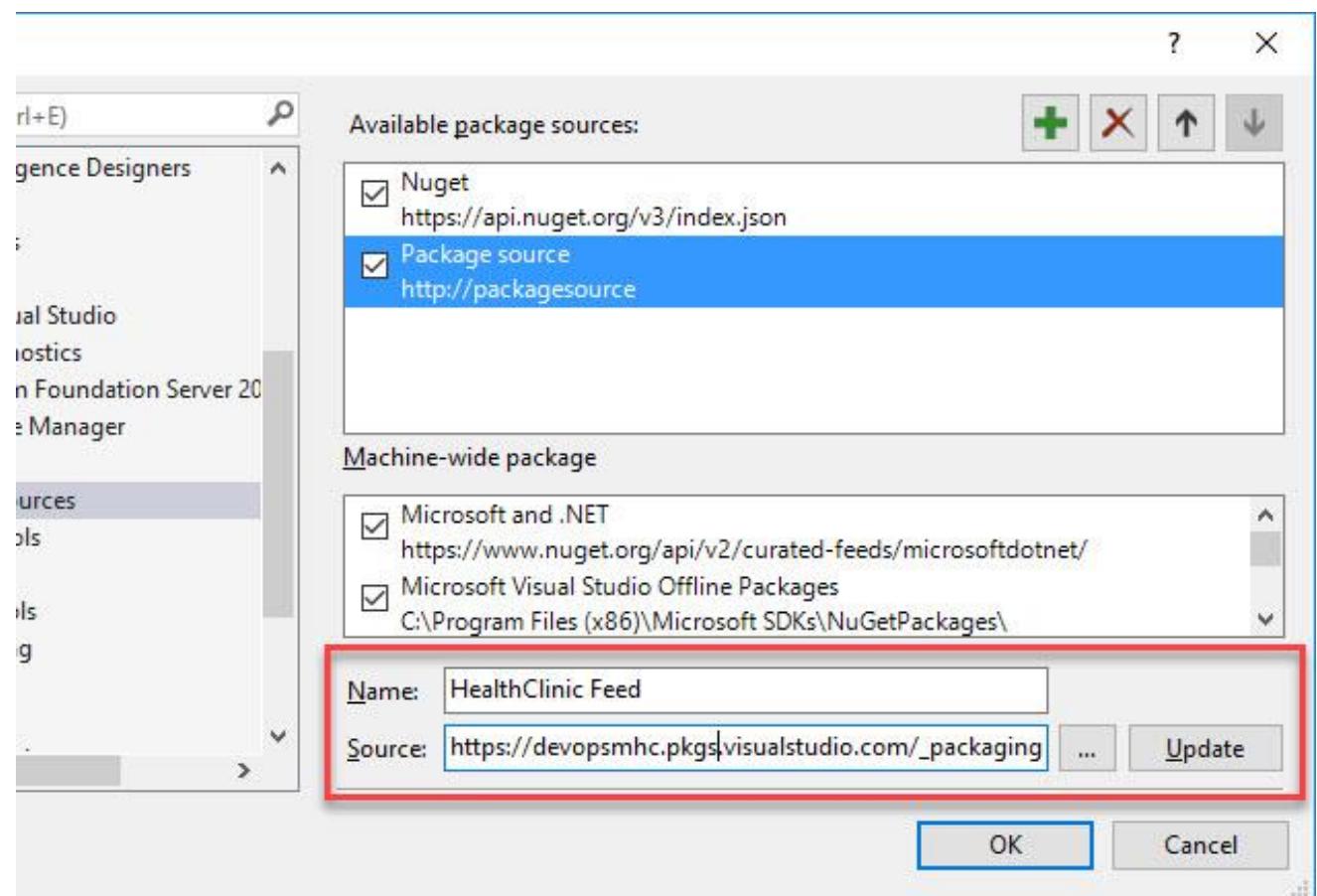
Include packages from npmjs.org in this feed

[Create](#) [Cancel](#)

Integrazione con Visual Studio

Dalle opzioni di Visual Studio è sufficiente aggiungere una nuova sorgente di NuGet.

L'url è il feed ottenuto dal package management



© 2022 iCubed Srl



La diffusione di questo materiale per scopi differenti da quelli per cui se ne è venuti in possesso è vietata.

iCubed s.r.l.

Piazza Duca D'Aosta, 12 20124 MILANO

Phone: +39 02 57501057

P.IVA 07284390965

