imola informatica

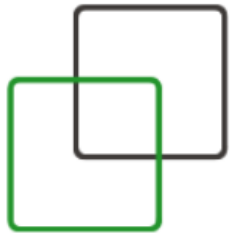
Università degli Studi di Padova

Seminario Unipd 22-11-2019

corso di INGEGNERIA del SOFTWARE - prof. Tullio Vardanega

Il Gruppo Imola

Cosa facciamo



Architettura

Progettiamo architetture agili e flessibili.
Guidiamo le evoluzioni di sistemi informativi complessi.

Servizi di advisory

Sperimentiamo tutte le novità più promettenti.
Selezioniamo le tecnologie più adatte al tuo mondo.

Gestione della conoscenza

Disegniamo modelli per gestire dati e informazioni.
Realizziamo servizi innovativi per migliorare il tuo business.

Sicurezza e privacy

Educhiamo le organizzazioni alla cultura della sicurezza.
Creiamo nuove consapevolezze su dati e GDPR.

Delivery

Sviluppiamo software per qualsiasi esigenza.
Garantiamo un risultato di valore per la tua azienda.

12 Factor Application

I. ***Codebase*** - *Una sola codebase sotto controllo di versione, tanti deployment*

II. ***Dipendenze*** - *Dipendenze dichiarate e isolate*

III. ***Configurazione*** - *Memorizza le informazioni di configurazione nell'ambiente*

IV. ***Backing Service*** - *Tratta i backing service come "risorse"*

V. ***Build, release, esecuzione*** - *Separare in modo netto lo stadio di build dall'esecuzione*

VI. ***Processi*** - *Esegui l'applicazione come uno o più processi stateless*

VII. ***Binding delle Porte*** - *Esporta i servizi tramite binding delle porte*

VIII. ***Concorrenza*** - *Scalare attraverso il process model*

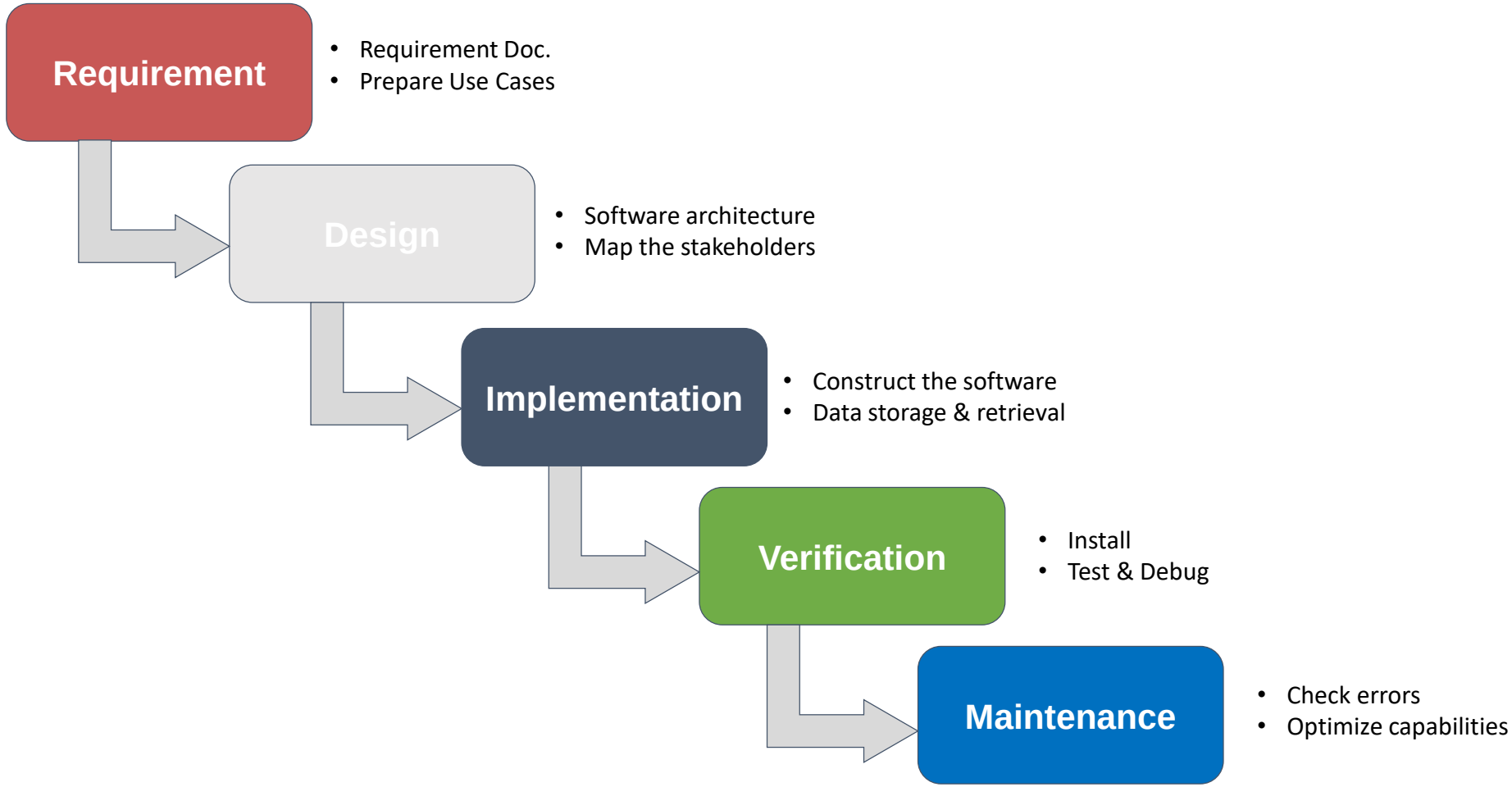
IX. ***Rilasciabilità*** - *Massimizzare la robustezza con avvii veloci e chiusure non brusche*

X. ***Parità tra Sviluppo e Produzione*** - *Mantieni lo sviluppo, staging e produzione simili il più possibile*

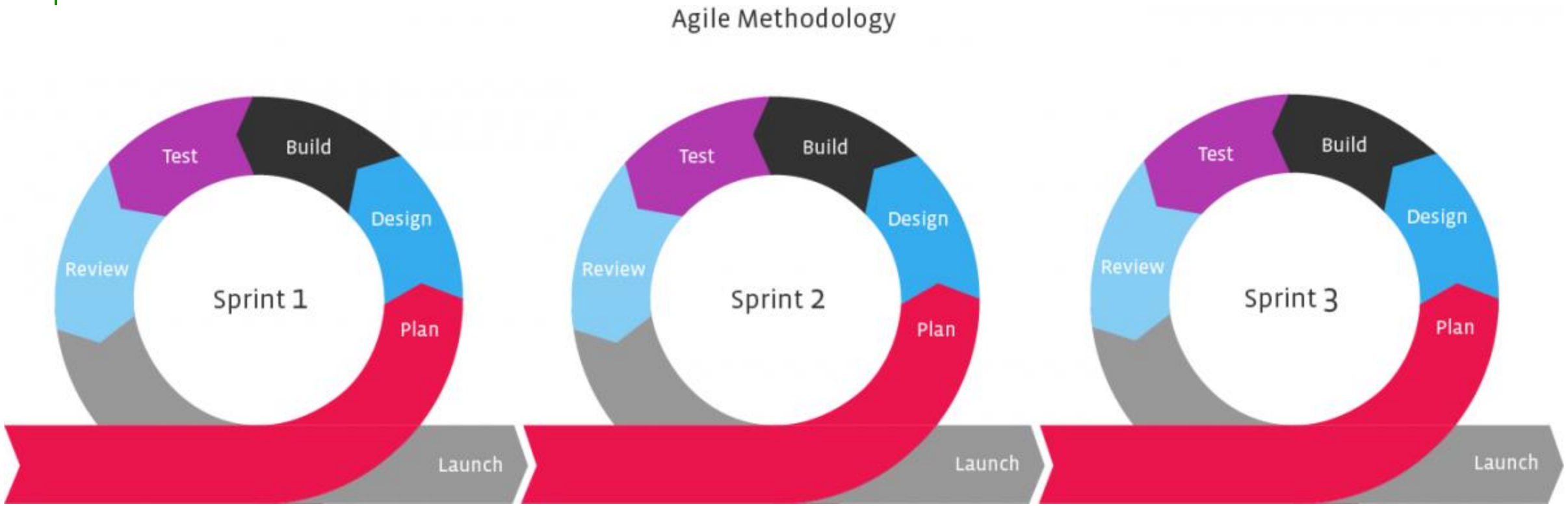
XI. ***Log*** - *Tratta i log come stream di eventi*

XII. ***Processi di Amministrazione*** - *Esegui i task di amministrazione/management come processi una tantum*

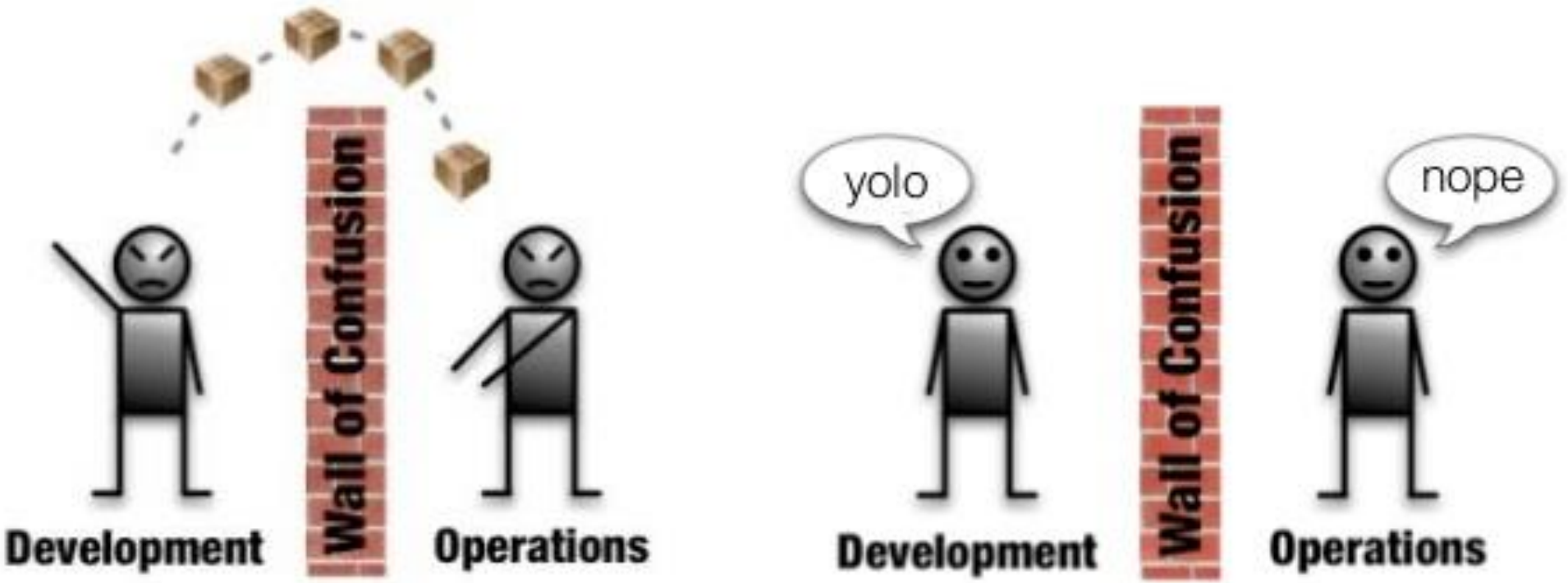
Waterfall


Requirement
• Requirement Doc.
• Prepare Use Cases
Design
• Software architecture
• Map the stakeholders
Implementation
• Construct the software
• Data storage & retrieval
Verification
• Install
• Test & Debug
Maintenance
• Check errors
• Optimize capabilities

Agile

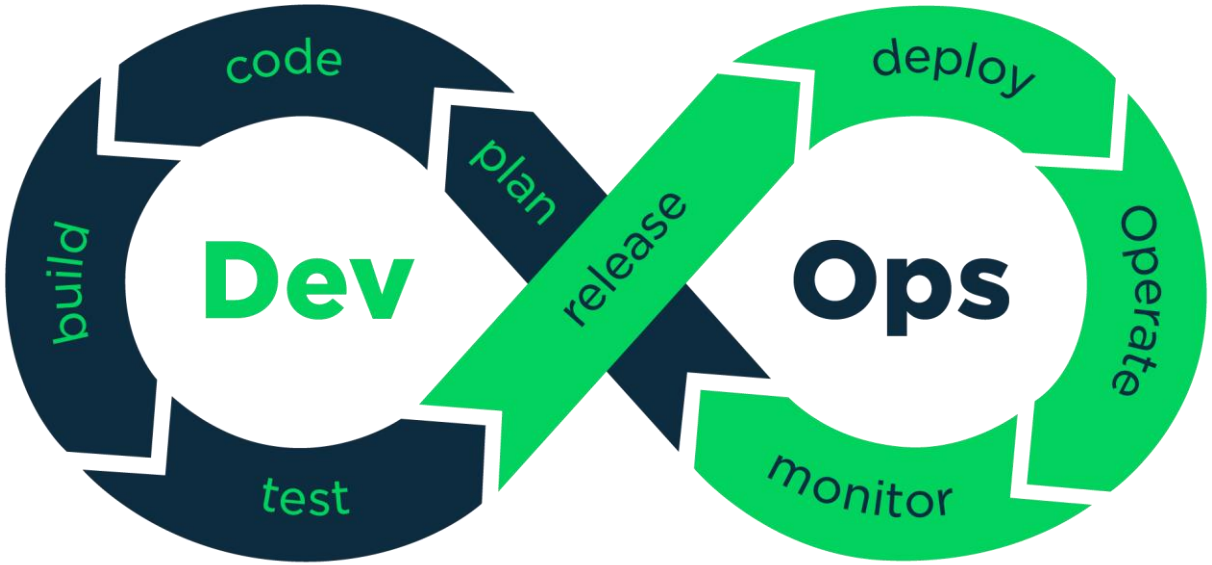

Agile Methodology
Test
Build
Design
Plan
Review
Sprint 1
Launch
Test
Build
Design
Plan
Review
Sprint 2
Launch
Test
Build
Design
Plan
Review
Sprint 3
Launch

The Wall of Confusion



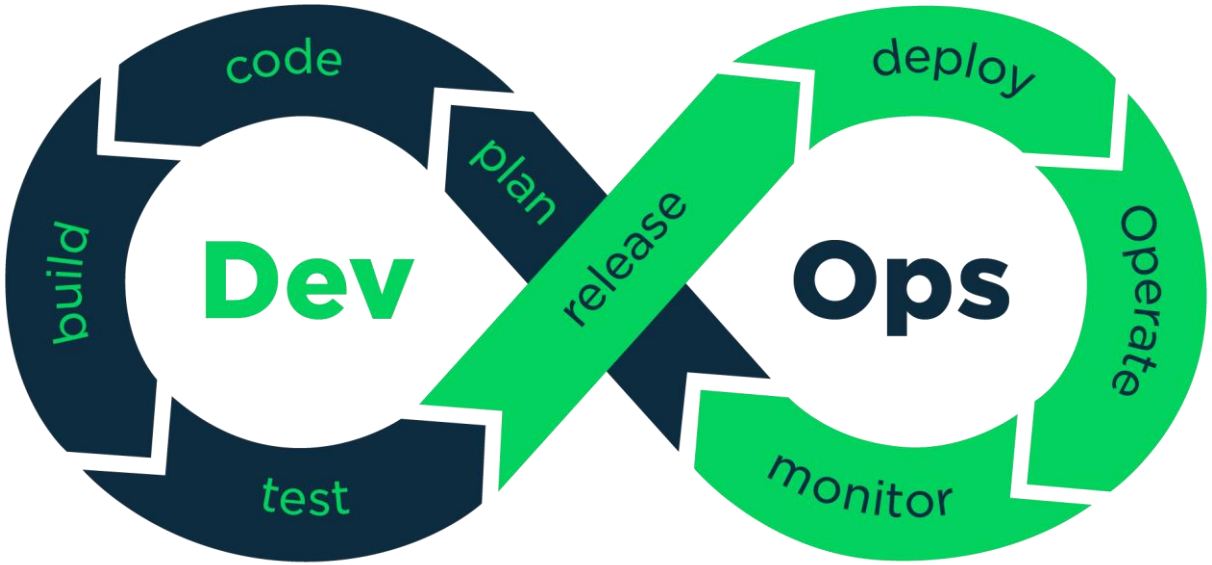
@bridgetkromhout

DevSecOps – obiettivi e benefici



- **Automazione**
- **Feedback continuo (Fail fast)**
- **Maggior Collaborazione**
- **Maggior Velocità di sviluppo e rilascio**
- **Ripetibilità e Affidabilità**

DevSecOps – pratiche abilitanti



- **Continuous Integration**
- **Continuous Delivery**
- **Continuous Testing**

Continuous Integration

https://martinfowler.com/articles/continuousIntegration.html

*"Continuous Integration is a **software development practice** where members of a team **integrate their work frequently**, usually each person integrates at least daily - leading to multiple integrations per day. Each integration is verified by an **automated build (including test)** to **detect integration errors as quickly** as possible."*

Continuous Integration è una ***pratica di sviluppo software*** dove i membri di un team integrano il loro lavoro frequentemente, possibilmente più volte al giorno. Ad ogni integrazione viene effettuata una ***build automatica (comprensiva di test)*** al fine di ***identificare eventuali errori al più presto possible***.

Continuous Integration - Pratiche

- *Maintain a Single Source Repository*
- *Automate the Build*
- *Make Your Build Self-Testing*
- *Everyone Commits To the Mainline Every Day*
- *Every Commit Should Build the Mainline on an Integration Machine*
- *Fix Broken Builds Immediately*
- *Keep the Build Fast*
- *Test in a Clone of the Production Environment*
- *Make it Easy for Anyone to Get the Latest Executable*
- *Everyone can see what's happening*

Continuous Delivery

https://martinfowler.com/bliki/ContinuousDelivery.html

*"Continuous Delivery is a **software development discipline** where you build software in such a way that the software can be released **to production at any time**."*

Continuous Delivery è una ***pratica di sviluppo software*** dove la build del software viene effettuata in maniera tale da poter effettuare un rilascio ***in produzione in qualunque momento***.

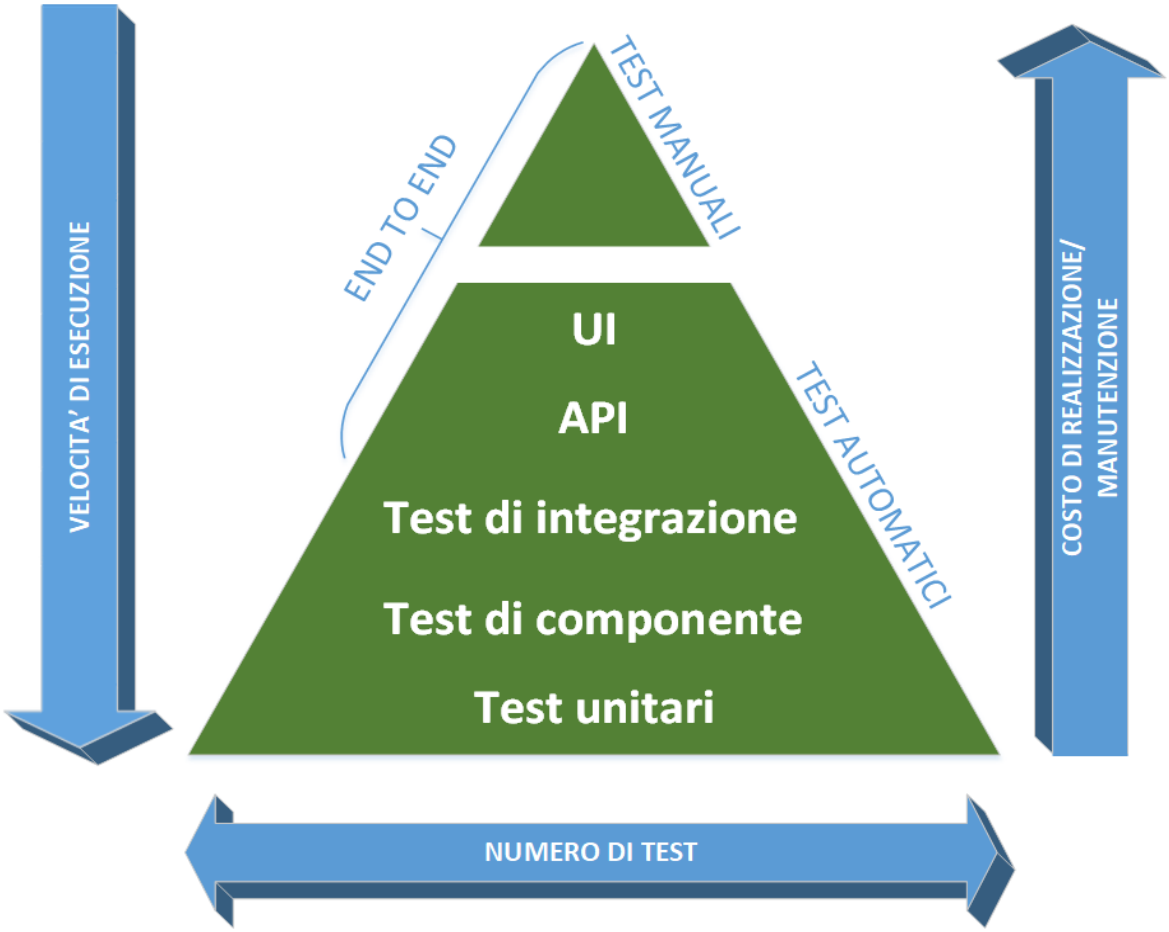
Continuous Delivery - Pratiche

Jez Humble "Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation"

- *Create a repeatable, reliable process for releasing software*
- *Automate (almost) everything*
- *Keep everything in version contron*
- *If it hurts, do it more frequently, and bring the pain forward*
- *Build quality in*
- *Done means released*
- *Everybody is responsible for the delivery process*
- *Continuous Improvement*

Continuous Testing

Continuous Testing è il processo di esecuzione di ***test automatici come parte del processo di integrazione e delivery del software*** *in modo tale da* ***ottenere feedback immediati*** *associati ad eventuali malfunzionamenti o bug.*



Continuous Testing - Pratiche

I test automatici devono essere:

- ***Atomici*** - devono essere eseguiti in qualsiasi ordine, non devono condividere uno stato con gli altri e non devono cambiare l'ambiente influendo sugli altri test
- ***Ripetibili***
- ***Parallelizzabili*** - (nice to have) in modo da impiegare meno tempo per l'esecuzione

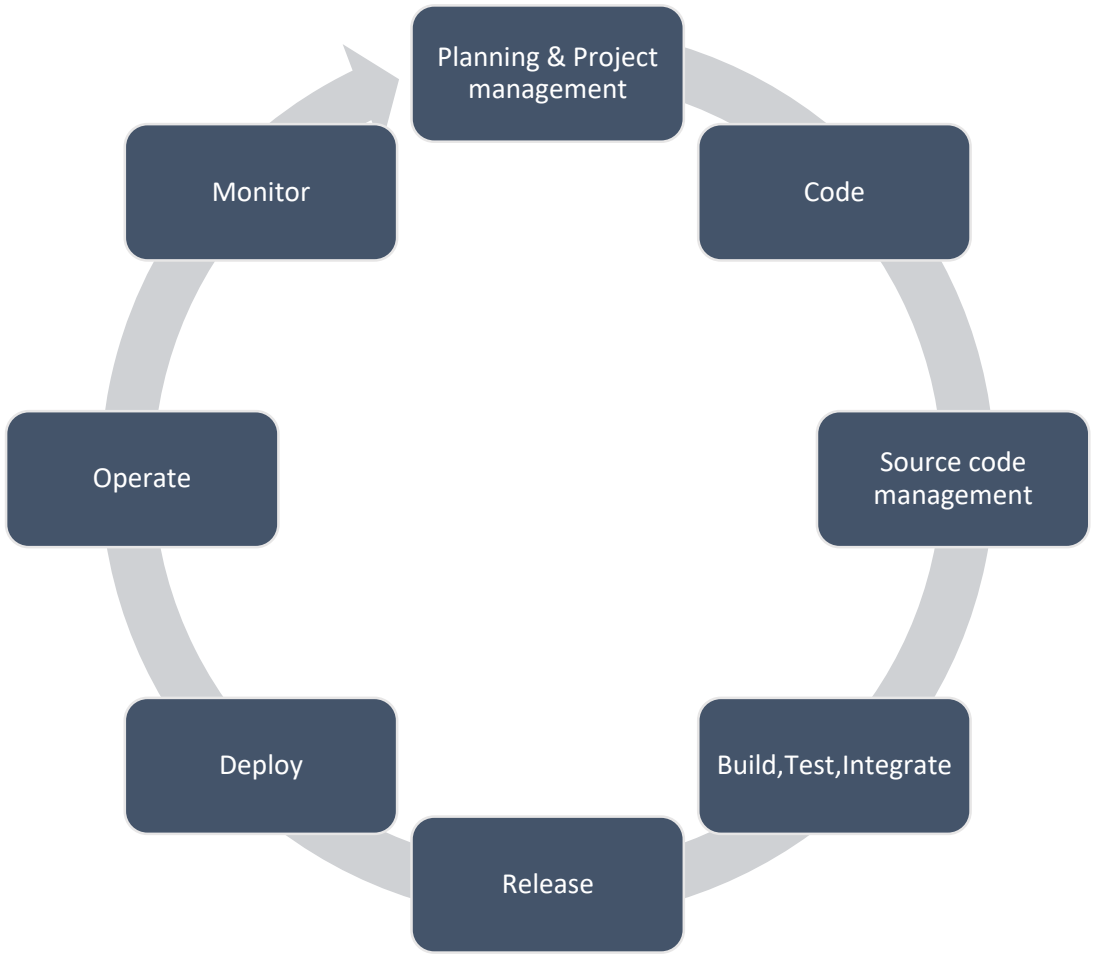
Possono essere realizzati in 2 modalità:

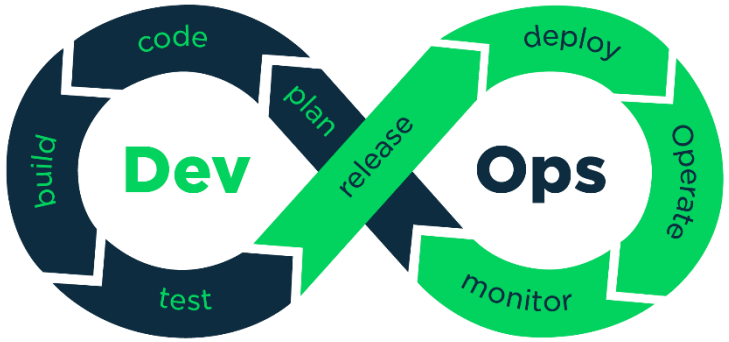
- ***Whitebox***: si basa sul fatto che la struttura del codice testato è conosciuta e si riescono a scrivere dei test per coprire tutti i flussi di esecuzione possibili

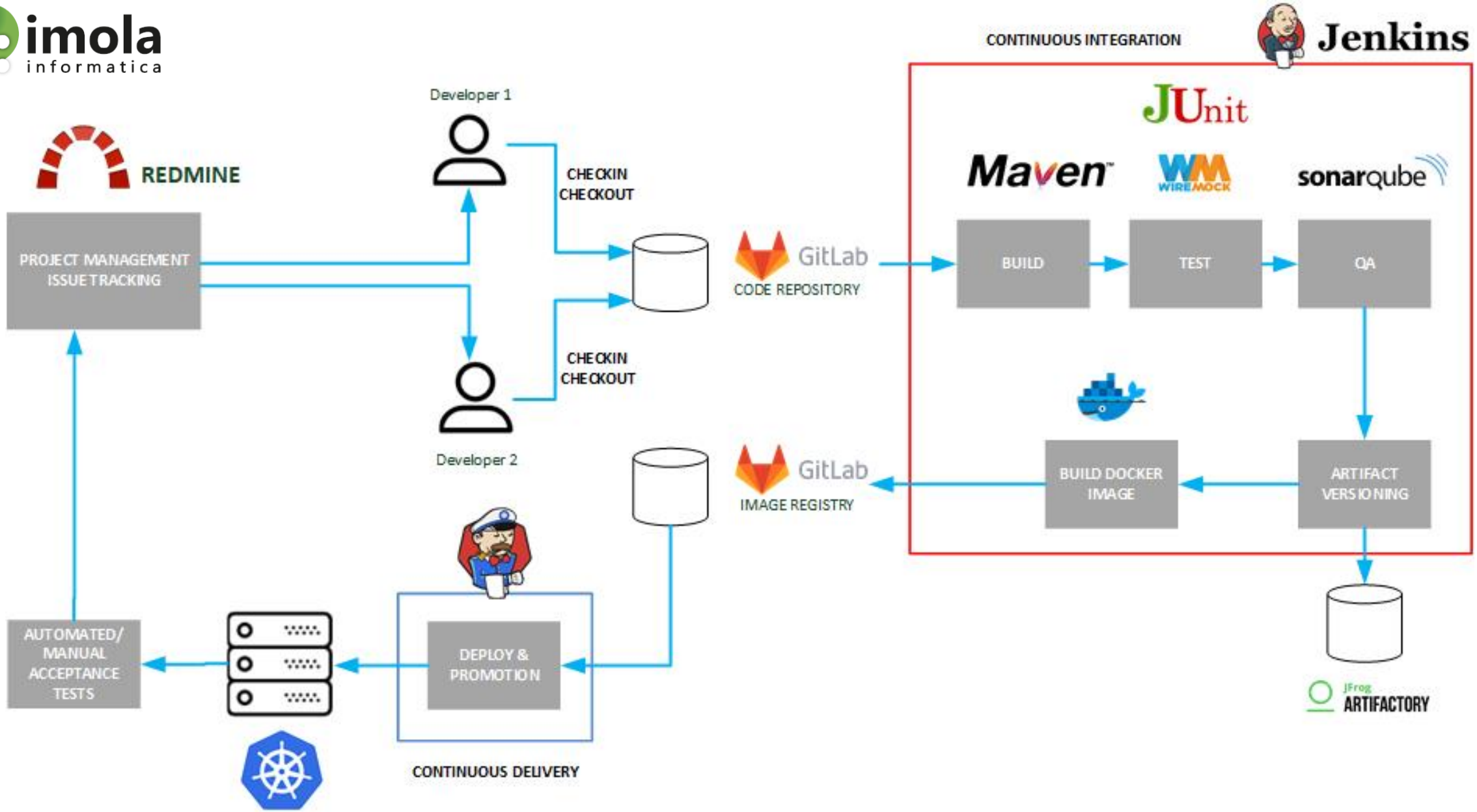
- ***Blackbox***: l'implementazione effettiva non è conosciuta ma si testa solo in base al comportamento atteso visibile all'esterno (output).

La creazione di una valida ed efficace suite di test prevede la scrittura di numerosi test a diversi livelli di granularità seguendo lo schema a «piramide» (vedi slide precedente)

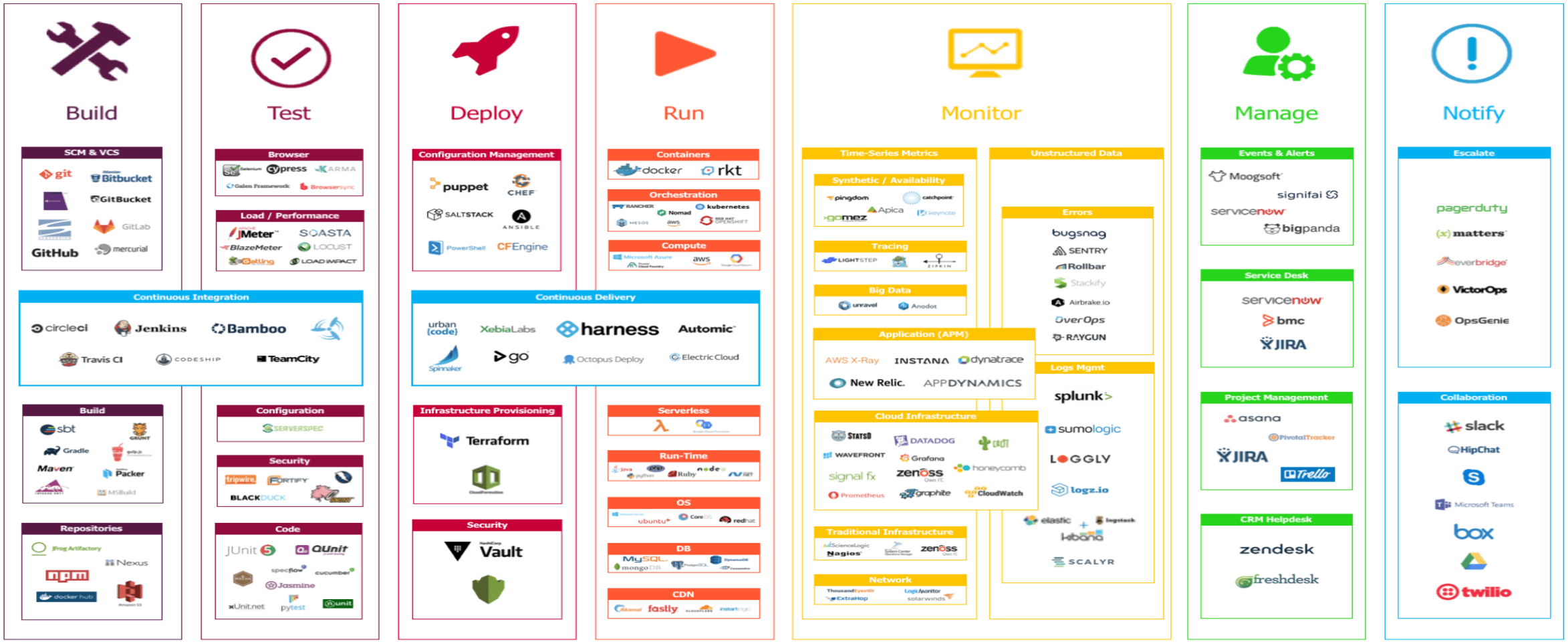
RECAP


Planning & Project management
Code
Source code management
Build,Test,Integrate
Release
Deploy
Operate
Monitor


code
plan
build
test
release
deploy
Operate
monitor
Dev
Ops


CONTINUOUS INTEGRATION
Jenkins
JUnit
Maven
WIREMOCK
sonarqube
REDMINE
Developer 1
CHECKIN
CHECKOUT
PROJECT MANAGEMENT
ISSUE TRACKING
GitLab
CODE REPOSITORY
BUILD
TEST
QA
CHECKIN
CHECKOUT
Developer 2
GitLab
IMAGE REGISTRY
BUILD DOCKER
IMAGE
ARTIFACT
VERSIONING
AUTOMATED/
MANUAL
ACCEPTANCE
TESTS
DEPLOY &
PROMOTION
JFrog
ARTIFACTORY
CONTINUOUS DELIVERY

DevOps Lifecycle Mesh



harness

Continuous Delivery As–A-Service

www.harness.io

Wireless Positioning Technologies

- **Metodologie di posizionamento di base**
  - Navigazione stimata (Dead reckoning)
  - Proximity sensing (Cell-ID)
  - Trilaterazione (TOA)
  - Multitrilaterazione (TDOA)
  - Triangolazione (AOA)
- **Sistemi di posizionamento satellitare**
  - GPS
  - Assisted GPS (A-GPS)

Metodologie di base (I)

- Il **dead reckoning** è il processamento della posizione di un punto mobile sulla base di:
  - Posizione determinata in precedenza
  - Velocità e accelerazione note
  - Direzione di movimento
  - Tempo trascorso
  - Distanza percorsa

- Nel **proximity sensing**, la posizione del punto mobile è ricavata sulla base delle coordinate si stazioni, tracciando il segnale che viene trasmesso da esse (**cell ID**). Ogni stazione tramette con un proprio pattern di segnale.

Metodologie di base (II)

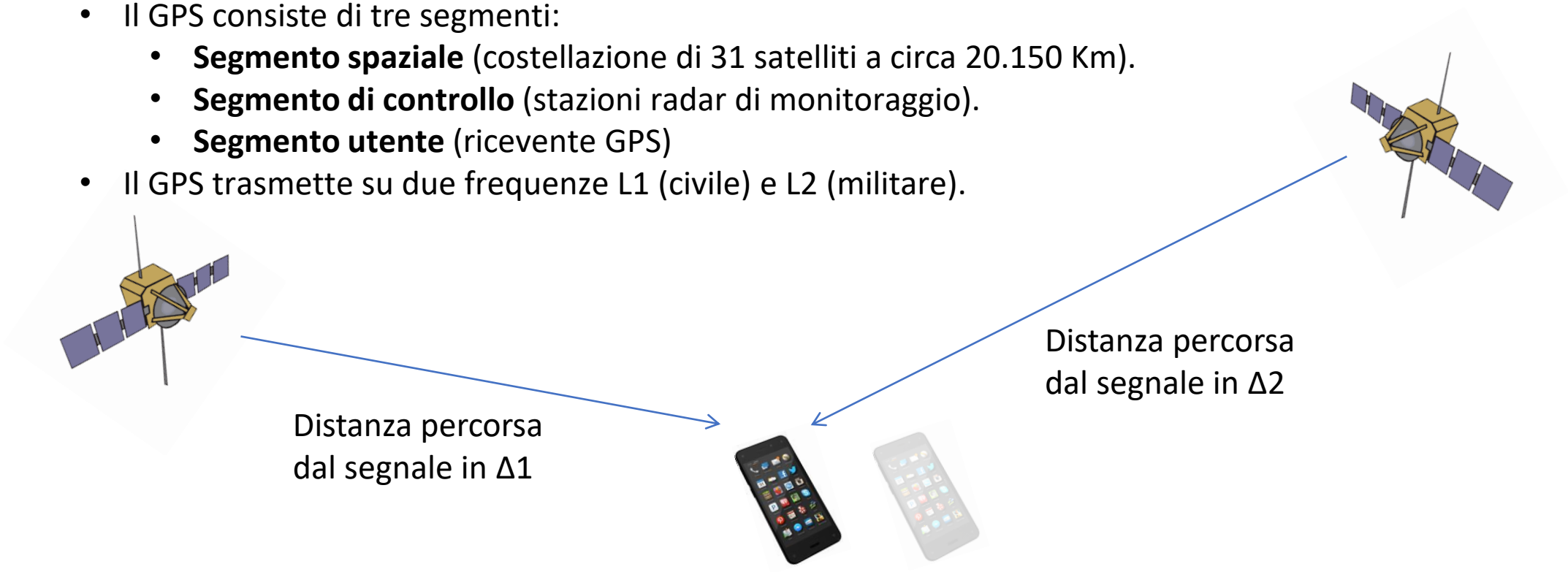
- La **trilaterazione** permette di determinare la posizione relativa di due oggetti sfruttando:
  - La posizione nota di due o più punti di riferimento
  - La distanza misurata tra il punto mobile ciascun punto di riferimento
- La distanza è proporzionale alla potenza del segnale ricevuto
- La distanza può essere ottenuta calcolando il tempo di volo tra l'invio e la ricezione del segnale (*TOA* time of arrival). Richiede la sincronizzazione tra gli orologi.

- La **multilaterazione** permette di ricavare la posizione sulla base della differenza dei tempi di arrivo (**TDOA**) del segnale trasmesso delle diverse stazioni. Non richiede la sincronizzazione degli orologi.

- La **triangolazione** permette di calcolare la posizione sulla base:
  - Angoli di arrivo (**AOA**) tra il punto mobile e i punti di riferimento
  - La distanza tra i punto di riferimento

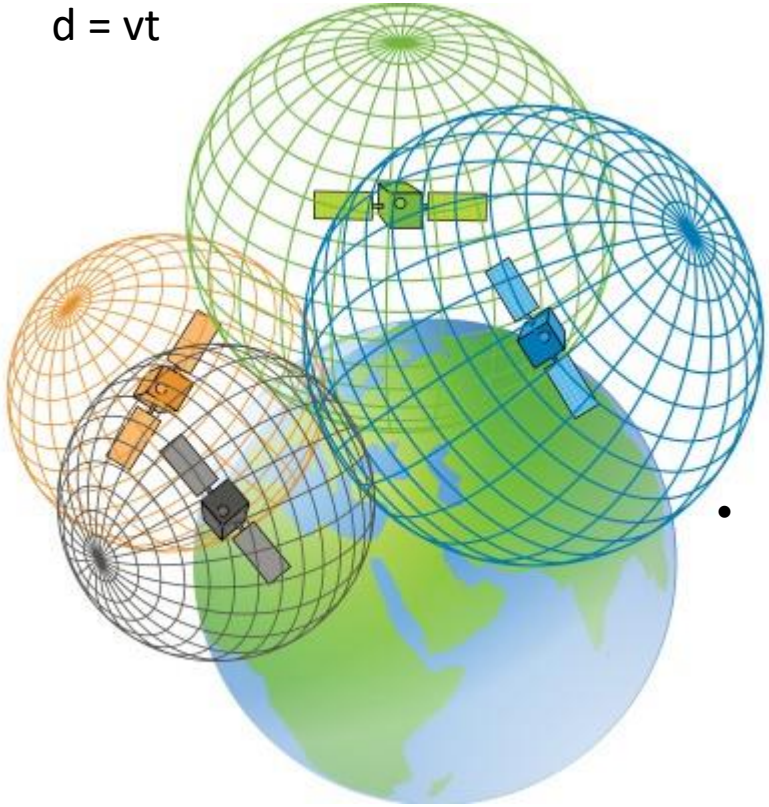
Sistema di posizionamento satellitare (GPS)

- Il GPS consiste di tre segmenti:
  - **Segmento spaziale** (costellazione di 31 satelliti a circa 20.150 Km).
  - **Segmento di controllo** (stazioni radar di monitoraggio).
  - **Segmento utente** (ricevente GPS)
- Il GPS trasmette su due frequenze L1 (civile) e L2 (militare).

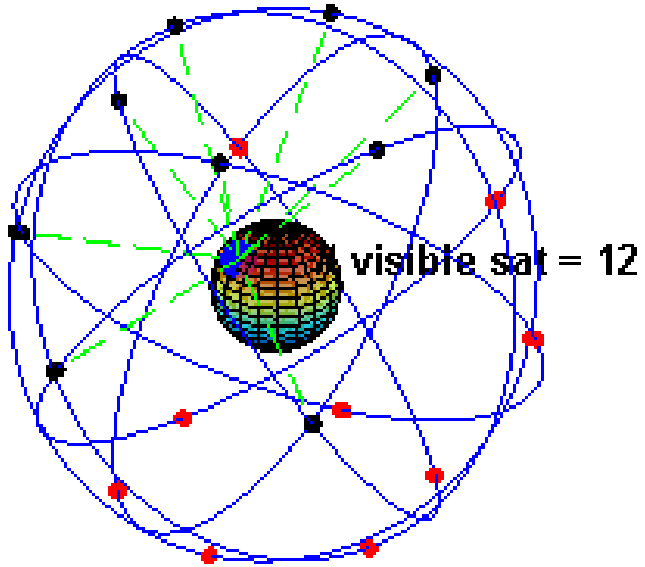


Funzionamento e limiti del GPS

- Per calcolare la distanza sfrutta la prima legge di Newton:
  d = vt



- Si richiede la **LOS libera** per almeno 4 satelliti
  - Non funziona dentro edifici

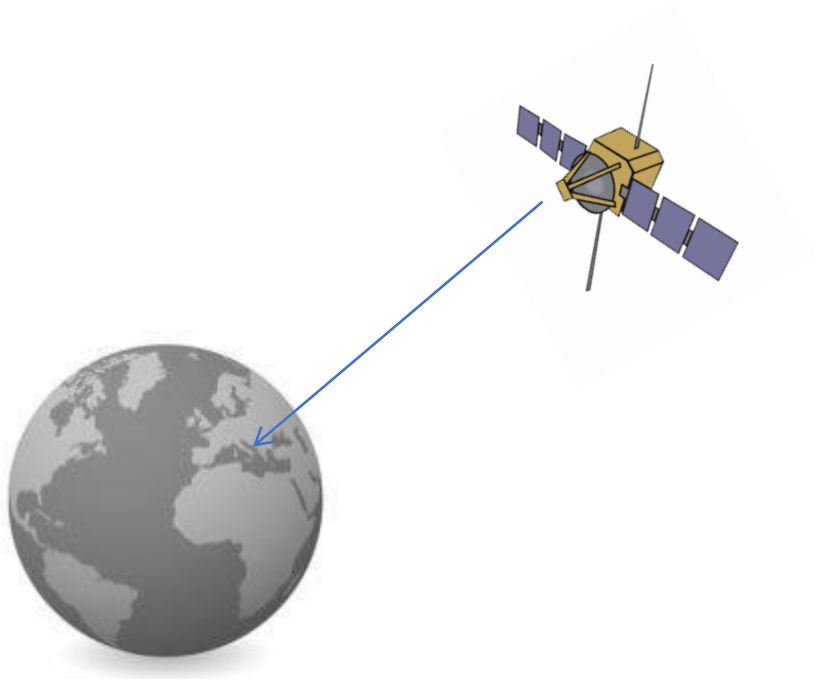


- In ogni punto della terra è sempre possibile "vedere" tra cinque e otto satelliti

Il segnale GPS

- Ogni satellite **GPS trasmette in continuazione due portanti a radiofrequenza**.
  - La portante **L1 (civile)**, a 1575,42 MHz, trasporta il segnale per la localizzazione grossolana ("coarse acquisition") e il segnale di tempo,
  - la portante **L2 (militare)**, a 1227,60 MHz, trasporta il segnale per la localizzazione di precisione.

- Le due portanti sono modulate in fase utilizzando tre diversi codici, quello detto C/A, che serve per la localizzazione grossolana, quello detto P, che serve per la localizzazione precisa e quello che trasporta i dati della navigazione, cioè quei bit che descrivono l'orbita del satellite, le correzioni al suo segnale di clock e altri parametri di sistema.



Tecniche e metodi di misura

- Tre tecniche:
  - Statico (per geodesia)
  - Rapido statico (per controllo locale)
  - Cinematico (utilizzo in movimento)

- Quattro metodi:
  - **Navigazione autonoma** (con singolo ricevitore, precisione <100m con L1)
  - **Posizione con differenza di base** (rilievi topografici, precisione <20mm)
  - **Doppia Frequenza** (usando sia L1 che L2 per ridurre effetti dovuti errore ionosfera)
  - **DGPS** (0.5-5m) sfruttando due o più ricevitori georeferenziati. La stazione calcola le correzioni e le variazioni nel tempo inviandole al rover in movimento

Errori di misura

- Le principali fonti di errore sono:
  - **Ritardo ionosferico** e atmosferico – quando il segnale passa attraverso ionosfera viene rallentato causando un errore nel calcolo della posizione che deriva da variazione velocità del segnale
  - **Errore dell'orologio**
    - del satellite – corretti da segmento di controllo
    - del ricevitore – necessario aggiornare ora frequentemente
  - **Multipath** – causato da riflessioni del segnale del satellite su larghe superfici riflettenti in prossimità del ricevitore. Le onde vengono riflesse dalla superficie e raggiungono l'antenna creando una falsa misura
  - **Dilution of Precision** (**DOP**) – legata alla geometria satelliti e delle loro orbite. In particolare GDOP Misura la degradazione del segnale nella posizione 3D e nel tempo. Aumentare numero di satelliti osservabili con altezza > 15° sull'orizzonte.
  - **Selective availability** (**S/A**) – introdotto dal DOD per degradare la precisione in caso di necessità. Viene introdotto un errore nel segnale e nelle effemeridi. Usare GDPS

Strategie di localizzazione – senza network (Q)

- Problema: Ci si trova in uno spazio aperto senza bussola e mappa. Si ha a disposizione un cellulare ma senza alcun accesso a network (GPS o Internet). Come si può determinare la nostra posizione?

Strategie di localizzazione – senza network (A)

- Problema: Ci si trova in uno spazio aperto senza bussola e mappa. Si ha a disposizione un cellulare ma senza alcun accesso a network (GPS o Internet). Come si può determinare la nostra posizione?

→ mediante **proximity sensing** . si possono ricavare i Cell-ID dei ricevitori vicini

→ CONTRO: senza internet ai avrebbe la necessità di avere già a disposizione un db con la posizione dei ripetitori e sfruttare quelle all'interno dell'applicazione

Quale strategia di localizzazione adottare per applicazioni mobile?

Tipo	PRO	CONTRO
DEAD RECKONING	Non richiede grossa potenza di calcolo	Richiede conoscenza esatta dello stato e del pregresso
PROXIMITY SENSING	Richiede infrastruttura on-ground	Richiede conoscenza geolocalizzazione di tutte le stazioni
TOA	Precisione	Richiede sincronizzazione degli orologi
TDOA	Non richiede sincronizzazione orologi rispetto TOA	Aumento complessità
AOA	--	Si necessità delle misure angolari
GPS	Ricevitore GPS già integrato	• Funziona solo all'esterno • Richiede LOS libera • Consumo elevato batteria • Output lento

Strategie di localizzazione – con internet (Q)

- Problema: Ci si trova in uno spazio aperto senza bussola e mappa. Si ha a disposizione un cellulare ma senza alcun accesso ai GPS. Come si può determinare la nostra posizione?

Strategie di localizzazione – senza network (A)

- Problema: Ci si trova in uno spazio aperto senza bussola e cartina. Si ha a disposizione un cellulare ma senza alcun accesso ai GPS. Come si può determinare la nostra posizione?

→ Ottenere la posizione dell'utente in Android funziona tramite callback. Indichi che desideri ricevere aggiornamenti sulla posizione dal LocationManager chiamando **requestLocationUpdates()**, passandogli un LocationListener. **LocationListener** deve implementare diversi metodi di callback che il LocationManager chiama quando cambia la posizione dell'utente o quando cambia lo stato del servizio.

Esempio di LocationListener

```
// Acquire a reference to the system Location Manager
LocationManager locationManager = (LocationManager) this.getSystemService(Context.LOCATION_SERVICE);

// Define a listener that responds to location updates
LocationListener locationListener = new LocationListener() {
    public void onLocationChanged(Location location) {
      // Called when a new location is found by the network location provider.
      makeUseOfNewLocation(location);
    }

    public void onStatusChanged(String provider, int status, Bundle extras) {}

    public void onProviderEnabled(String provider) {}

    public void onProviderDisabled(String provider) {}
  };

// Register the listener with the Location Manager to receive location updates
locationManager.requestLocationUpdates(LocationManager.NETWORK_PROVIDER, 0, 0, locationListener);
```

Esempio di LocationListener

```
// Acquire a reference to the system Location Manager
LocationManager locationManager = (LocationManager) this.getSystemService(Context.LOCATION_SERVICE);

// Define a listener that responds to location updates
LocationListener locationListener = new LocationListener() {
    public void onLocationChanged(Location location) {
      // Called when a new location is found by the network location provider.
      makeUseOfNewLocation(location);
    }

    public void onStatusChanged(String provider, int status, Bundle extras) {}

    public void onProviderEnabled(String provider) {}

    public void onProviderDisabled(String provider) {}
  };

// Register the listener with the Location Manager to receive location updates
locationManager.requestLocationUpdates(LocationManager.NETWORK_PROVIDER, 0, 0, locationListener);
```

- Indica quale location provider usare:
  - NETWORK_PROVIDER per cell tower e wifi
  - GPS_PROVIDER per GPS
- NOTA: si possono anche mettere due requestLocationUpdates:
  uno per network e uno per gps

Problematiche da tenere in conto

- **<users-permission>**
  Dichiarare nel **manifest** file che si intende richiedere accesso a **NETWORK_PROVIDER** o **GPS_PROVIDER**
  va richiesto il permesso per **android.permission.ACCESS_FINE_LOCATION** (per network + gps) o
  android.permission.ACCESS_COARSE_LOCATION (solo network)

- **<users-feature> (> Android 5.0 (API level 21))**
  Dichiarare nel manifest file che si intende usare
  android.hardware.location.network o **android.hardware.location.gps**
  per potere usare **NETWORK_PROVIDER** o **GPS_PROVIDER**

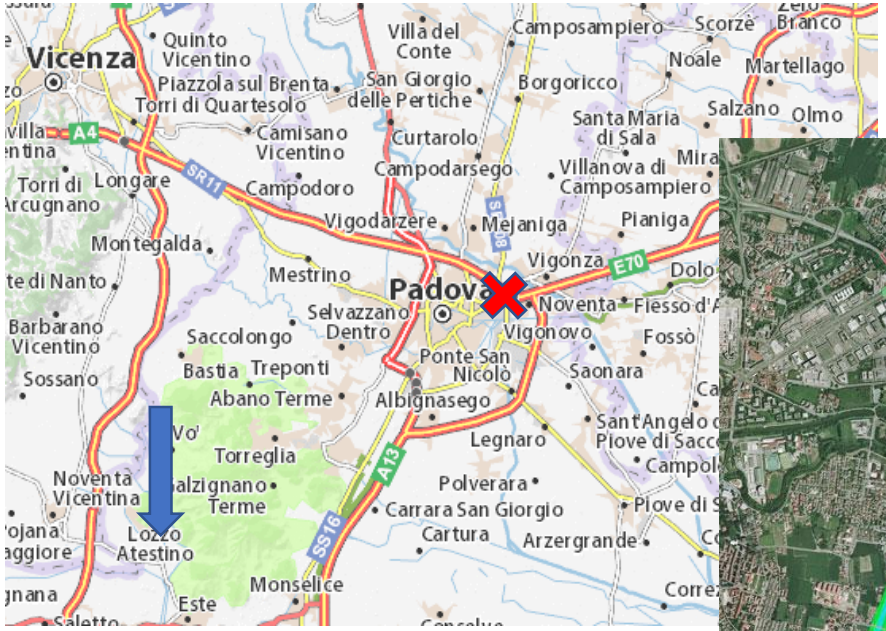
- **Background Location Limit (> Android 8.0 (API level 26))**
  Se in background le applicazioni possono ricevere aggiornamento sulla localizzazione solo poche volte ogni ora

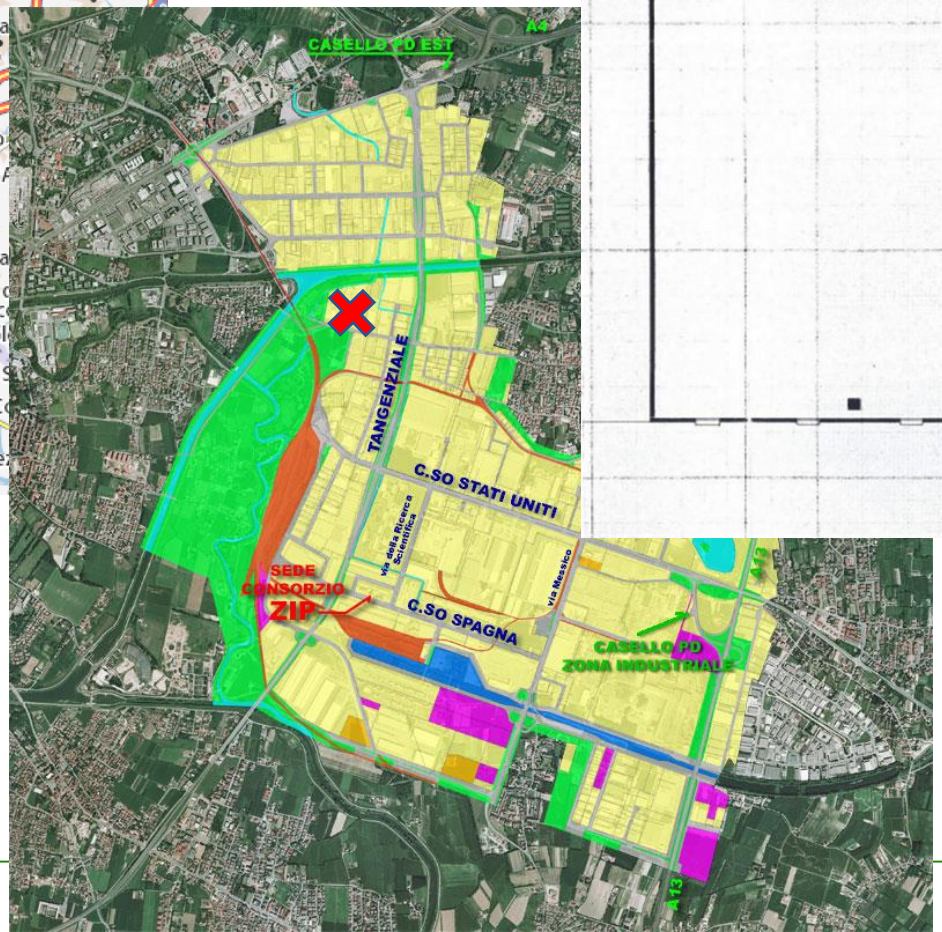
- **Foreground Limit (> Android 7.1.1 (API level 25))**
  Nessuna limitazione, ma riduzione durata batteria

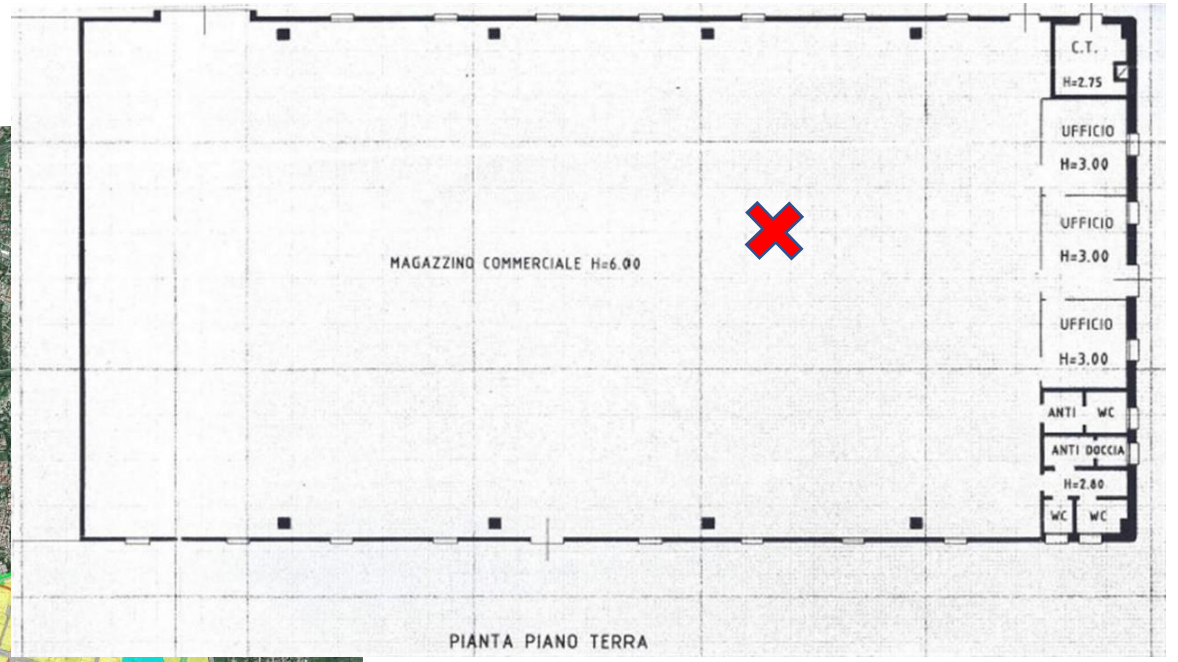
Strategie di localizzazione – batteria limitata (Q)

- Problema: Ci si trova in uno spazio aperto e si deve raggiungere un determinato punto in uno spazio chiuso predefinito. Si ha a disposizione un cellulare con accesso GPS e internet ma la batteria è limitata. Come procedere?

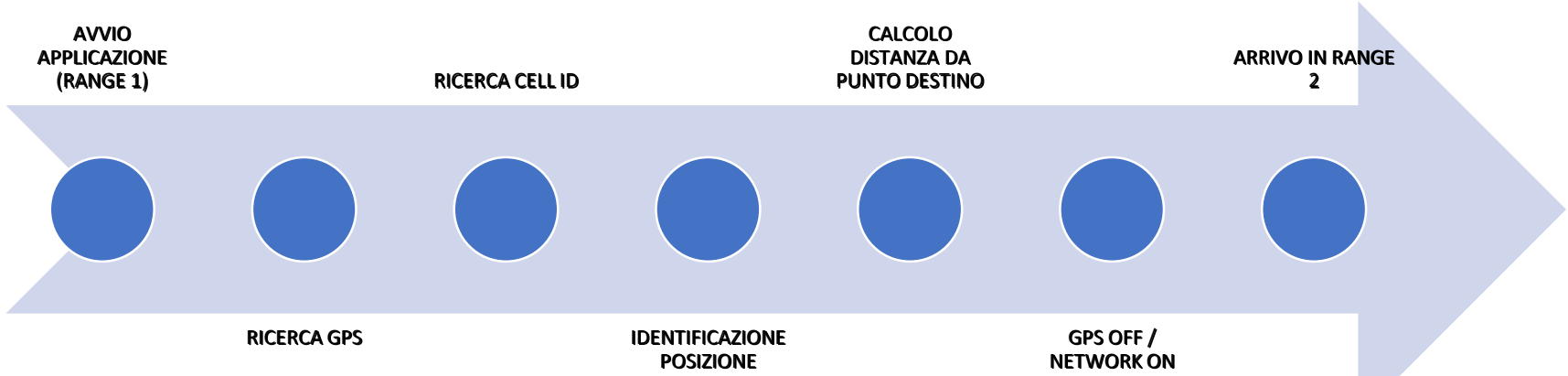
Strategie di localizzazione – batteria limitata (Q)







Strategie di localizzazione – batteria limitata (A)



Strategie di localizzazione – batteria limitata (A)

PARTENZA DA RANGE 2

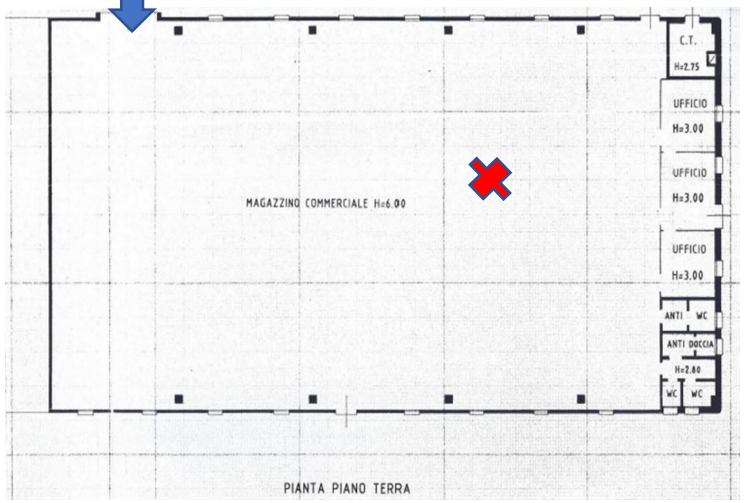
GPS ON / NETWORK OFF

RICERCA GPS

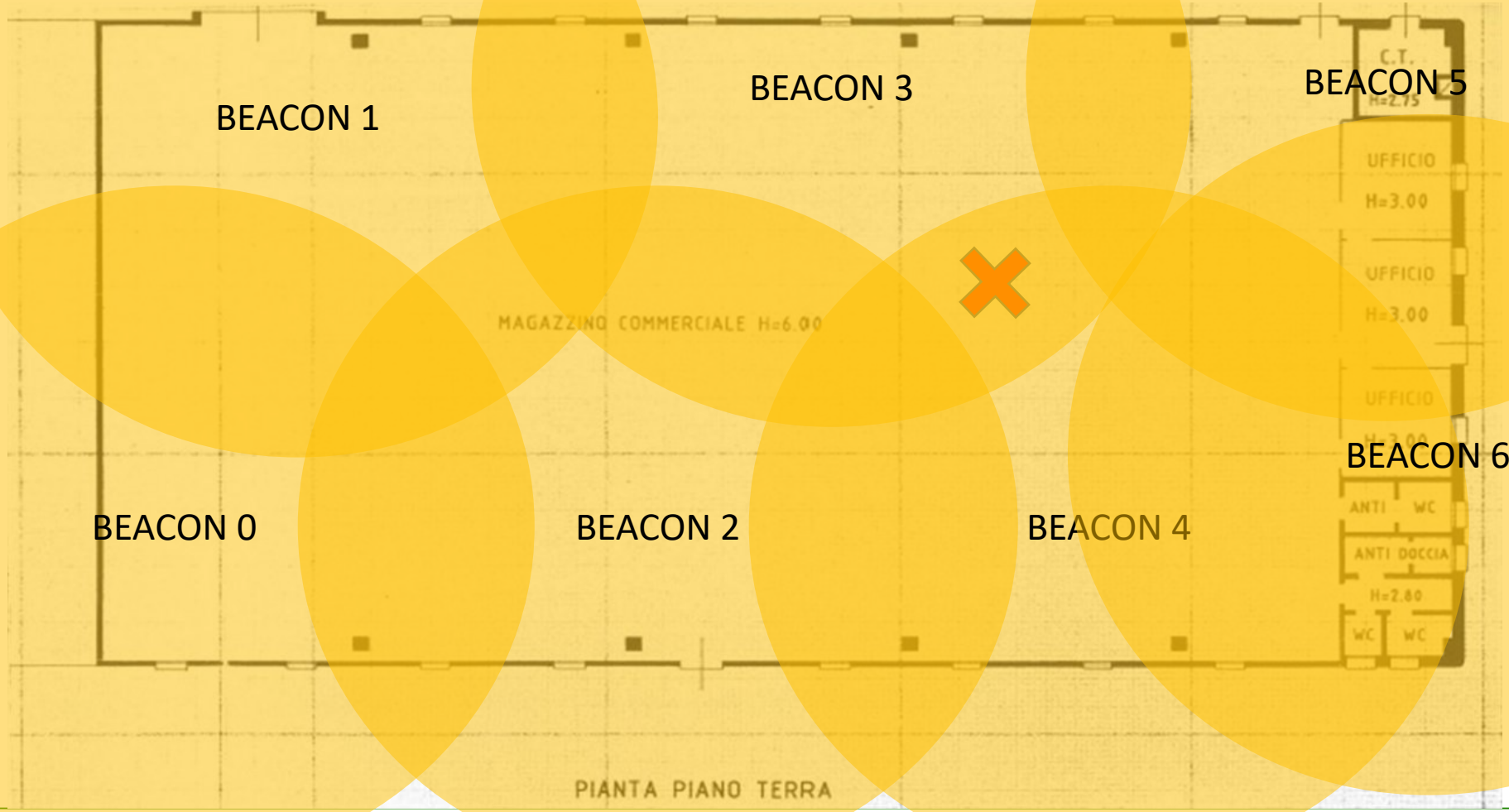
IDENTIFICAZIONE POSIZIONE

CALCOLO DISTANZA DA PUNTO DESTINO

ARRIVO IN RANGE 3

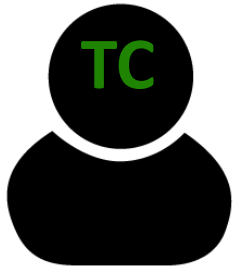


Strategie di localizzazione – batteria limitata (A)



Grazie per l’attenzione

Davide Zanetti
dzanetti@imolainformatica.it



Tommaso Cardona
tcardona@imolainformatica.it

www.imolainformatica.it
blog.imolainformatica.it
www.mokabyte.it