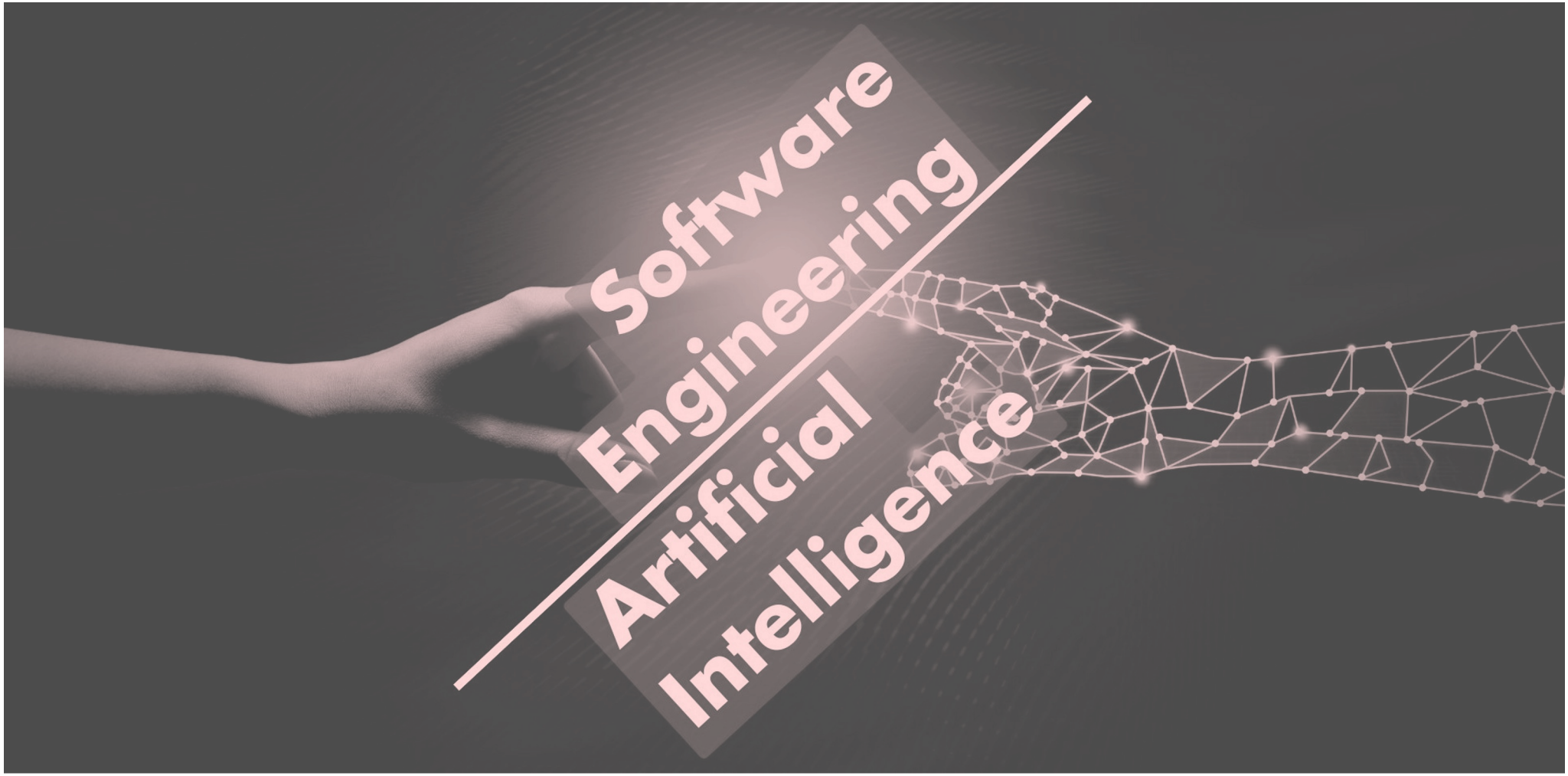


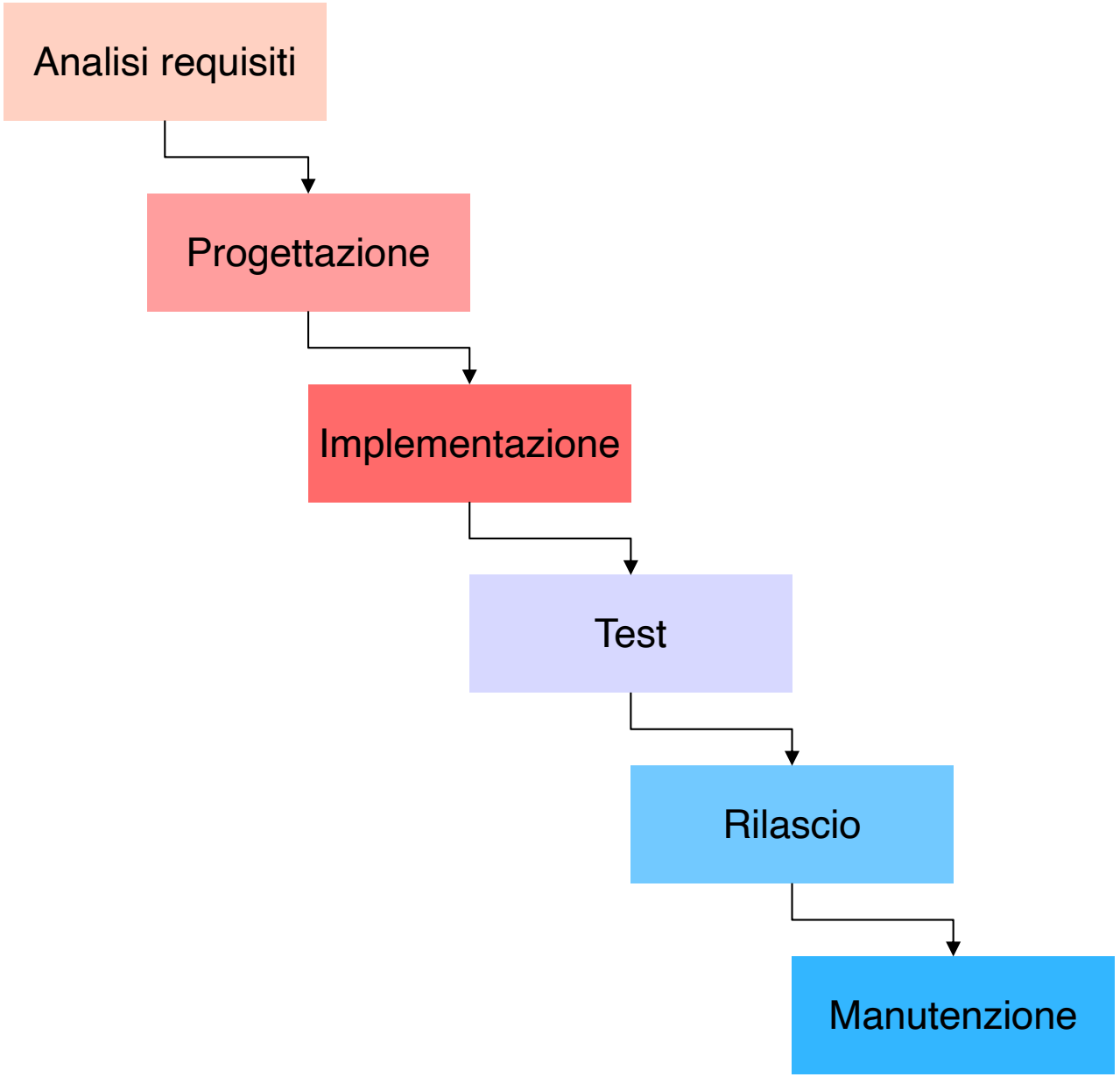
Opportunità e rischi della IA nel software

Luca Mainetti

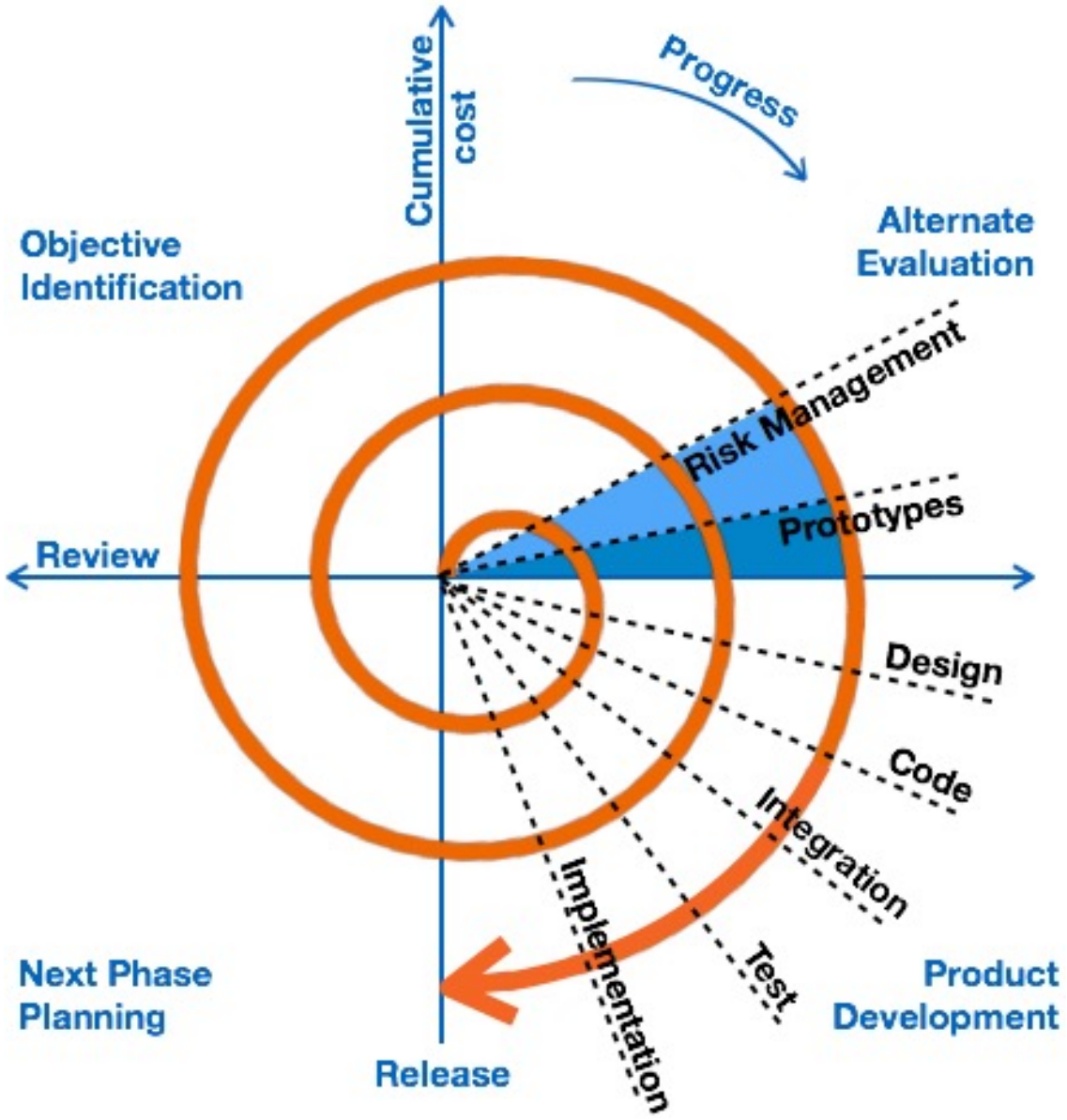
UNIVERSITÀ DEL SALENTO

PROLOGO

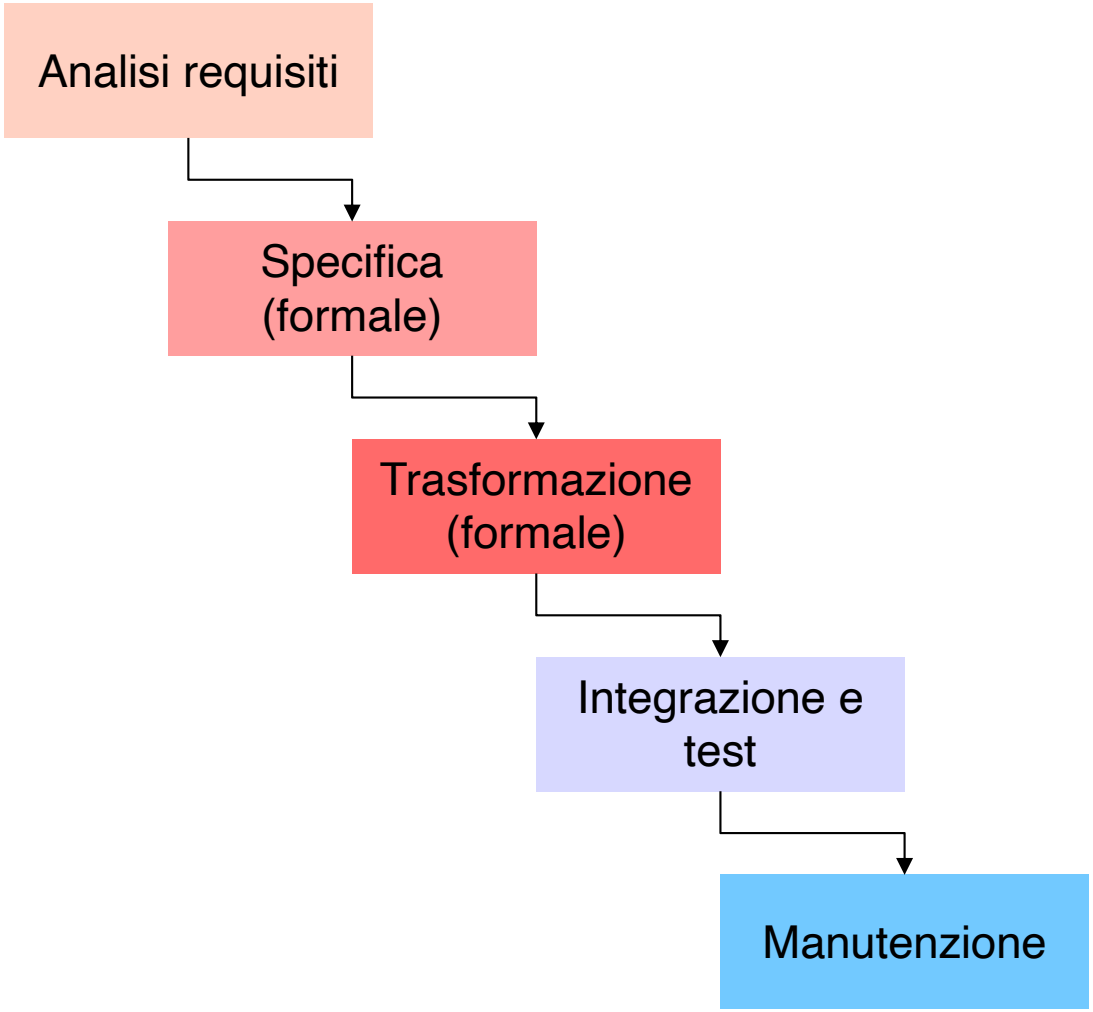
Complessità



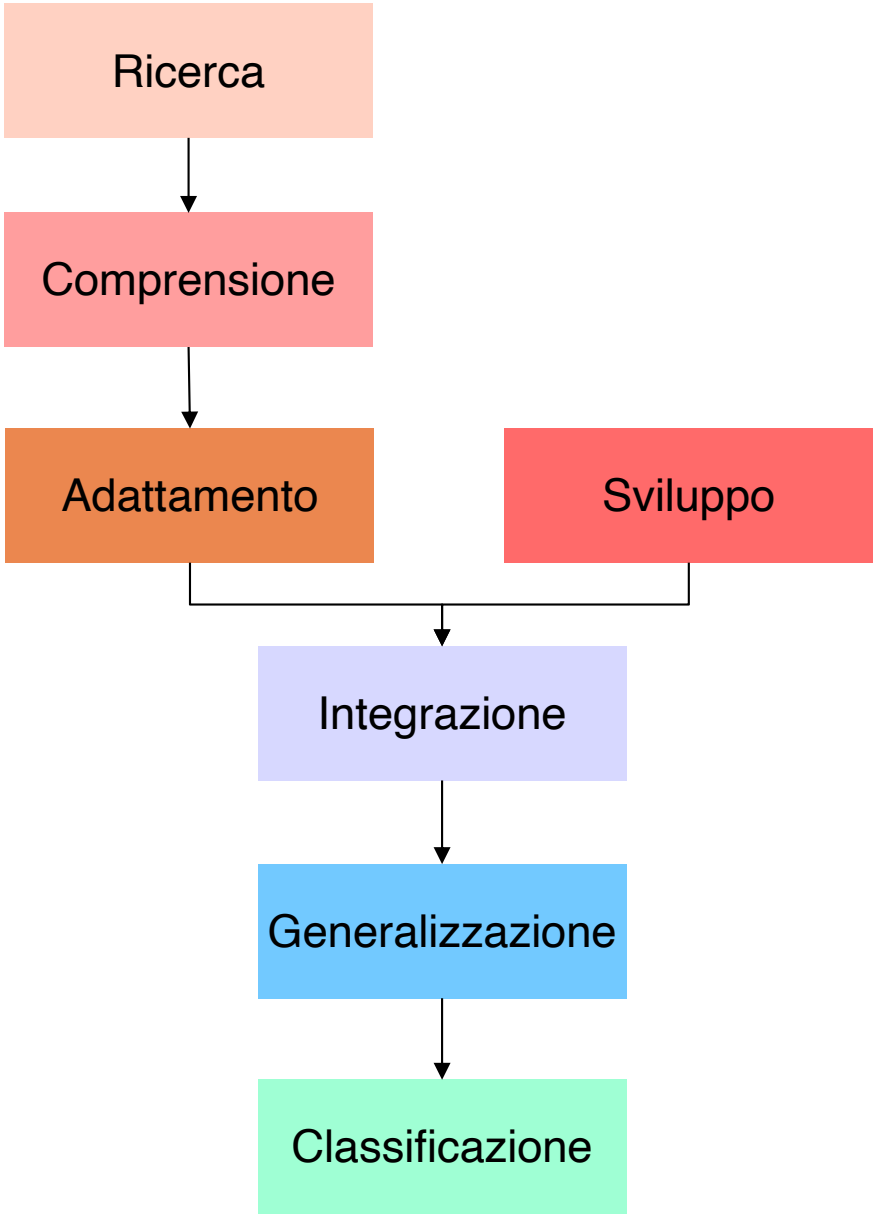
Cambiamento



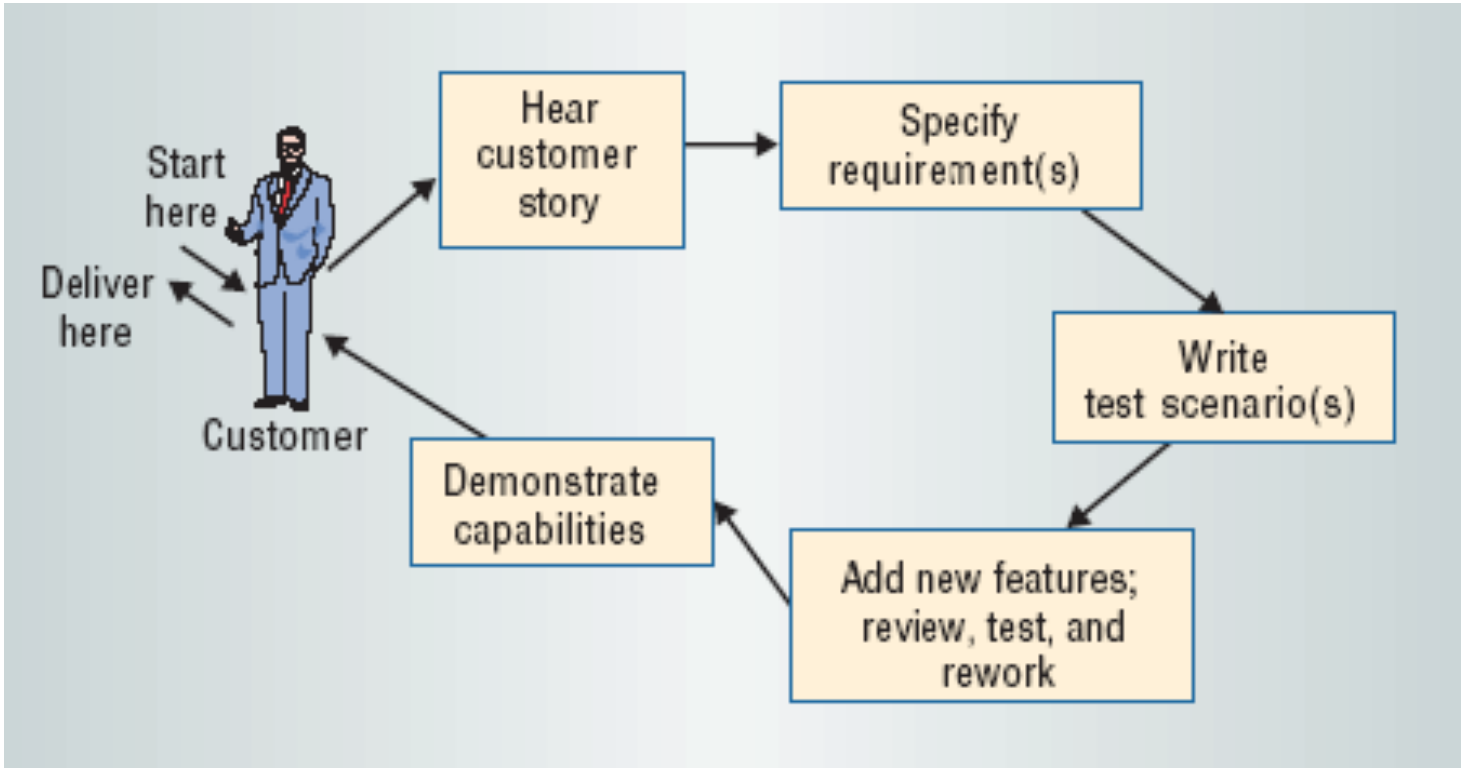
Trasformazione



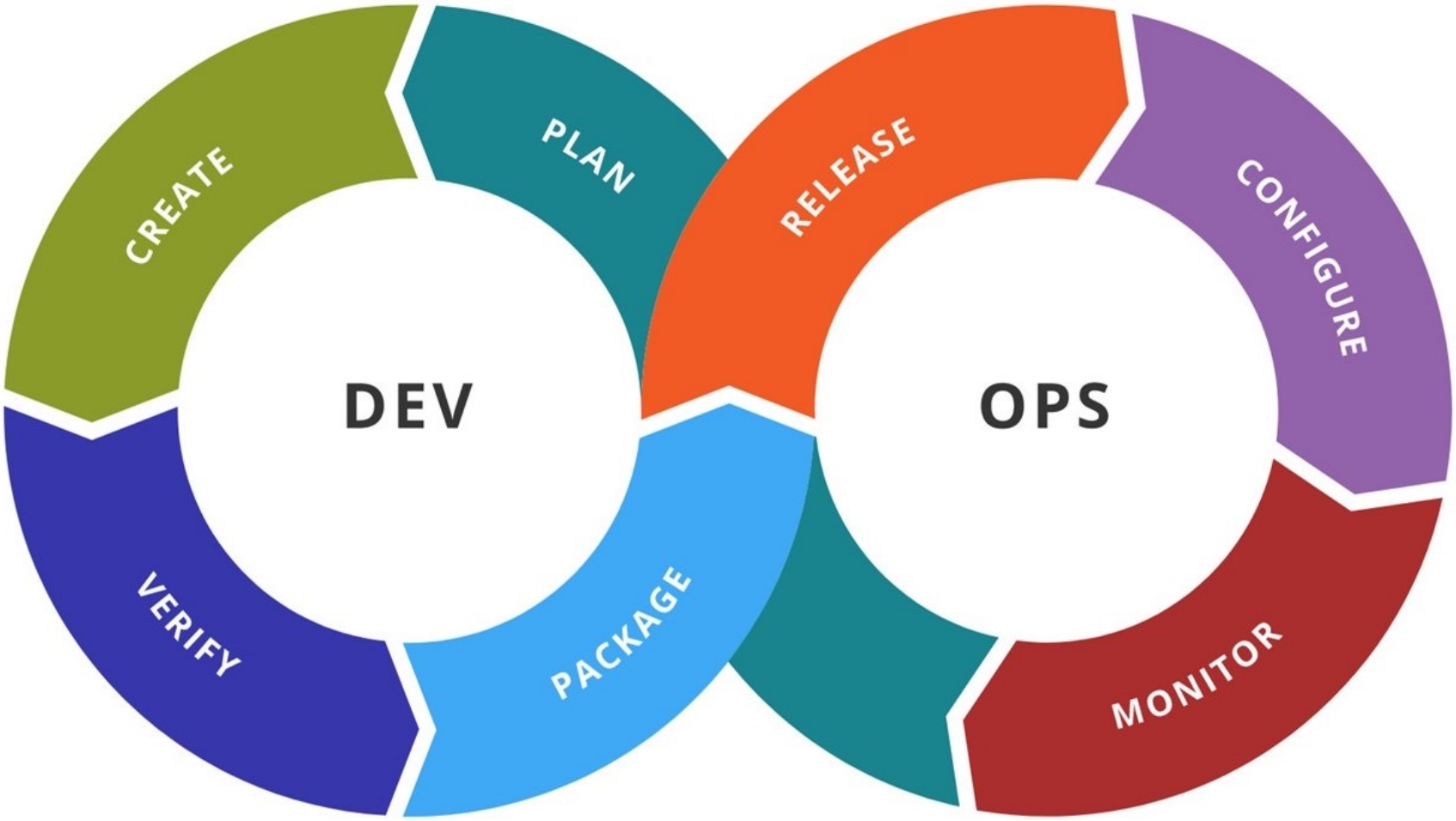
Riuso e integrazione



Agile


Start here
Deliver here
Customer
Hear customer story
Specify requirement(s)
Write test scenario(s)
Add new features; review, test, and rework
Demonstrate capabilities

Sviluppo e operatività



NARRAZIONE

Come evolve l'ingegneria del software per produrre sistemi intelligenti (SE → AI)

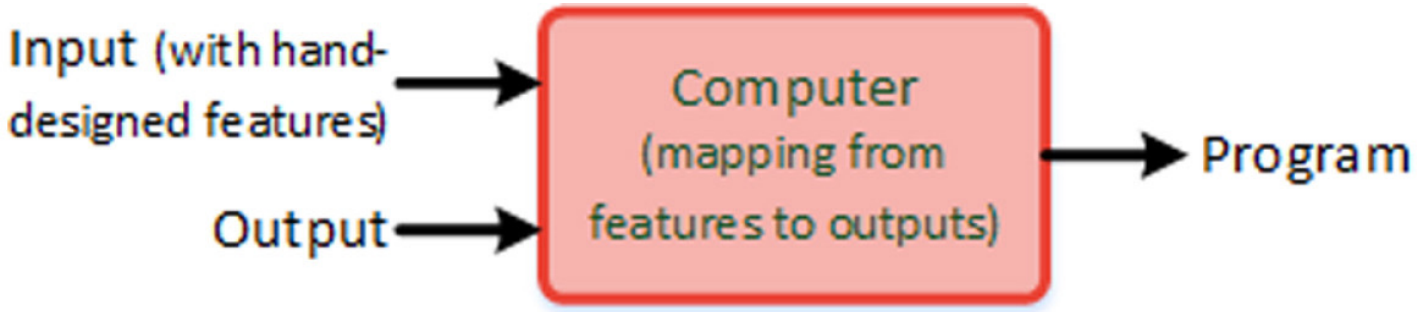
Software tradizionale e sistemi intelligenti

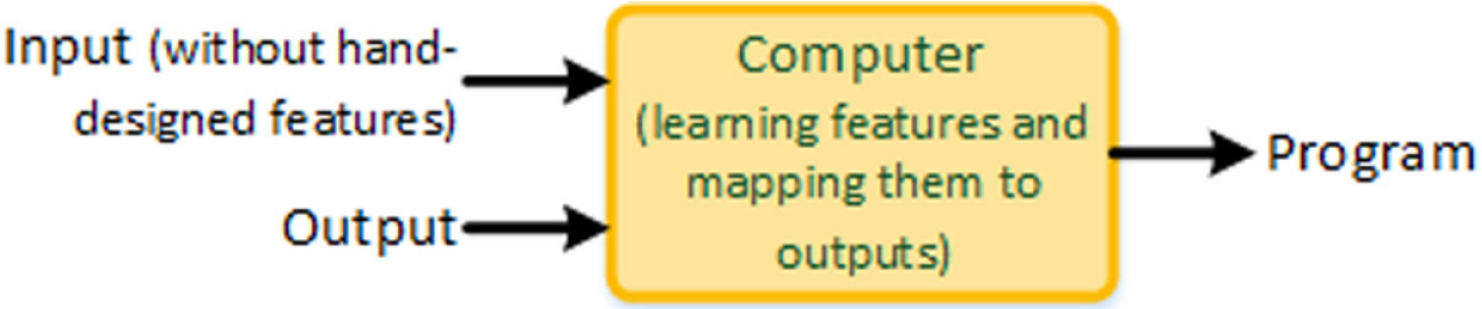
Contents lists available at ScienceDirect
The Journal of Systems & Software
journal homepage: www.elsevier.com/locate/jss

A software engineering perspective on engineering machine learning systems: State of the art and challenges[☆]

Görkem Giray

Sistemi intelligenti con ML e DL





Contents lists available at ScienceDirect
The Journal of Systems & Software
journal homepage: www.elsevier.com/locate/jss

A software engineering perspective on engineering machine learning systems: State of the art and challenges[☆]

Görkem Giray

Aree della SE che affrontano scenari di ML

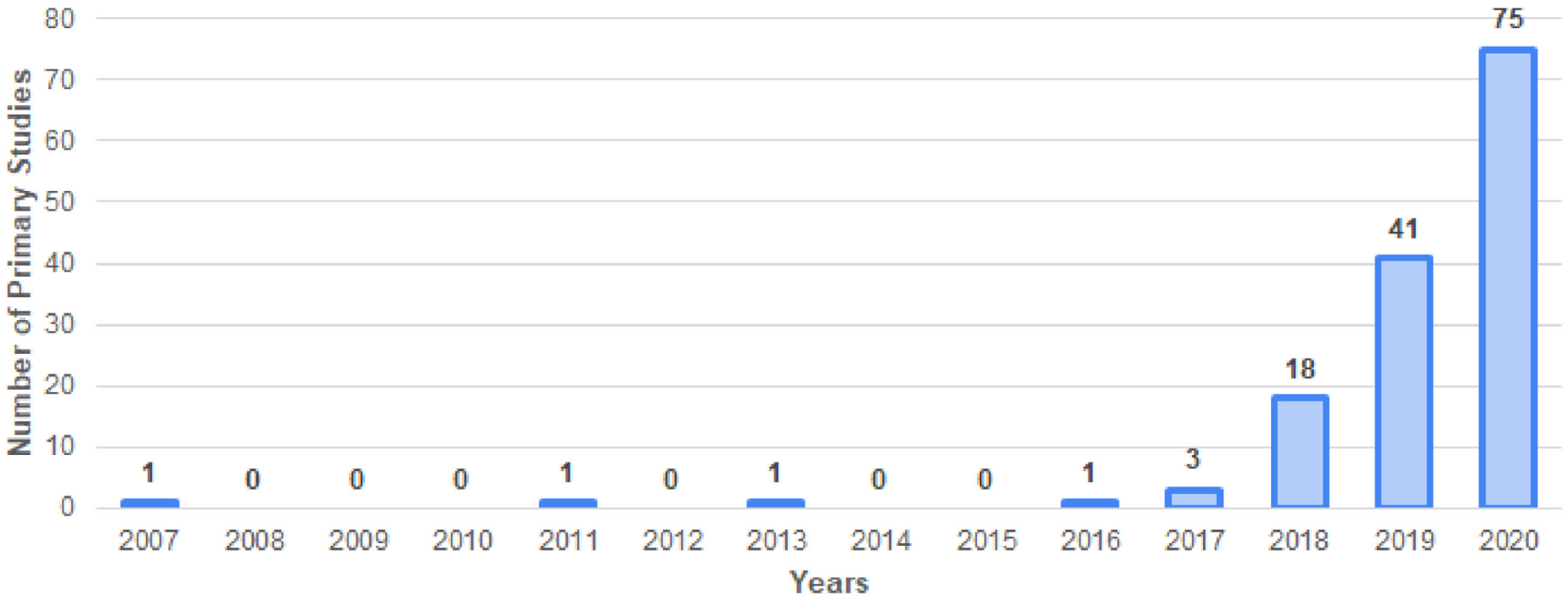
SE Knowledge Area / *Application Scenario*	Requirements Engineering	Design	Software Development & Tools	Testing & Quality	Maintenance & Configuration Management	Software Eng. Process & Management	Organizational Aspects
Image classification			13	44	3		
Classification (except image)	1		4	17	2	3	1
Autonomous driving	1		1	9	3	1	
Analysis of ML Framework/Library				4	1		
Prediction of a continuous value	1		2	2	1	1	1
NLP & Machine translation			1	2			
Object detection			2	1	1	1	
Ranking				1			
Mountain car problem				1			
No application scenario	1						

Contents lists available at ScienceDirect
The Journal of Systems & Software
journal homepage: www.elsevier.com/locate/jss

A software engineering perspective on engineering machine learning systems: State of the art and challenges☆

Görkem Giray

Numero di studi per anno

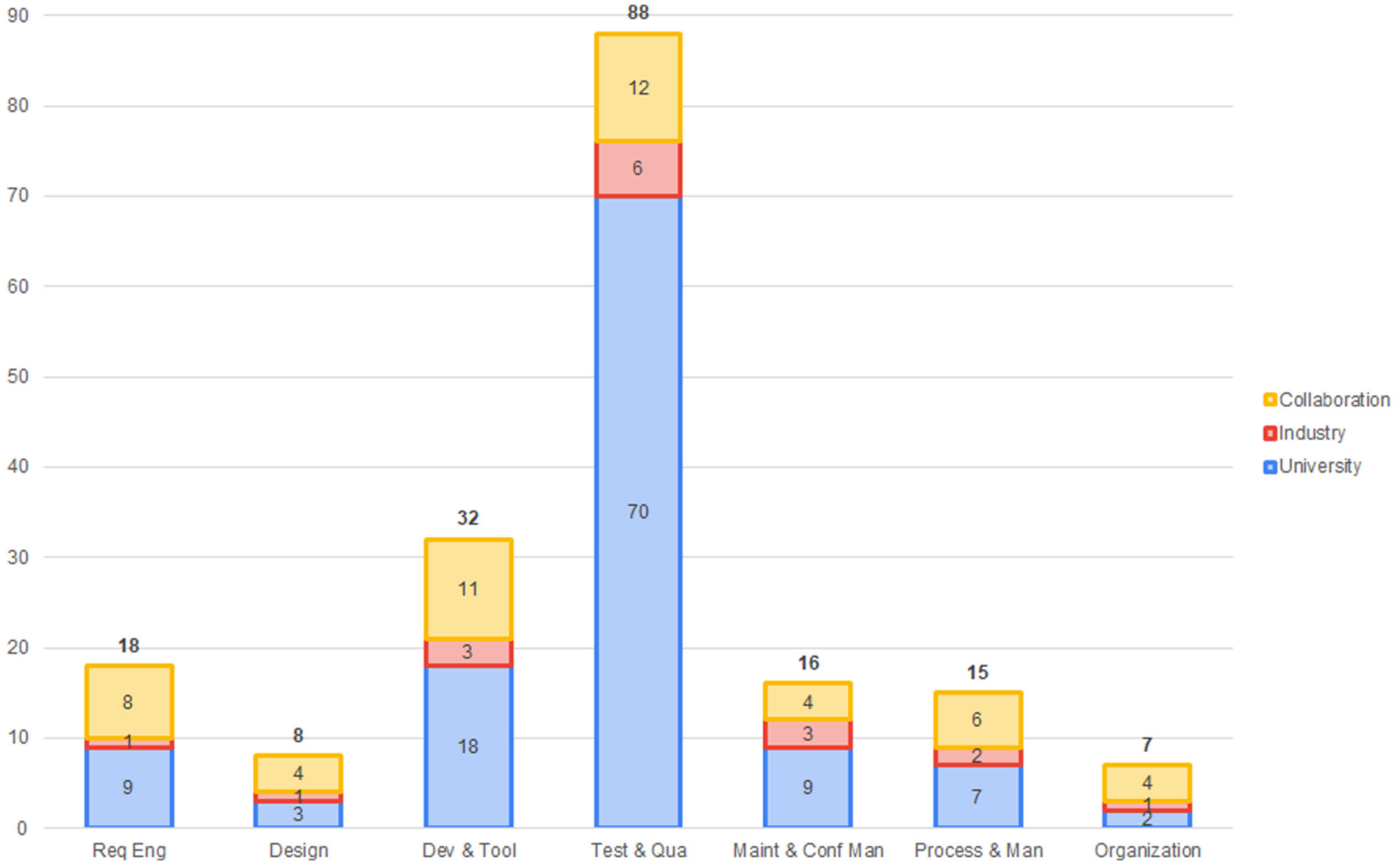


Contents lists available at ScienceDirect
The Journal of Systems & Software
journal homepage: www.elsevier.com/locate/jss

A software engineering perspective on engineering machine learning systems: State of the art and challenges☆

Görkem Giray

Provenienza degli studiosi per area della SE



Contents lists available at ScienceDirect
The Journal of Systems & Software
journal homepage: www.elsevier.com/locate/jss

A software engineering perspective on engineering machine learning systems: State of the art and challenges☆

Görkem Giray

Requisiti del software

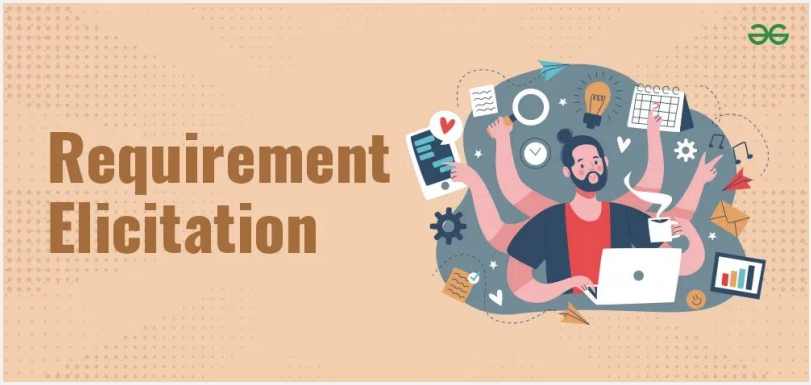
Gestire le aspettative dei committenti: già non è semplice illustrare le caratteristiche di un software tradizionale, ancora di più lo è per l'AI

Raccogliere e analizzare i requisiti: i sistemi di AI dipendono dalle caratteristiche dei dati che sono molto variabili in contesti diversi e cambiano costantemente

Specificare i requisiti: è meno efficace usare i casi d'uso sui dati quanto invece metriche quali precisione, accuratezza, recall

Gestire nuovi attributi di qualità: ad esempio la «explainability», la «sustainability», la «fairness» dell'AI

Gestire nuovi tipi di requisiti: la distinzione tra requisiti funzionali e non funzionali non si adatta bene ai dati



Progettazione del software

Nuovi schemi di progettazione per monitorare le prestazioni in produzione: «data drift»

Nuovi design pattern: è difficile isolare le dipendenze tra moduli in un sistema di ML

Nuovi modelli architetturali: vedi apprendimento federato

Progettare per gestire grandi volumi di dati: dati strutturati e dati non strutturati

Sviluppo del software

Gestire i dati: le fasi di preparazione dei data set sono nuove per chi sviluppa sistemi software tradizionali

Conoscere gli algoritmi di AI e ML e le differenti librerie disponibili

Sviluppare i modelli di AI e ML: definire le caratteristiche dei dati («feature»), addestrare i modelli, valutare i modelli

Gestire le dipendenze: non soltanto dalle librerie software esterne (tipico del software tradizionale) ma anche dai dati

Riusare modelli: pratica più difficile con l'AI e il ML

Usare nuovi strumenti di sviluppo: sia software sia hardware (GPU)

Verifica e qualità del software

Progettare casi di test: non sono sufficienti i casi d'uso ma devono anche essere considerate le configurazioni dei dati

Preparare i dati di test: richiede anche la raccolta di data set di test

Eseguire i test: richiede anche l'esecuzione degli algoritmi di ML, quindi la disponibilità dei data set e dell'infrastruttura di ML

Valutare i test: richiede la presenza di esperti di data science vista la natura non deterministica del ML

Debugging e correzione: richiede di eseguire debug e correzioni anche nei data set

Evoluzione e configurazione del software

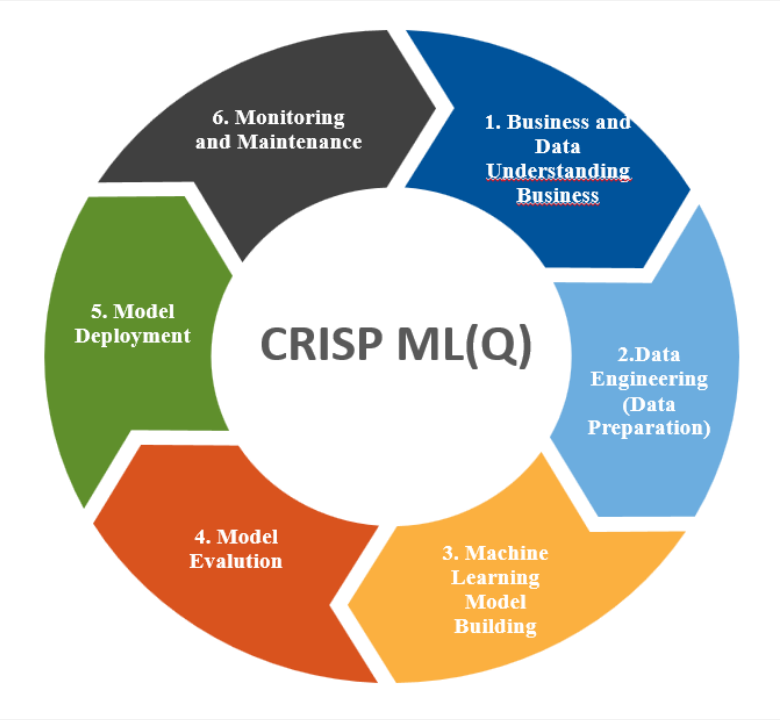
Gestire le diverse versioni dei dati e degli algoritmi di ML (ad esempio nuovi iperparametri)

Tenere traccia non solo delle versioni software ma anche degli esperimenti svolti sui dati

Gestire le necessità di addestrare nuovamente algoritmi in precedenza già allenati, ad esempio in seguito a eventi di «data drift»

Automatizzare i processi di CI/CD in modo da coinvolgere anche i dati nel processo di test/deployment

Evoluzione del modello DevOps



Il DevOps classico ha già subito evoluzioni ad esempio per garantire l'integrazione di test di sicurezza del software (**DevSecOps**)

Vi è la necessità di **armonizzare il DevOps** con i vari modelli del ciclo di vita del ML (Cross-Industry Standard Process for ML with Quality **CRISP-ML(Q)**)

Esistono alcune **proposte** che vanno maggiormente consolidate e diffuse

DataOps inserisce nel team DevOps anche i data engineer e i data scientist

ModelOps simile a DataOps ma con enfasi allo sviluppo di nuovi modelli di AI

Di fatto si tratta di **nuovi modelli di pipeline** per il DevOps

Evoluzione dei modelli agili

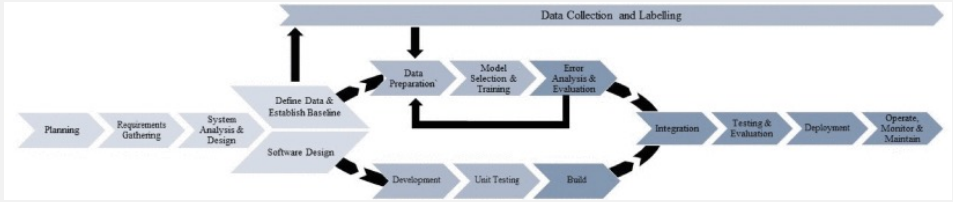
Anche i modelli agili del ciclo di vita del software stanno evolvendo in **due direzioni**

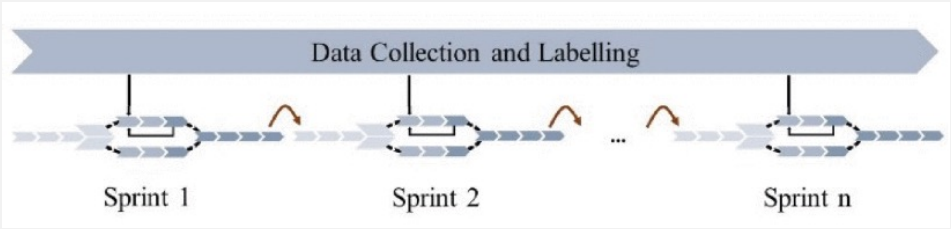
Modelli agili per lo sviluppo di sistemi intelligenti

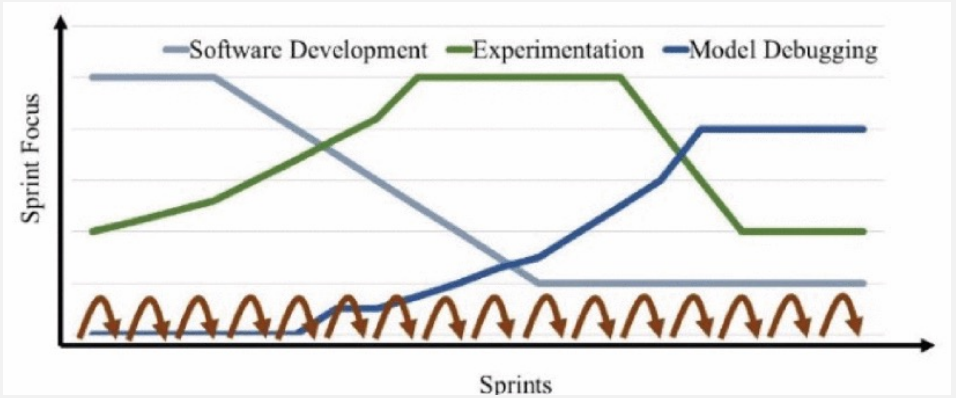
Uso di **algoritmi e strumenti di ML per** supportare i modelli agili di sviluppo del software

Vi è un'ampia evidenza empirica che l'uso di modelli agili tradizionali per lo sviluppo di sistemi intelligenti **non sia efficiente**

Serve una **maggiore rapidità delle iterazioni** e dei processi di integrazione e test con parti di sistema **non deterministiche** che solo l'elevata automatizzazione può supportare







Conferences > 2021 5th SLAAI International ...

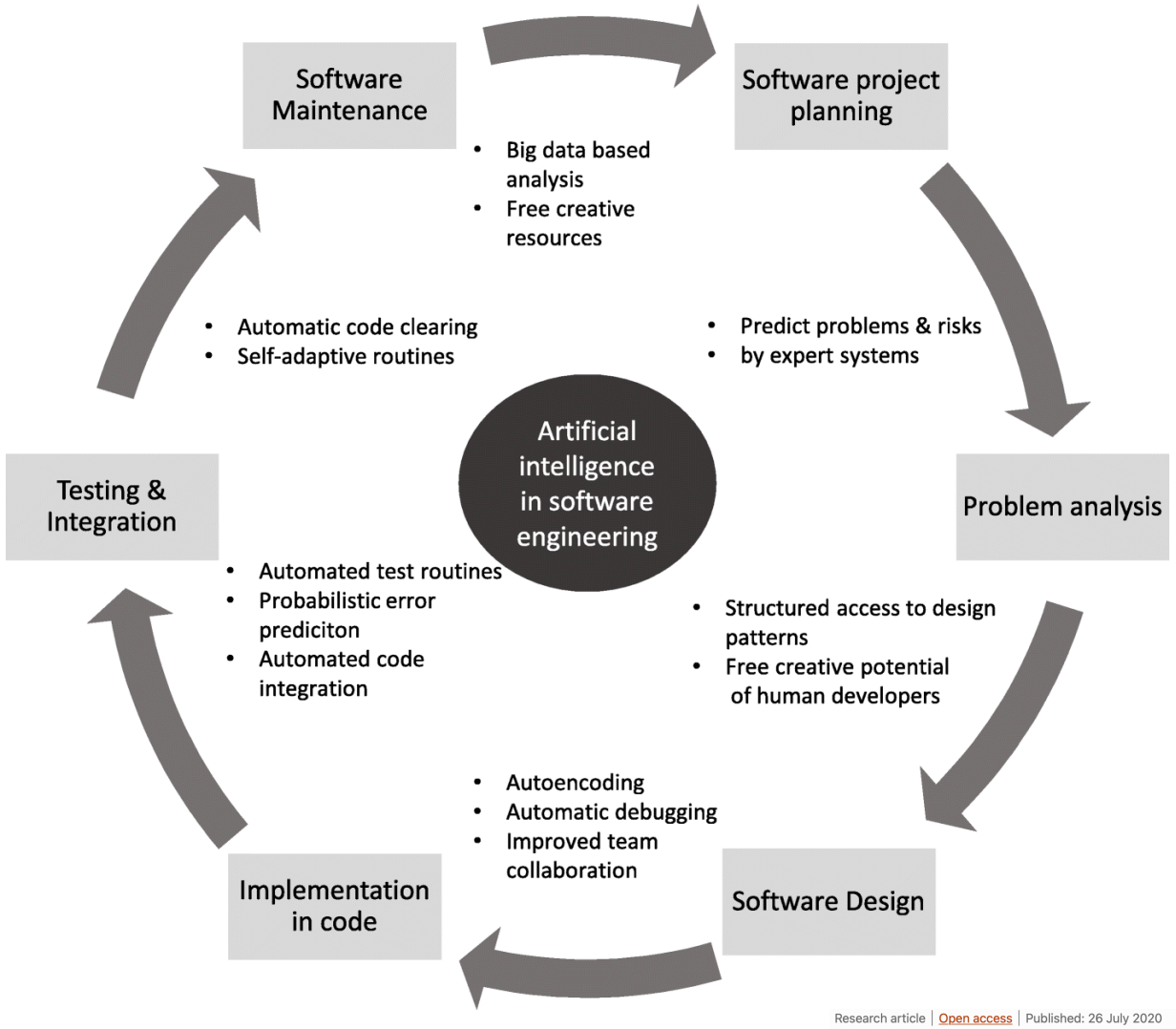
An Agile Software Development Life Cycle Model for Machine Learning Application Development

Publisher: IEEE | Cite This | PDF

Romesh Ranawana ; Asoka S. Karunananda | All Authors

Come l’intelligenza artificiale modifica la produzione del software (AI → SE)

Strumenti di AI nel ciclo di sviluppo della SE



Research article | Open access | Published: 26 July 2020

Applications of AI in classical software engineering

Marco Barenkamp, Jonas Rebstadt & Oliver Thomas

AI Perspectives 2, Article number: 1 (2020) | Cite this article

Springer Open

Pianificazione del software e analisi dei requisiti

Uso di strumenti di AI per **stimare durata e costi** di un progetto software

Per **assegnamento dei task** nel piano di lavoro

Per **ottimizzazione del piano di lavoro** in modo da minimizzare i costi e/o massimizzare la qualità del software

Uso di strumenti di AI per la **predizione di rischi** associati al progetto di sviluppo del software

Per **predizione di problemi** che saranno riscontrati durante il processo di sviluppo

Per **predizione di iterazioni critiche** e ausilio a riformulare il piano di lavoro

Progettazione del software

Uso di strumenti di AI generativa per **dettagliare la specifica** di casi d'uso

Per **definire casi di test** del software

Per **includere design pattern** adottati in precedenza in altri progetti

Per **predire il numero di difetti** attesi e la loro localizzazione nei componenti software

Per **ridurre il tempo** che sarebbe dedicato ad attività standard, così da aumentare la creatività dei progettisti e degli analisti

Strumenti di AI per la progettazione del software

In questo caso il supporto della AI è molto immaturo perché soprattutto volto allo **sviluppo del software**

Strumenti specifici che estraggano il modello del software tramite tecniche di AI sono **molto carenti**

Si predilige partire da una **narrazione dei requisiti** funzionali e non funzionali

Gli strumenti di sviluppo supportano il **riconoscimento automatico di design pattern**

Ci sono esperienze di uso dell'AI nel **gaming**, simulando azioni da parte di giocatori e aiutando a progettare strategie di gioco

Sviluppo del software

Uso di strumenti di AI generativa per **trasformare specifiche** testuali in codice

Per **debugging automatico** del codice e indicazioni operative al miglioramento

Per **generare automaticamente la documentazione** del codice

Per **ridurre i tempi** e i costi di sviluppo

Per favorire e **migliorare la collaborazione** tra i componenti di un gruppo di sviluppatori

Resta da affrontare il rischio che il codice prodotto automaticamente dalla AI **sia comprensibile e manutenibile**

Strumenti di AI per lo sviluppo del software

Cody by Sourcegraph: assistente open source per lo sviluppo del software

Tabnine: controllo degli errori, completamento del codice, refactoring

Mintlify: documentazione automatica

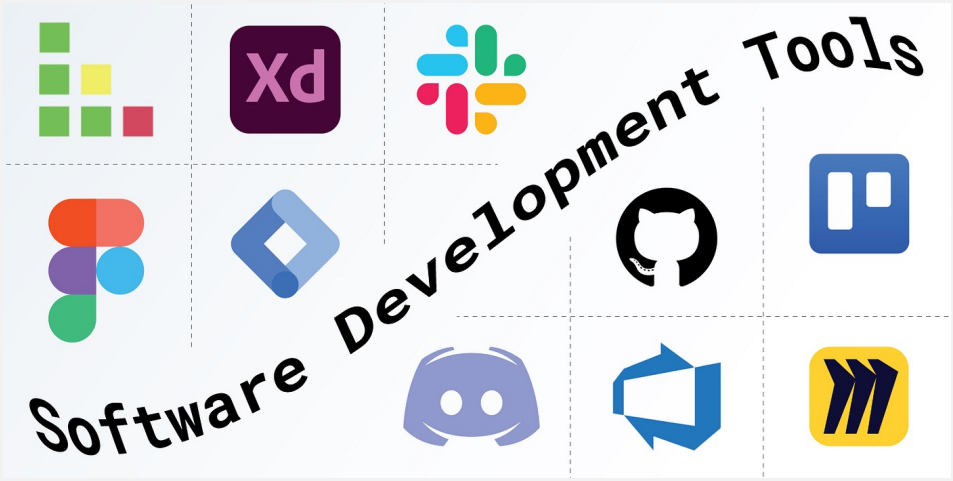
Code4Me: sviluppo, nato da un progetto di ricerca di TU Delft

IntelliJ AI-assistant: sviluppo, test, refactoring

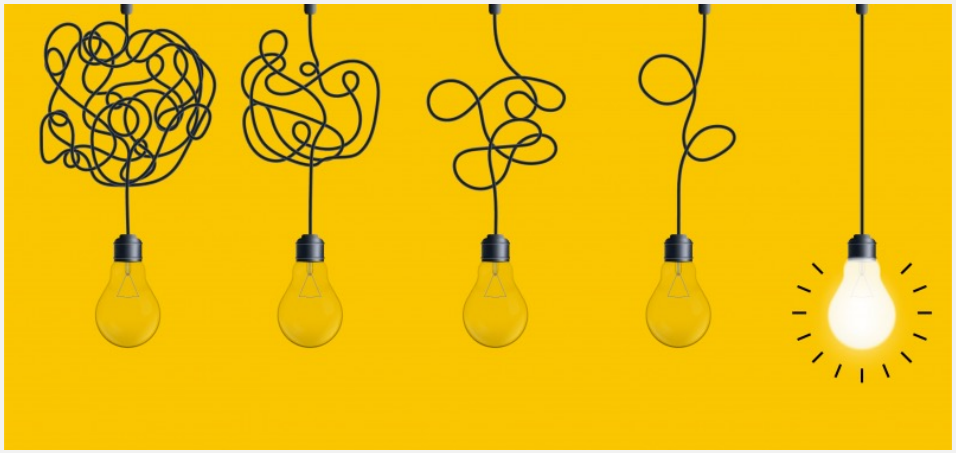
GitHub Copilot: sviluppo, controllo degli errori, documentazione

What The Diff: metriche di qualità, refactoring

Snyk: riconoscimento automatico di vulnerabilità nel codice



Comprensibilità del prodotto generato



Un tema aperto soprattutto per le **prime versioni** degli strumenti di sviluppo

Il tema di ricerca generale è la **Explainable AI**

Fornire link alle fonti

Strumenti di **descrizione** del codice generato

Dimostra la differenza tra la generazione del codice tramite modelli di AI e il prodotto di strumenti di sviluppo **low-code e no-code**

Proprietà del prodotto generato

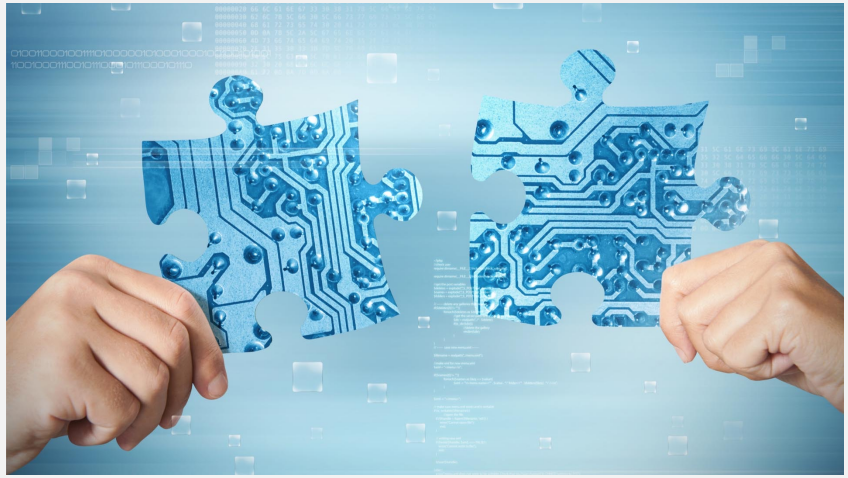
Rimane un **problema aperto** che richiede una normativa specifica

La EU **non** offre protezione con **copyright** per l'output dell'AI generativa

Il tema è **trasversale** a tutte le aree di applicazione della AI generativa

Nel caso del software il tema della **proprietà intellettuale** e della relativa tutela è complesso nel suo insieme, considerata la malleabilità del risultato

Test e integrazione del software



Uso di strumenti di AI per **generare codice di test** e test case

Per **generare dati di test**

Per **generare programmi «mutanti»**

Per **generare predizioni sul numero di difetti** e sulla loro localizzazione nel codice

Punto di attenzione sulla effettiva capacità di **comprendere i test generati** dalla AI

Uso di strumenti di AI per **velocizzare il processo di integrazione** di servizi in architetture cloud

Strumenti di AI per test e integrazione

Gli strumenti di AI per lo sviluppo del software in generale hanno **anche** funzionalità per il test

Le piattaforme consolidate per il test **aggiungono moduli** di AI

Selenium: generazione automatica di test case con l'AI

Code Intelligence: analisi dinamica delle vulnerabilità

Functionize: integrazione dei test nel DevOps grazie all'AI

Digital.ai Continuous Testing: copertura supportata dall'AI per i test funzionali, di performance e di accessibilità

ACCELQ: generazione di test supportata dall'AI

Manutenzione del software

Uso di strumenti di AI per **classificare le richieste di evoluzione** o di nuove funzionalità da parte degli utenti

Per **adattare dinamicamente i componenti** software alla qualità della rete dati

Per **identificare codice duplicato** o inutilizzato in sistemi legacy

Per **simulare attacchi di sicurezza** a sistemi esistenti in modo da far emergere potenziali vulnerabilità

Per **produrre automaticamente la documentazione** di sistemi software ad uso dei team di sviluppatori che si occuperanno della manutenzione evolutiva

Strumenti di AI per la manutenzione del software

Vale un discorso **analogo** a quanto affermato per l'attività di test e integrazione

L'uso dell'AI sposta l'attenzione verso la **manutenzione predittiva** (predictive analytics, condition monitoring, optimized scheduling, remote monitoring, automatic intervention)

Appvance: uso dell'AI generativa per produrre automaticamente percorsi di test e manutenzione

DeepCode: uso dell'AI per predire e identificare difetti e vulnerabilità nel codice

CodeGuru: uso dell'AI per automatizzare la review e l'analisi del codice

EPILOGO

Il punto di vista del manager

MIT: il **94% dei leader aziendali** impiega già l'IA. Il 38% degli intervistati si aspetta un impatto "sostanziale" sul software lifecycle entro **uno-tre anni**, mentre un ulteriore 31% entro **quattro-dieci anni**

L'integrazione dell'IA nei processi di sviluppo **non è automatico**: l'organizzazione della software factory deve essere ripensata

Policy sull'uso responsabile dell'IA: governance con "**human-in-the-loop**"

Il **25% del codice è già generato** automaticamente: lo sviluppatore deve diventare un **direttore** di sistemi

I **contratti** verso il cliente devono essere rivisti: il cliente deve **pagare codice generato automaticamente**?

Il punto di vista del software engineer

Affidabilità: il codice generato contiene bug e vulnerabilità; i generatori non comprendono il contesto

Sicurezza: i generatori creano dipendenze a librerie con problemi di licenze e di sicurezza, a volte librerie inesistenti

Proprietà intellettuale: rischio di plagio o uso improprio di licenze (es. GPL, copyleft)

Approccio cognitivo: eccessiva dipendenza dall'IA, automatizzazione senza comprensione

Manutenzione: codice poco leggibile, Difficile da integrare o testare

Rischi etici e legali: chi è responsabile di codice generato e non funzionante?

Luca Mainetti

Ordinario di Ingegneria del Software

Dip. Ingegneria dell'Innovazione

Università del Salento

luca.mainetti@unisalento.it

39 335 6614327

UNIVERSITÀ DEL SALENTO