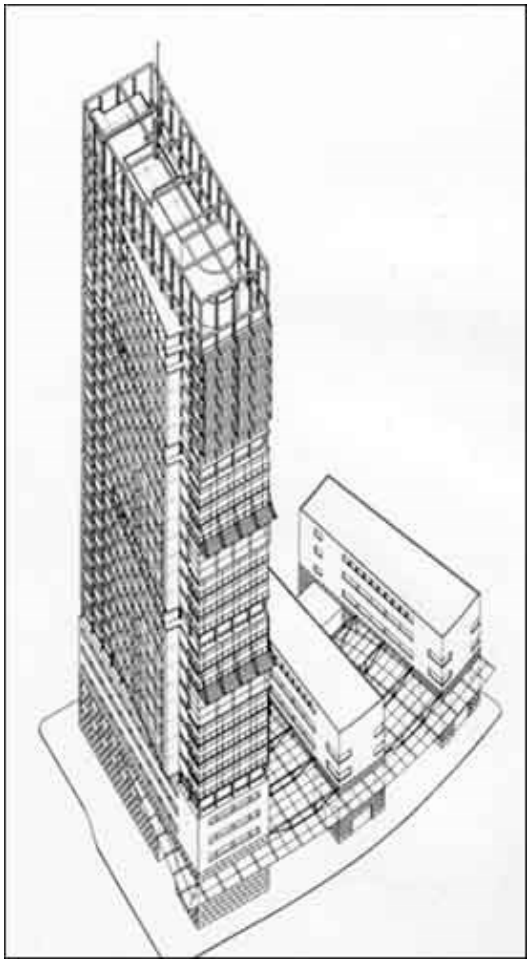
Evoluzione del ruolo dell'informatico nell'ambito dello sviluppo del software: una prospettiva storica

Sistemi software - 1980-1990

- **Complessità: bassa**
  - Automazione di compiti ripetitivi (contabilità)
  - Produttività individuale (spreadsheet, word processor)
- **Distribuzione: bassa**
  - Diffusione dei personal computer
  - Applicazioni monoutente su PC o multiutente su mainframe/mini
- **Approccio funzionale**
  - Applicazione = insieme di semplici funzioni utili per i compiti svolti da un singolo utente
  - Struttura monolitica

Sistemi software – 1990-2000

- **Complessità: media**
  - Sistemi gestionali che incidono sull'organizzazione
  - Prime funzioni di collaborazione
- **Distribuzione: media**
  - Diffusione delle reti locali
  - Applicazioni multiutente su LAN
- **Approccio macrofunzionale**
  - Applicazione = insieme di funzioni complesse utili per i compiti svolti da un singolo utente o da gruppi di utenti omogenei
  - Prime forme di modularizzazione



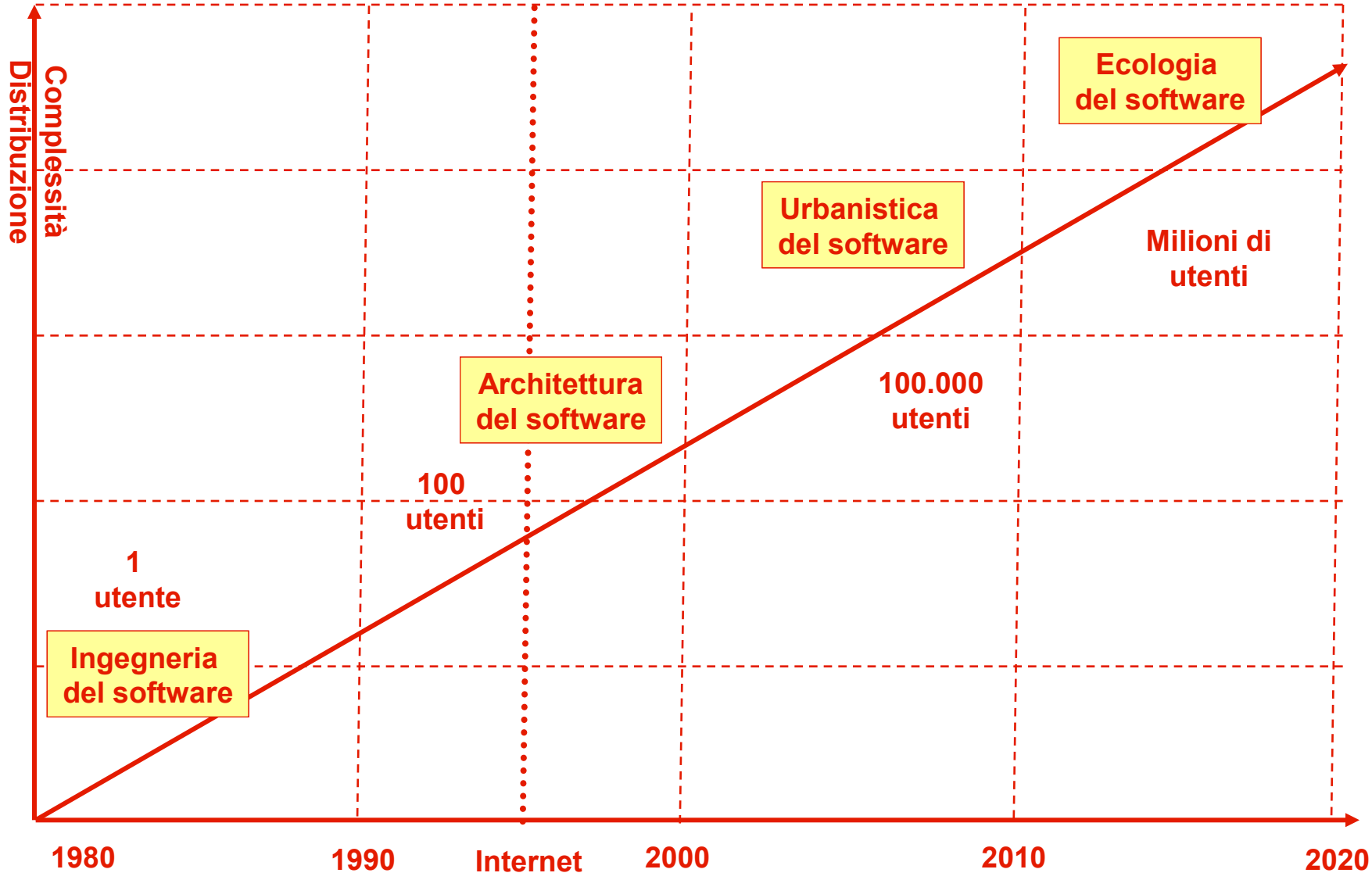
Sistemi software – 2000-2010

- **Complessità: elevata**
  - Sistemi che guidano i cambiamenti organizzativi
  - Enfasi sulle funzioni di collaborazione
- **Distribuzione: elevata**
  - Diffusione di internet
  - Applicazioni Web aperte agli attori esterni
- **Approccio per processi**
  - Applicazione = insieme di processi.
  - Coordinamento di molti compiti semplici affidati a molti attori diversi
  - Architetture fortemente modulari (SOA)

Sistemi software – 2010-2020

- **Complessità: molto elevata**
  - Sistemi che definiscono in modo pervasivo l'organizzazione (Digital Trasformation)
- **Distribuzione: molto elevata**
  - Prevalenza dei dispositivi mobili (mobile first)
  - Internet of things
- **Approccio per ecosistemi**
  - Applicazioni come aggregatori di servizi variamente componibili
  - Architetture fortemente modulari (microservizi)
  - Automazione dei processi di sviluppo (Devops)

Evoluzione dei sistemi software



Urbanistica ed ecologia vs. Ingegneria e architettura

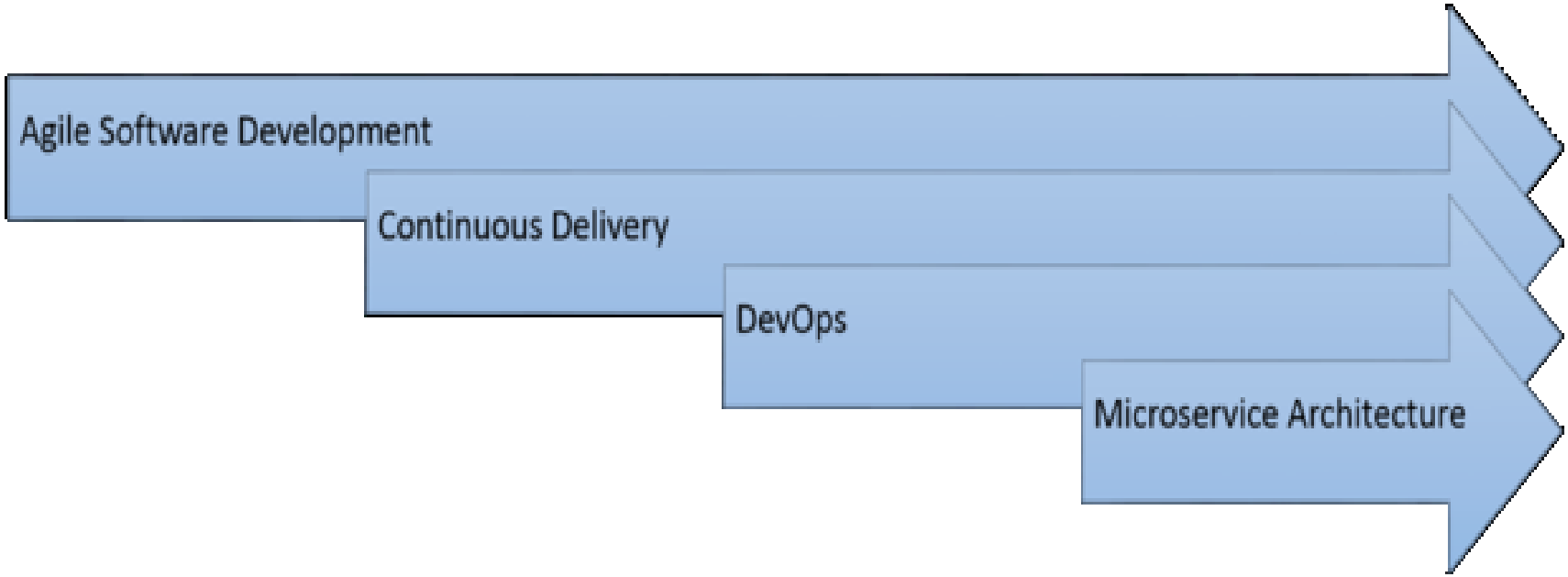
- **L'urbanistica e l'ecologia non riguardano solo le costruzioni**
- Devono tener conto di
  - Vivibilità
  - Sostenibilità
  - Impatto sociale
- Devono preoccuparsi di gestire
  - Le infrastrutture
  - Le reti di relazioni
  - L'organizzazione e i suoi cambiamenti
- **Richiedono una gestione della complessità**

Gestire la complessità

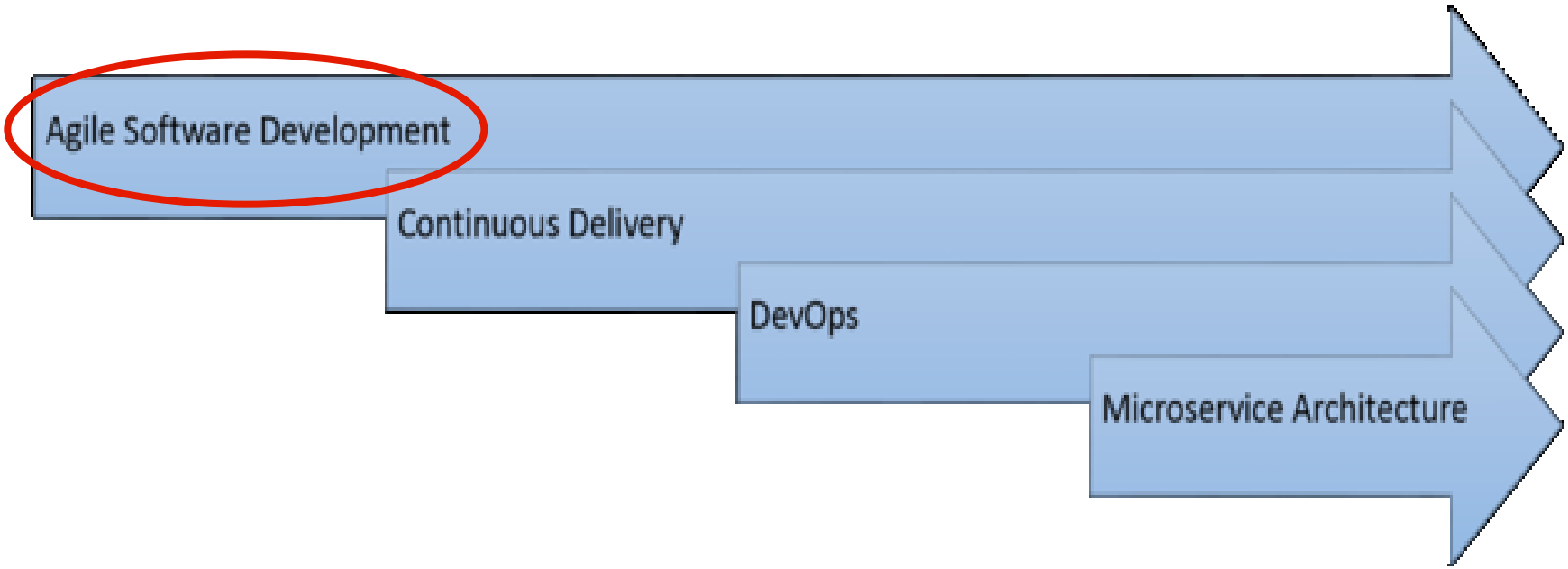
- **Bisogna quindi**
  - Gestire sistemi molto complessi
  - Con una forte modularità
  - Su cui lavorano molte persone
  - In un contesto in continuo cambiamento
- **Per riuscirci occorre un approccio rigoroso:**
  - Metodi di sviluppo orientati al cambiamento
  - Strumenti per gestire in modo efficiente (e automatizzato) lo sviluppo e il rilascio del software
  - Integrazione fra sviluppo e gestione dei sistemi
  - Infrastrutture modulari

The agile progression

- L'insieme di questi metodi e strumenti prende abitualmente il nome di **agile progression**
- Comprende quattro filoni principali:



The agile progression – Agile Software Development



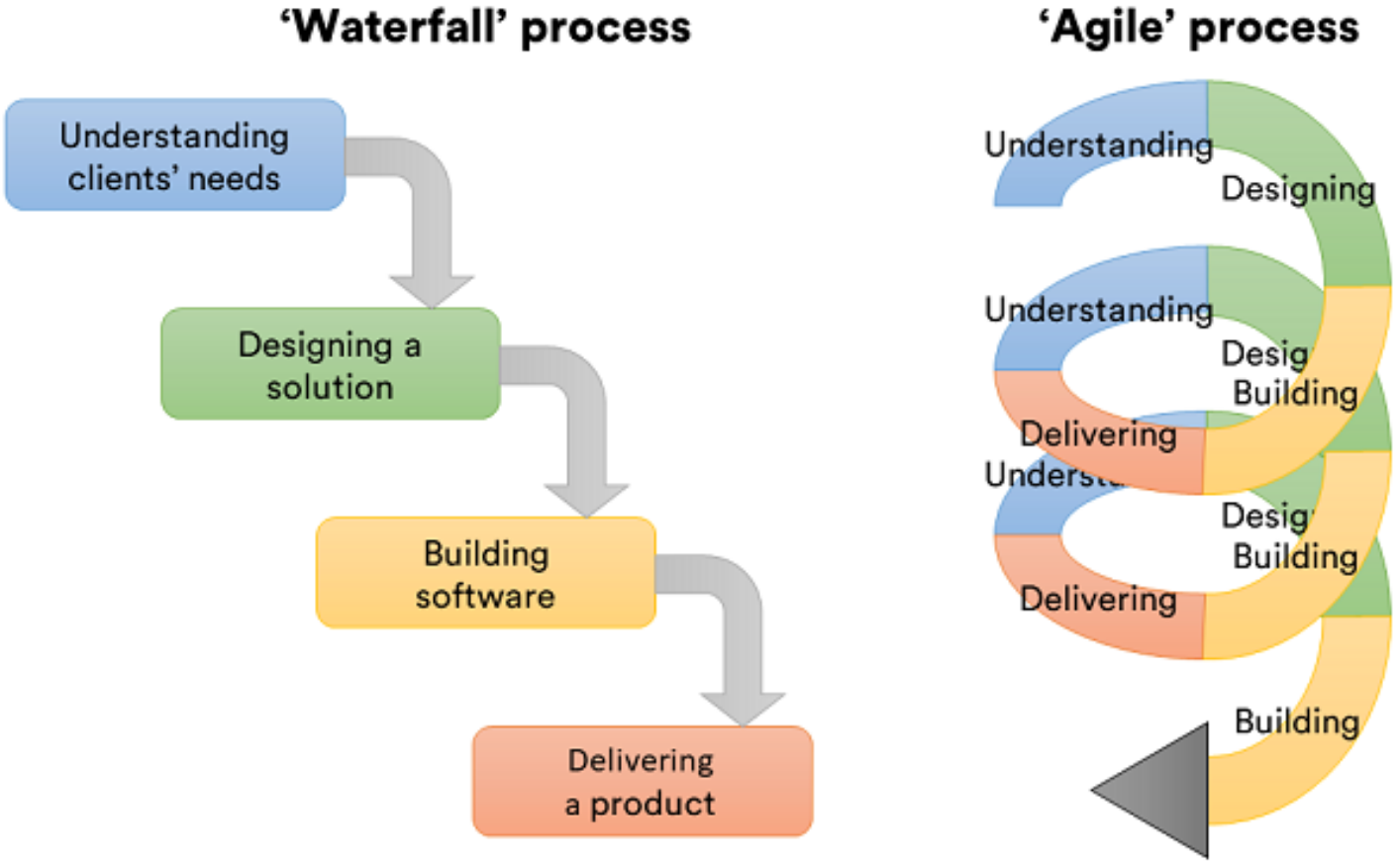
Agile Software Development (ASD)

- Insieme di metodi di sviluppo del software emersi a partire dai primi anni 2000 e fondati su un insieme di principi comuni (XP, SCRUM…)
- Derivano dai principi del **Manifesto for Agile Software Development** (2001)
- Si contrappongono al modello a cascata e ad altri modelli di sviluppo tradizionali.
- Sono focalizzati sull'obiettivo di consegnare al cliente software funzionante e di qualità.
- Sono focalizzati sulla **gestione del cambiamento** piuttosto che sulla pianificazione
- Propongono un approccio di tipo **iterativo** e **incrementale**

I principi del manifesto

- Stiamo scoprendo modi migliori di creare software, sviluppandolo e aiutando gli altri a fare lo stesso.
- Grazie a questa attività siamo arrivati a considerare importanti:
  - **Gli individui e le interazioni** più che i processi e gli strumenti
  - **Il software funzionante** più che la documentazione esaustiva
  - **La collaborazione col cliente** più che la negoziazione dei contratti
  - **Rispondere al cambiamento** più che seguire un piano
- Ovvero, fermo restando il valore delle voci a destra, consideriamo più importanti le voci a sinistra.

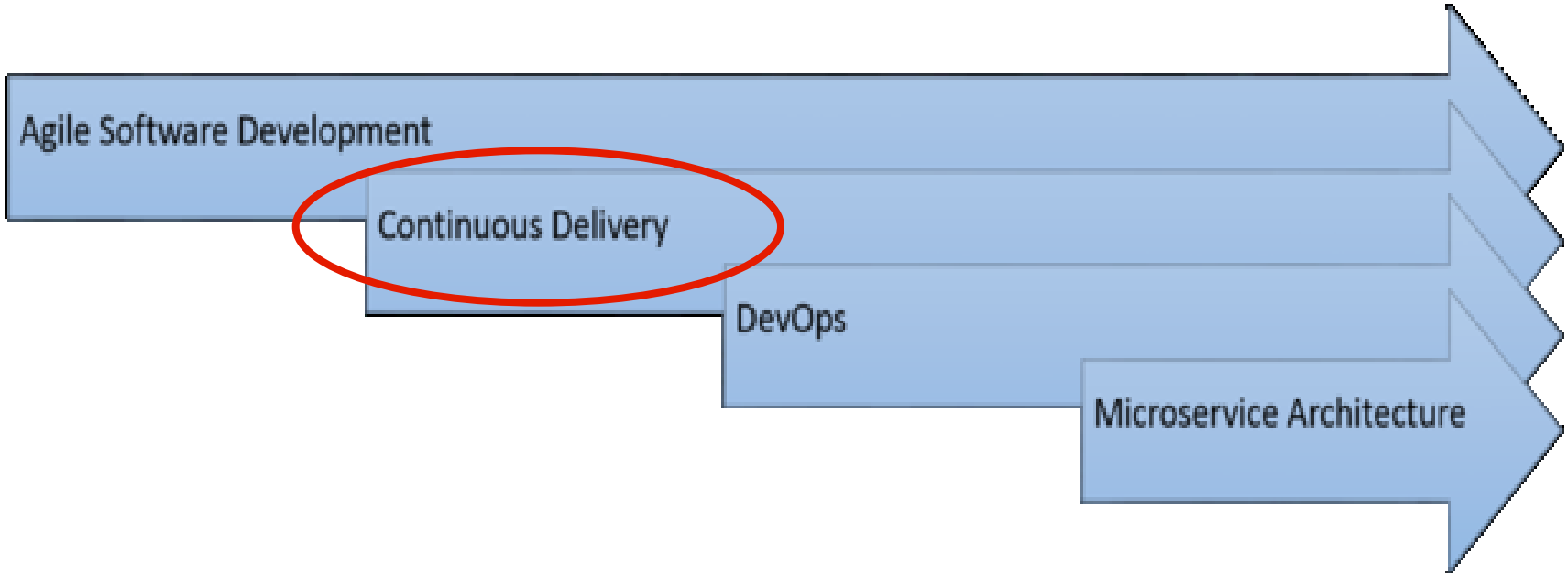
Agile vs Waterfall



Agile e organizzazione del lavoro

- I modelli di sviluppo agili non hanno quindi solo una componente tecnologica: **implicano cambiamenti culturali!**
- In termini organizzativi adottare una metodologia agile significa (tra l'altro):
  - **Professionalizzazione delle persone** (modello dell'orchestra)
  - **Piccoli gruppi di sviluppo** (ma capacità di coordinarsi con gli altri gruppi)
  - **Lavorare a fianco a fianco con i clienti** condividendo le scelte (capacità di relazione e flessibilità)
  - **Automazione delle funzioni ripetitive** (ad esempio i test o il rilascio)

The agile progression – Continuous Delivery



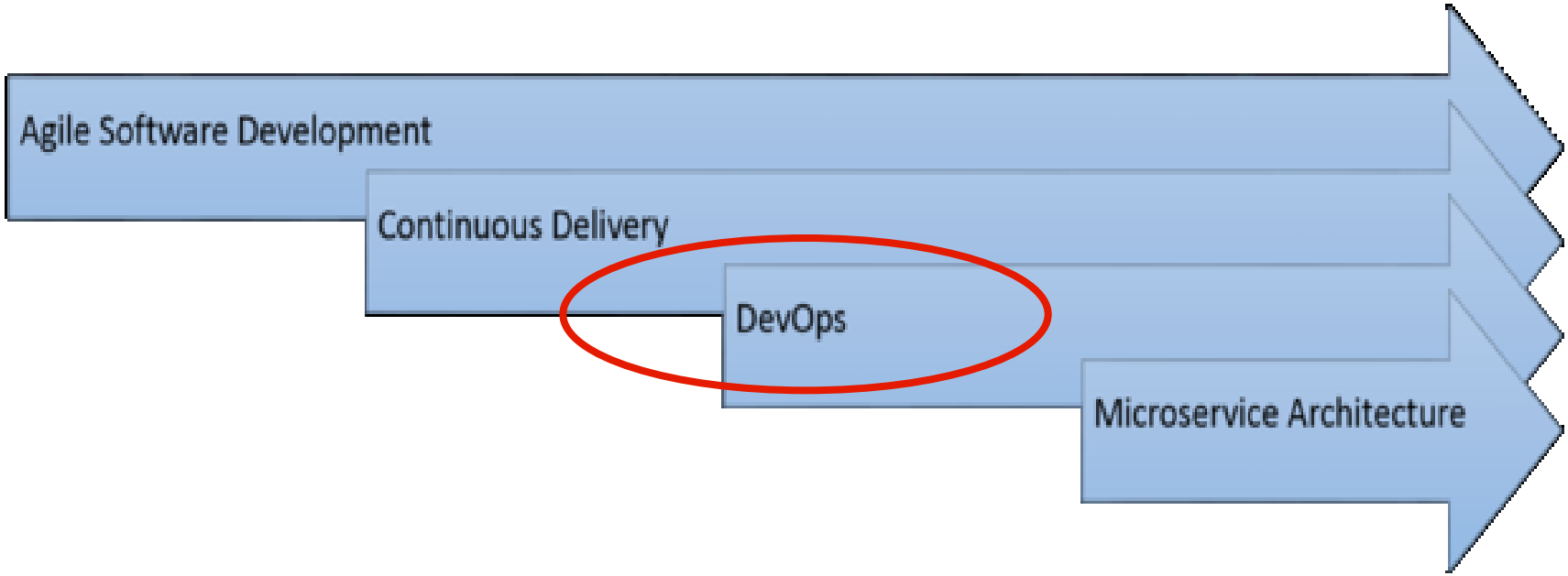
Automazione

- Le metodologie agili prevedono rilasci molto frequenti
- I sistemi modulari sono costituiti da un numero molto elevato di componenti e di interazioni fra di essi
- Ad ogni rilascio bisogna verificare che:
  - Il singolo componente si comporti correttamente
  - Le interazioni fra i componenti rimangano integre
- I test si realizzano assieme al software (**Test Driven Development**) e sono automatizzati
- Si usano **sistemi di versionamento** per far sì che ogni persona del gruppo possa lavorare indipendentemente su parti diverse del prodotto
- I contributi dei vari sviluppatori vengono integrati con una frequenza molto elevata (almeno 1 volta al giorno)

Continuous integration e continuous delivery

- Per garantire un flusso di lavoro veramente «agile» è necessario rendere il più possibile automatico:
  - Il processo di consegna dei moduli (**continuous integration**):
    - Build
    - Test di unità
    - Test di integrazione
  - Il processo di rilascio delle nuove versioni delle applicazioni (**continuous delivery**)

The agile progression - DevOps



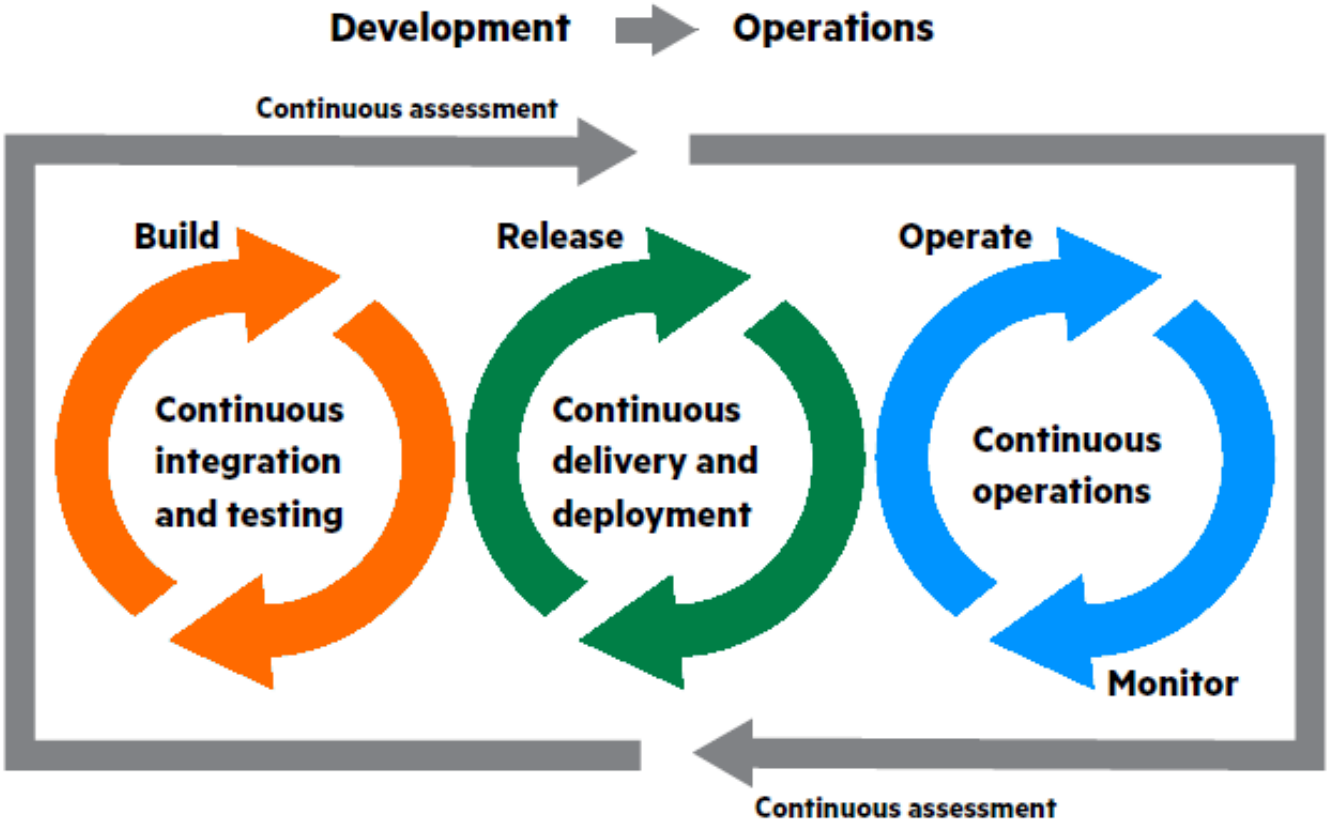
DevOps

- In una realtà strutturata 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** (IaaC)
  - **Metodologie di lavoro** uniformi e condivise fra sviluppatori e sistemisti (di tipo agile).

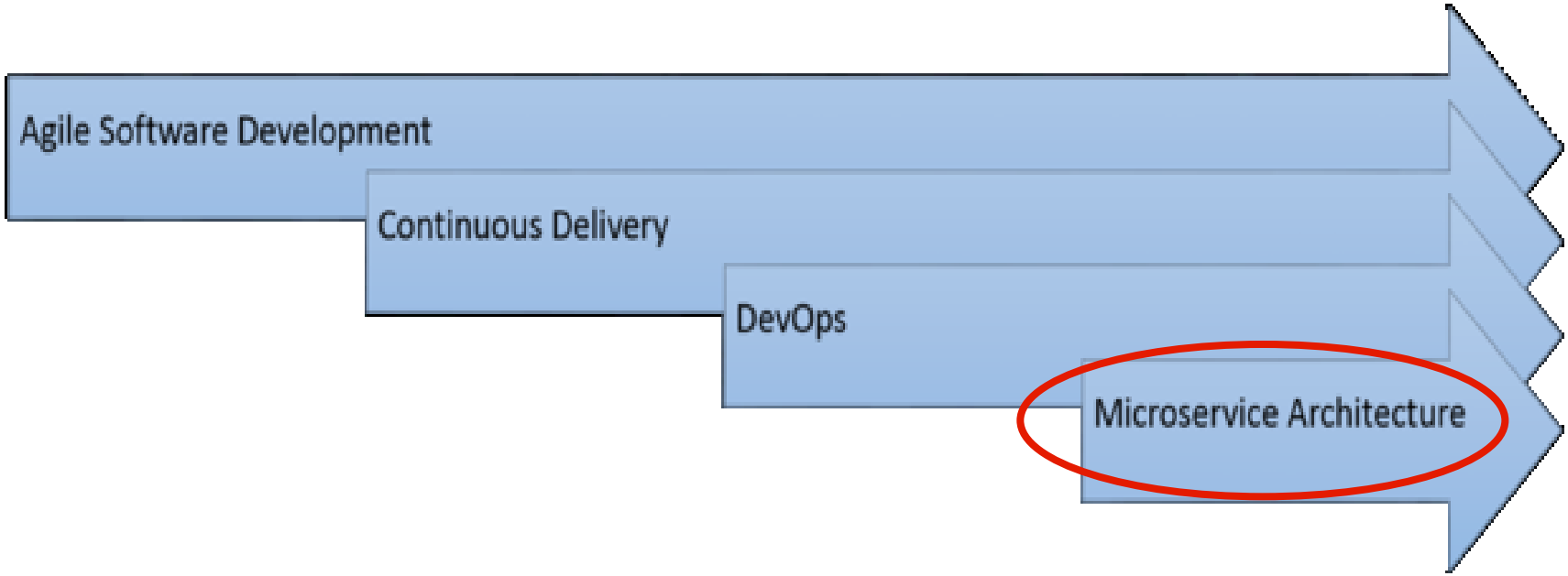
Devops - CALMS

- I principi **DevOps** sono sintetizzati nell'acronimo **C.A.L.M.S.**
  - **Culture:** gestire il cambiamento focalizzandosi sulla collaborazione
  - **Automation:** eliminare le azioni manuali ripetitive (Continuous Integration, Continuous Delivery, Infrastructure-as-a-code)
  - **Lean:** utilizzare i principi Lean per velocizzare, standardizzare e rendere efficienti le attività
  - **Metrics:** misurare qualsiasi cosa, utilizzando i risultati per rifinire costantemente le attività
  - **Sharing:** condividere le esperienze di successo e di fallimento per una crescita diffusa

DevOps e automazione



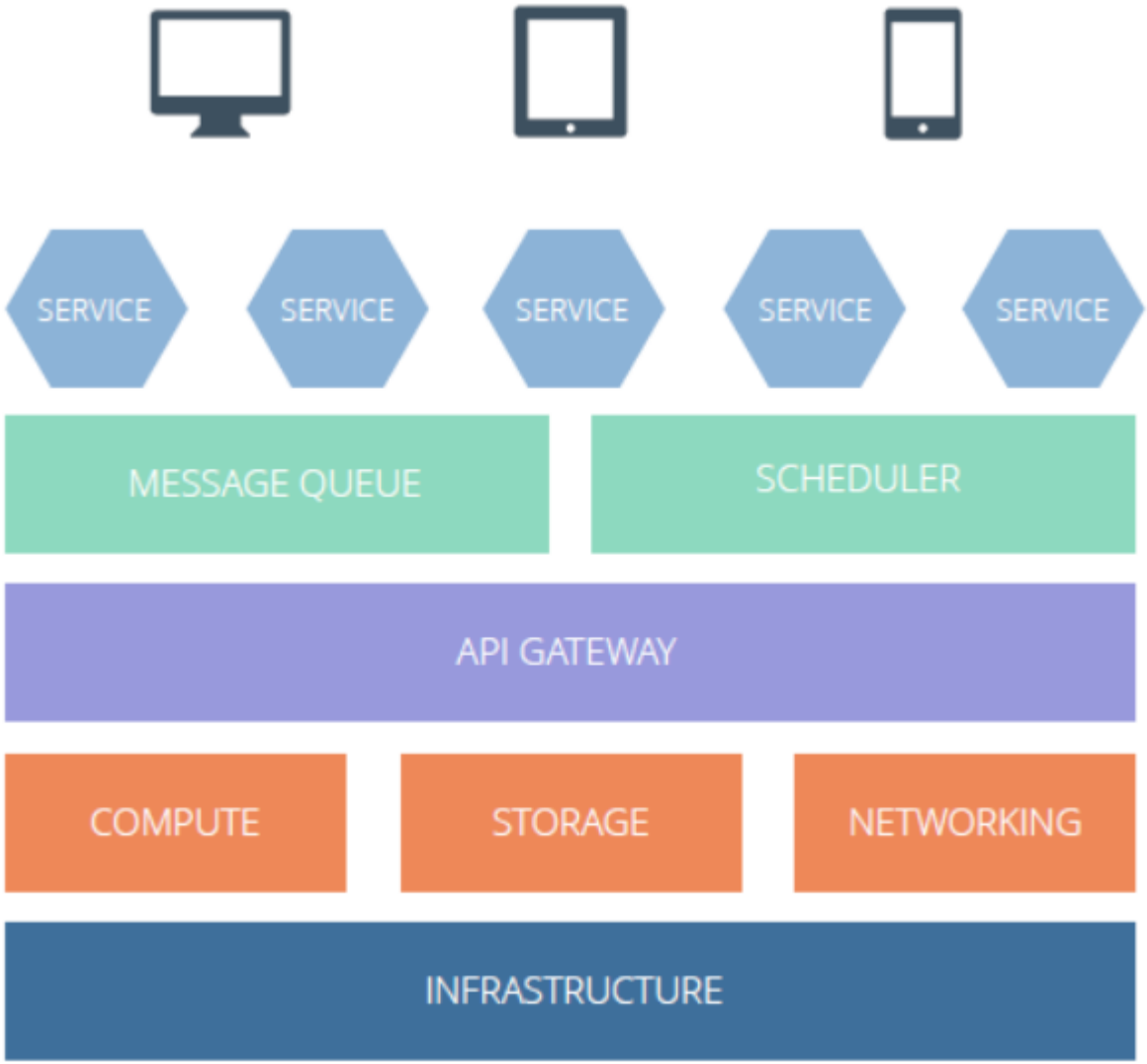
The agile progression – Microservice Architecture



Modularizzazione e microservizi

- Come si è detto la modularizzazione è un aspetto essenziale dell'odierno 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
- Non è un idea totalmente nuova, è in qualche modo una 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.

Schema di architettura a microservizi



The agile progression