Progetti di Ingegneria, Gestione, ed Evoluzione del Software

A.A. 2022/23

Tutor:
Manuel De Stefano
Emanuele Iannone

17/03/2023

Regole di Ingaggio

In questa presentazione vedremo alcune possibili idee di progetto, che consistono in manutenzione ed evoluzione di applicativi esistenti. Questi applicativi possono essere tool/script di analisi/manutenzione del codice e modelli predittivi per assistere attività di sviluppo.

Regole di Ingaggio

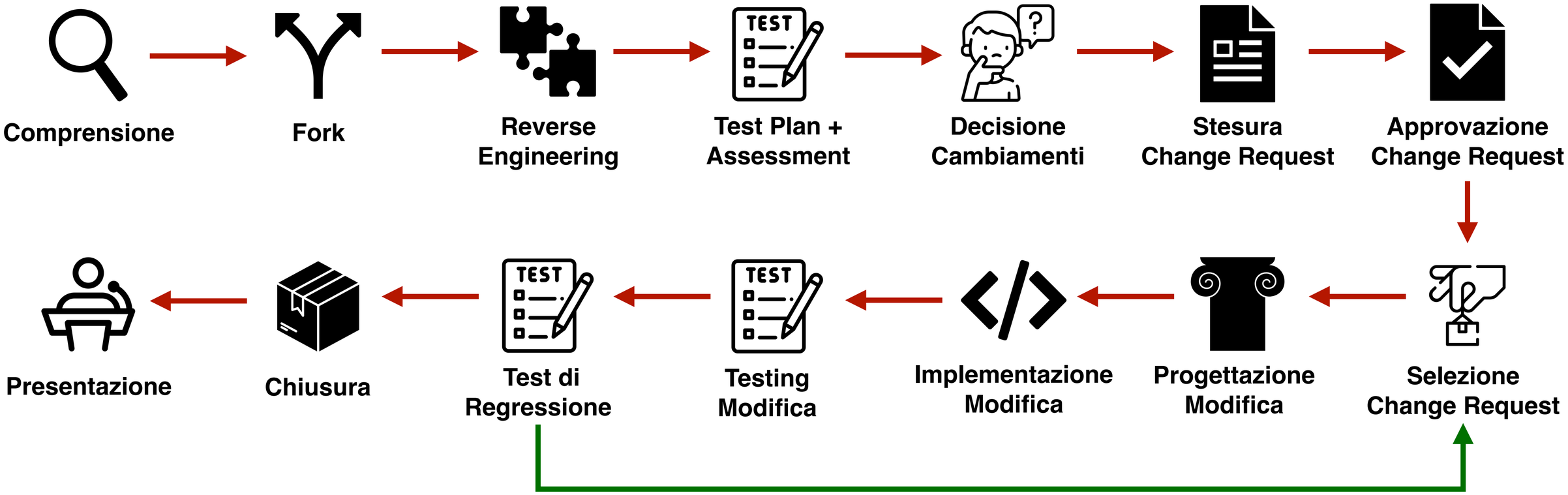
In questa presentazione vedremo alcune possibili idee di progetto, che consistono in manutenzione ed evoluzione di applicativi esistenti. Questi applicativi possono essere tool/script di analisi/manutenzione del codice e modelli predittivi per assistere attività di sviluppo.

Questo è il processo che si **dovrebbe** seguire per il completamento del progetto:

Regole di Ingaggio

In questa presentazione vedremo alcune possibili idee di progetto, che consistono in manutenzione ed evoluzione di applicativi esistenti. Questi applicativi possono essere tool/script di analisi/manutenzione del codice e modelli predittivi per assistere attività di sviluppo.

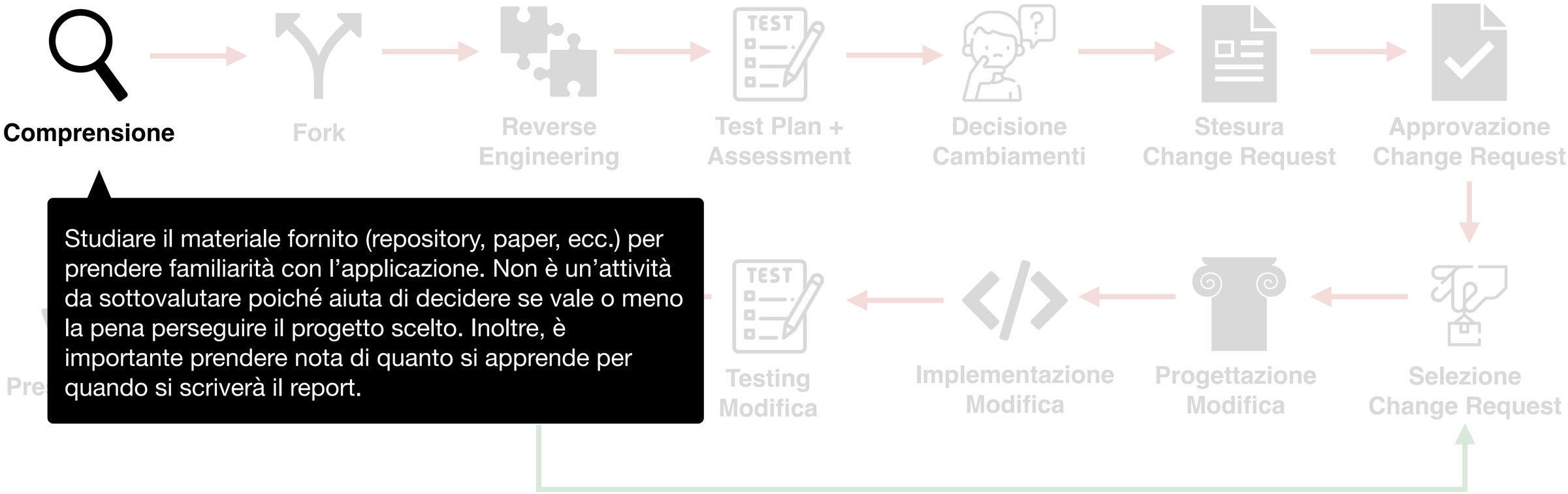
Questo è il processo che si **dovrebbe** seguire per il completamento del progetto:



Regole di Ingaggio

In questa presentazione vedremo alcune possibili idee di progetto, che consistono in manutenzione ed evoluzione di applicativi esistenti. Questi applicativi possono essere tool/script di analisi/manutenzione del codice e modelli predittivi per assistere attività di sviluppo.

Questo è il processo che si **dovrebbe** seguire per il completamento del progetto:

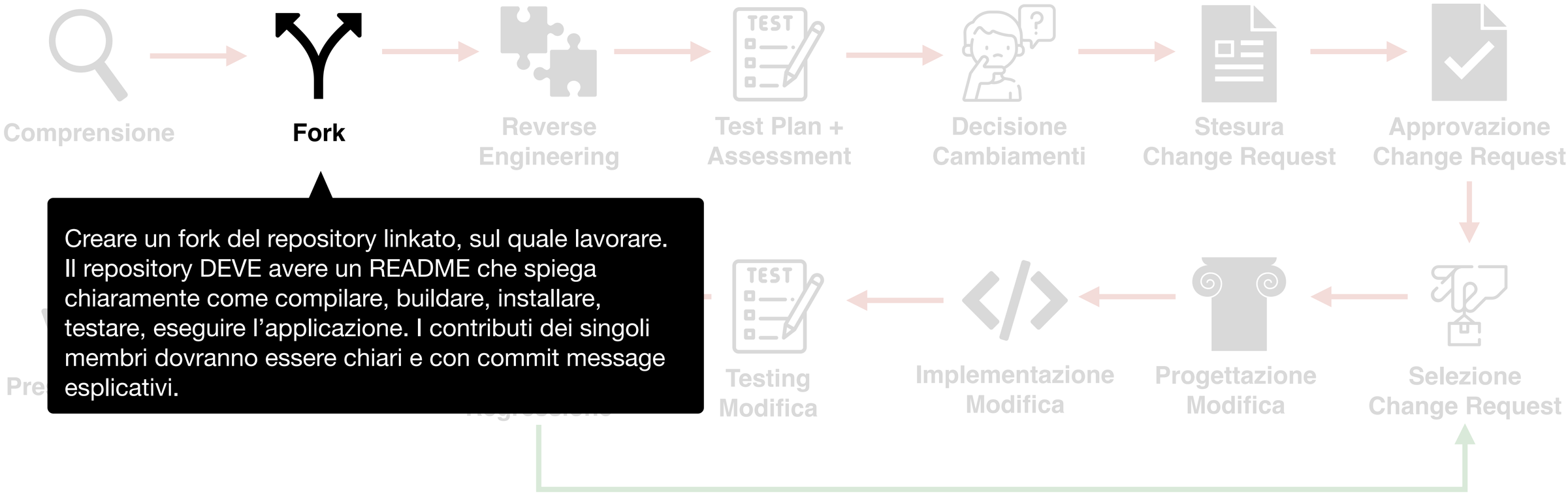


Studiare il materiale fornito (repository, paper, ecc.) per prendere familiarità con l'applicazione. Non è un'attività da sottovalutare poiché aiuta di decidere se vale o meno la pena perseguire il progetto scelto. Inoltre, è importante prendere nota di quanto si apprende per quando si scriverà il report.

Regole di Ingaggio

In questa presentazione vedremo alcune possibili idee di progetto, che consistono in manutenzione ed evoluzione di applicativi esistenti. Questi applicativi possono essere tool/script di analisi/manutenzione del codice e modelli predittivi per assistere attività di sviluppo.

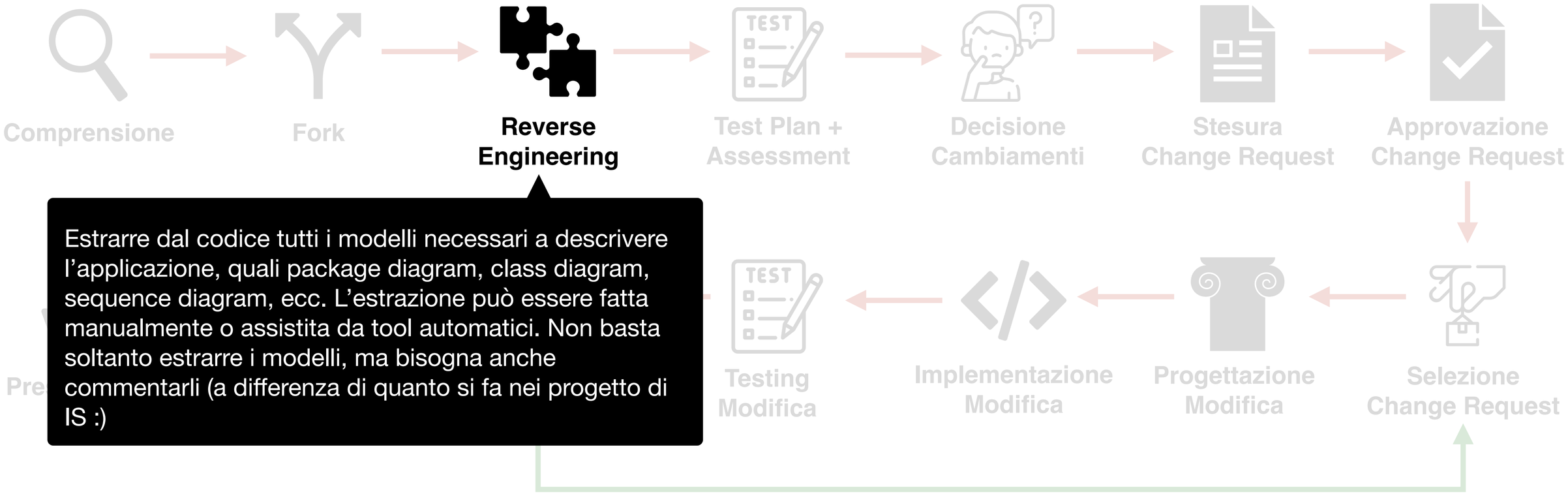
Questo è il processo che si **dovrebbe** seguire per il completamento del progetto:



Regole di Ingaggio

In questa presentazione vedremo alcune possibili idee di progetto, che consistono in manutenzione ed evoluzione di applicativi esistenti. Questi applicativi possono essere tool/script di analisi/manutenzione del codice e modelli predittivi per assistere attività di sviluppo.

Questo è il processo che si **dovrebbe** seguire per il completamento del progetto:

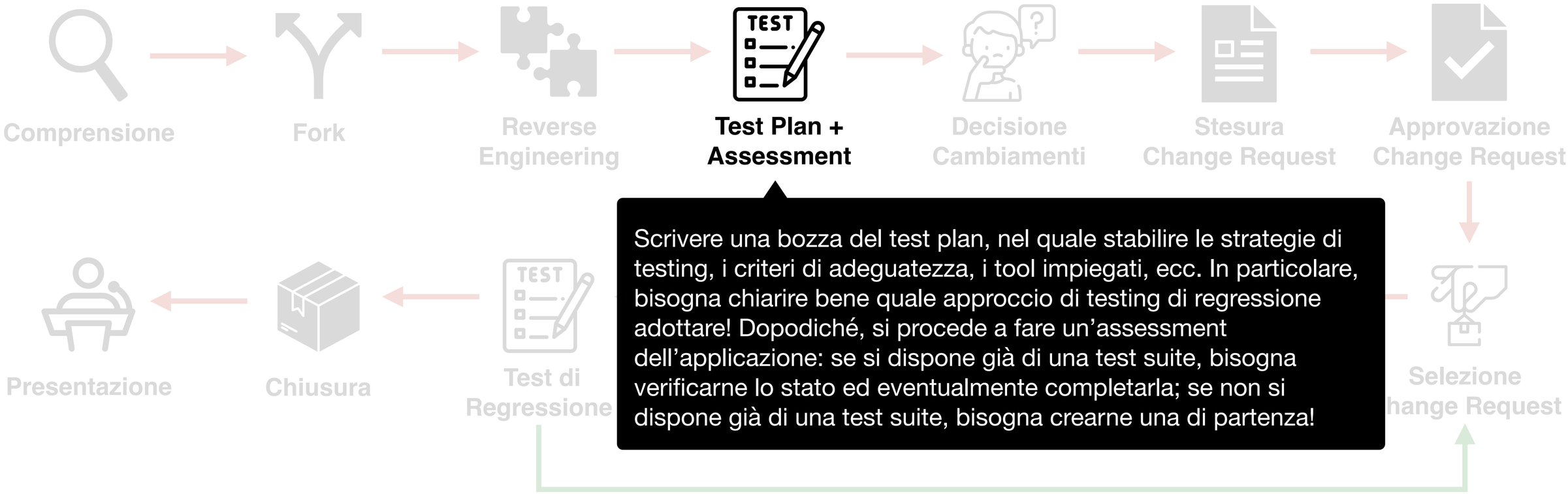


Estrarre dal codice tutti i modelli necessari a descrivere l'applicazione, quali package diagram, class diagram, sequence diagram, ecc. L'estrazione può essere fatta manualmente o assistita da tool automatici. Non basta soltanto estrarre i modelli, ma bisogna anche commentarli (a differenza di quanto si fa nei progetto di IS :)

Regole di Ingaggio

In questa presentazione vedremo alcune possibili idee di progetto, che consistono in manutenzione ed evoluzione di applicativi esistenti. Questi applicativi possono essere tool/script di analisi/manutenzione del codice e modelli predittivi per assistere attività di sviluppo.

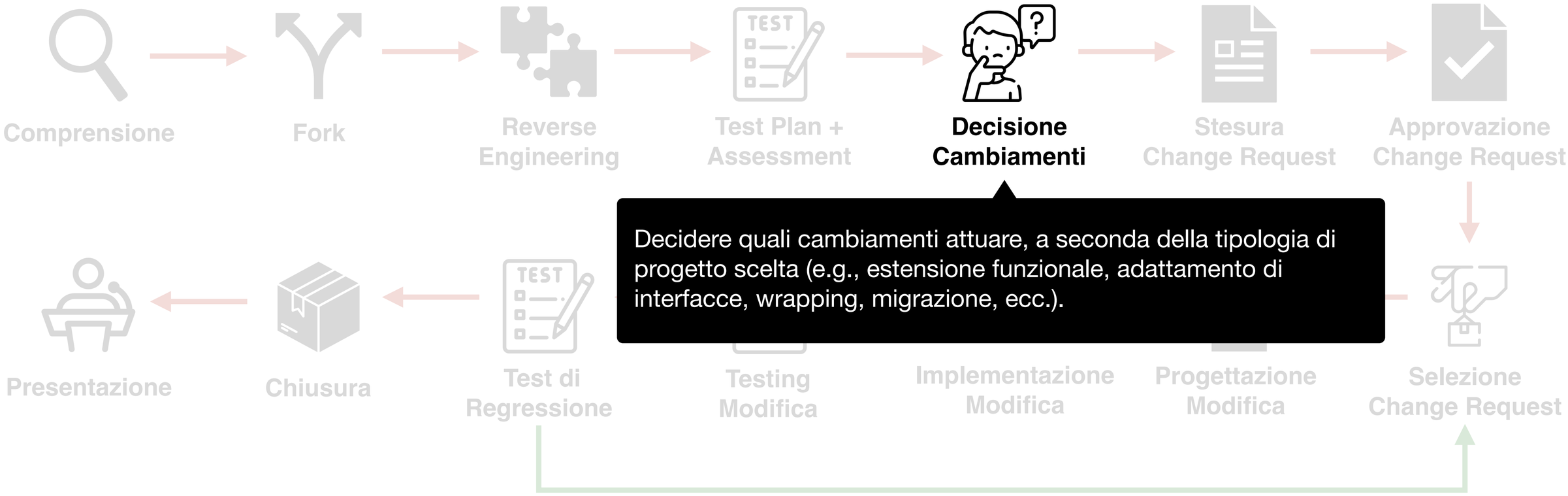
Questo è il processo che si **dovrebbe** seguire per il completamento del progetto:



Regole di Ingaggio

In questa presentazione vedremo alcune possibili idee di progetto, che consistono in manutenzione ed evoluzione di applicativi esistenti. Questi applicativi possono essere tool/script di analisi/manutenzione del codice e modelli predittivi per assistere attività di sviluppo.

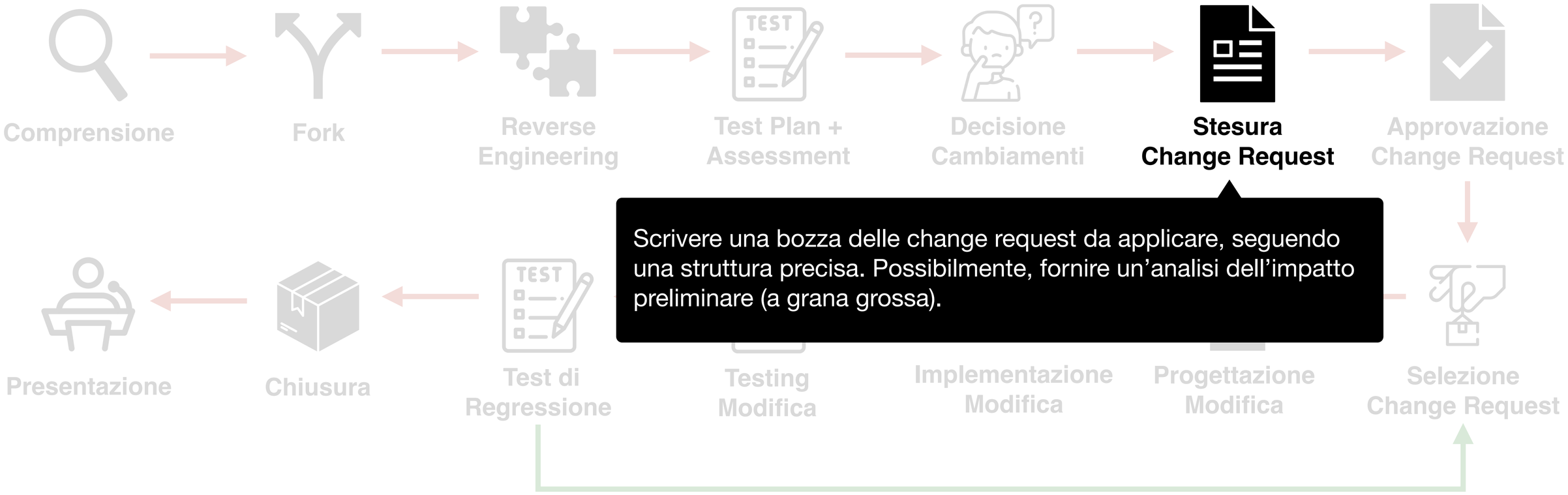
Questo è il processo che si **dovrebbe** seguire per il completamento del progetto:



Regole di Ingaggio

In questa presentazione vedremo alcune possibili idee di progetto, che consistono in manutenzione ed evoluzione di applicativi esistenti. Questi applicativi possono essere tool/script di analisi/manutenzione del codice e modelli predittivi per assistere attività di sviluppo.

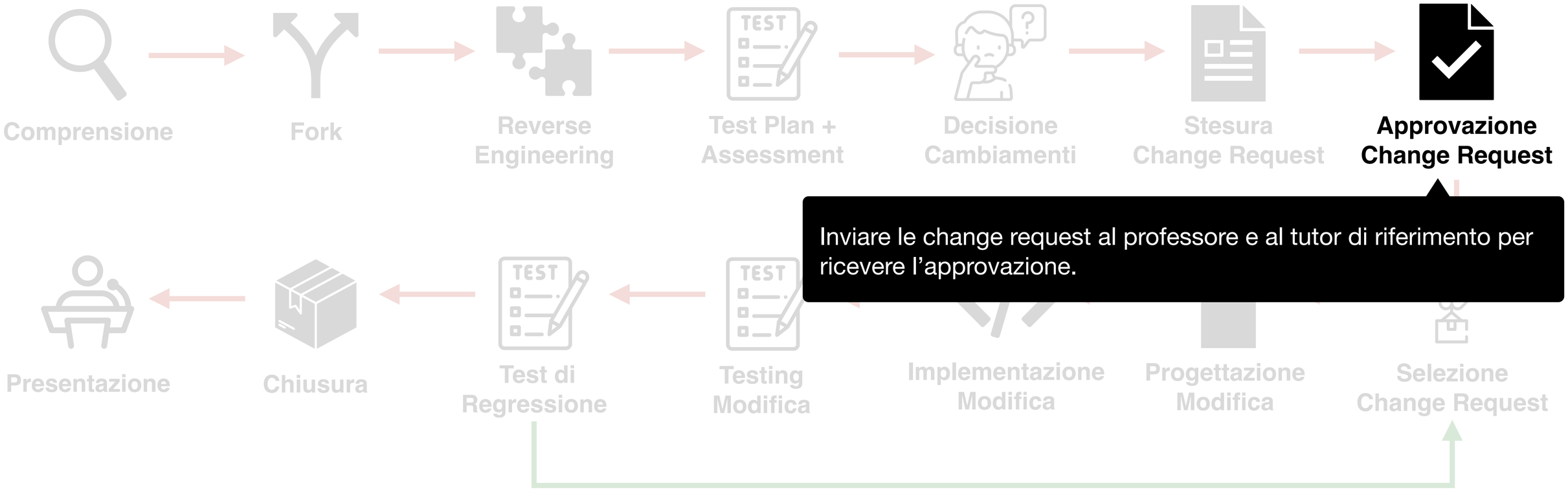
Questo è il processo che si **dovrebbe** seguire per il completamento del progetto:



Regole di Ingaggio

In questa presentazione vedremo alcune possibili idee di progetto, che consistono in manutenzione ed evoluzione di applicativi esistenti. Questi applicativi possono essere tool/script di analisi/manutenzione del codice e modelli predittivi per assistere attività di sviluppo.

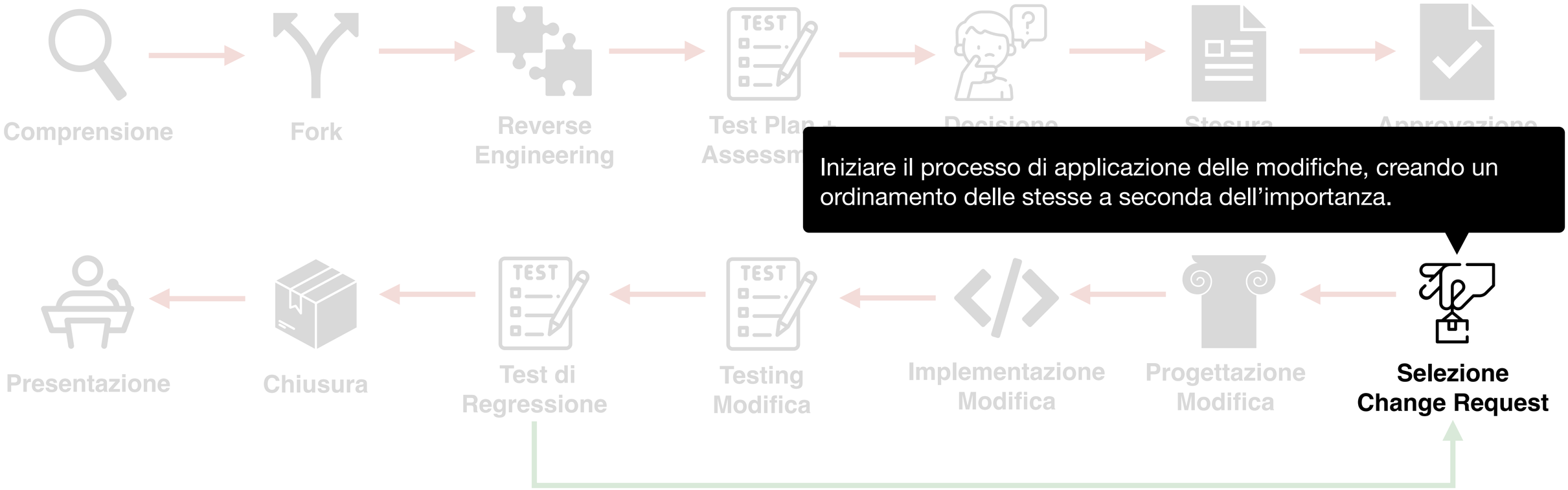
Questo è il processo che si **dovrebbe** seguire per il completamento del progetto:



Regole di Ingaggio

In questa presentazione vedremo alcune possibili idee di progetto, che consistono in manutenzione ed evoluzione di applicativi esistenti. Questi applicativi possono essere tool/script di analisi/manutenzione del codice e modelli predittivi per assistere attività di sviluppo.

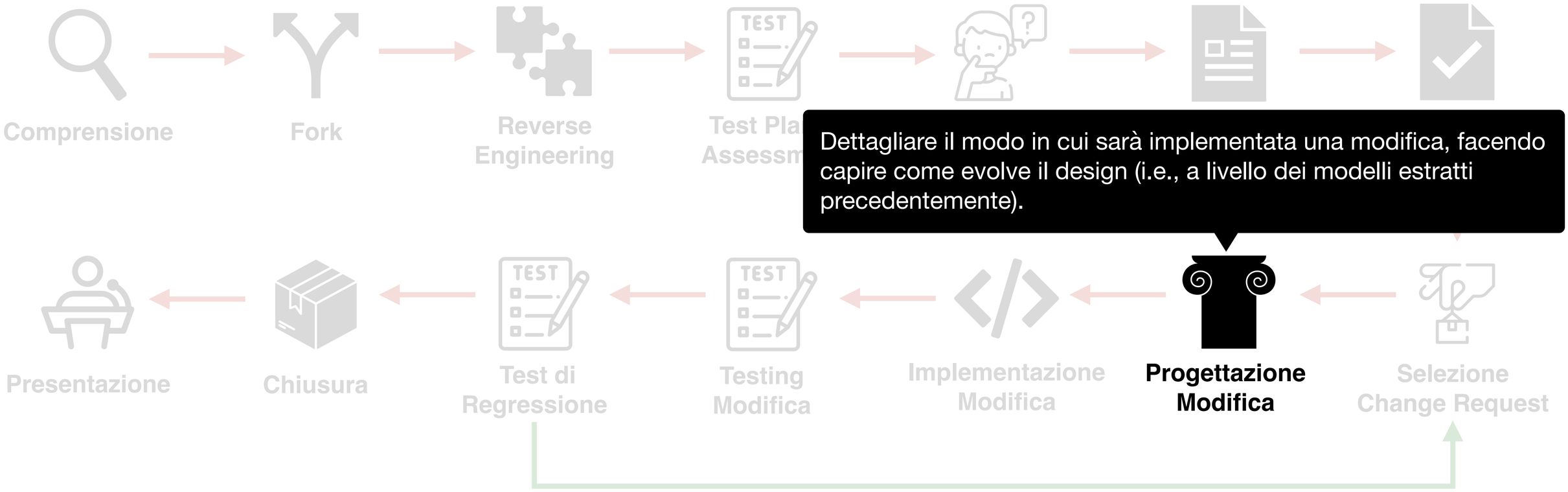
Questo è il processo che si **dovrebbe** seguire per il completamento del progetto:



Regole di Ingaggio

In questa presentazione vedremo alcune possibili idee di progetto, che consistono in manutenzione ed evoluzione di applicativi esistenti. Questi applicativi possono essere tool/script di analisi/manutenzione del codice e modelli predittivi per assistere attività di sviluppo.

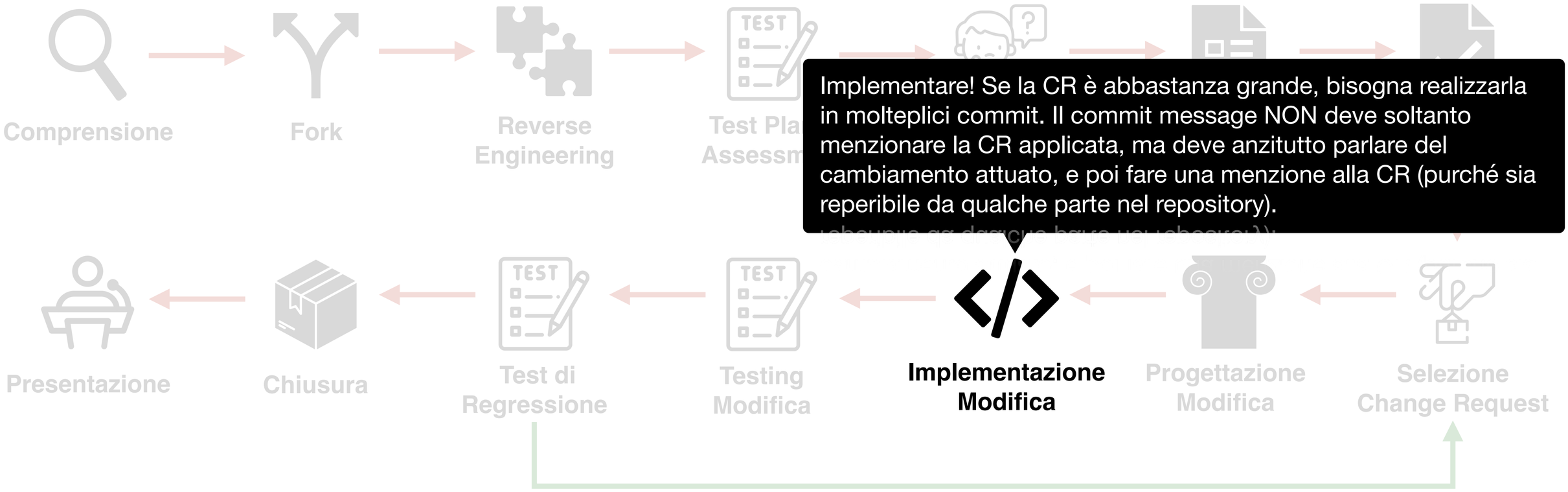
Questo è il processo che si **dovrebbe** seguire per il completamento del progetto:



Regole di Ingaggio

In questa presentazione vedremo alcune possibili idee di progetto, che consistono in manutenzione ed evoluzione di applicativi esistenti. Questi applicativi possono essere tool/script di analisi/manutenzione del codice e modelli predittivi per assistere attività di sviluppo.

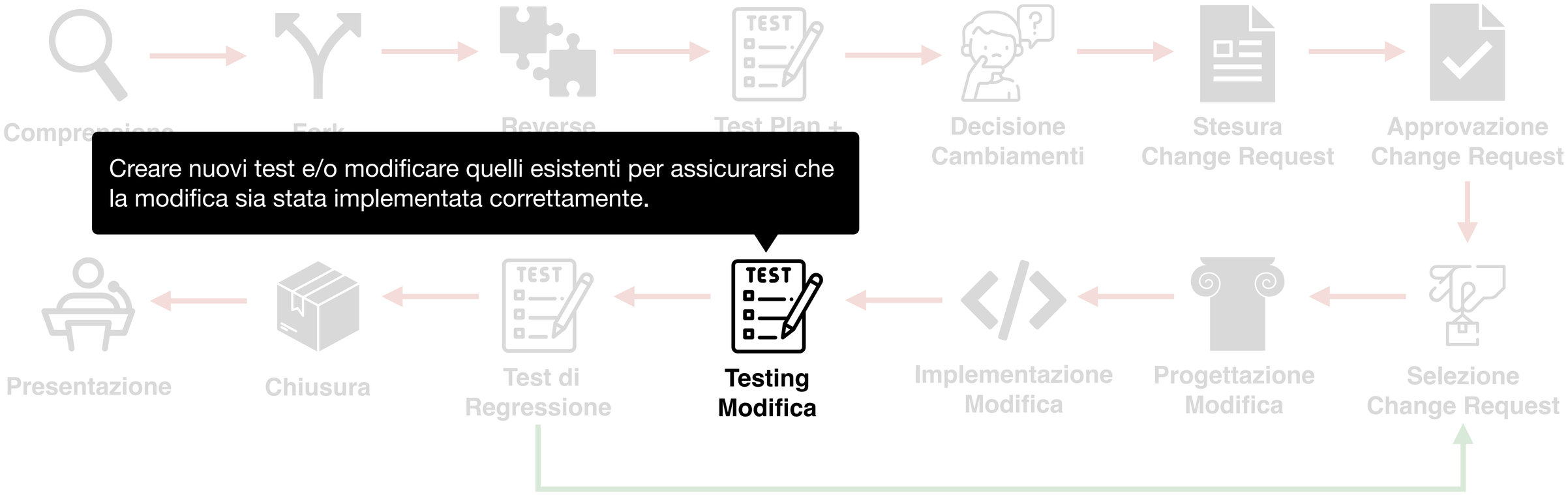
Questo è il processo che si **dovrebbe** seguire per il completamento del progetto:



Regole di Ingaggio

In questa presentazione vedremo alcune possibili idee di progetto, che consistono in manutenzione ed evoluzione di applicativi esistenti. Questi applicativi possono essere tool/script di analisi/manutenzione del codice e modelli predittivi per assistere attività di sviluppo.

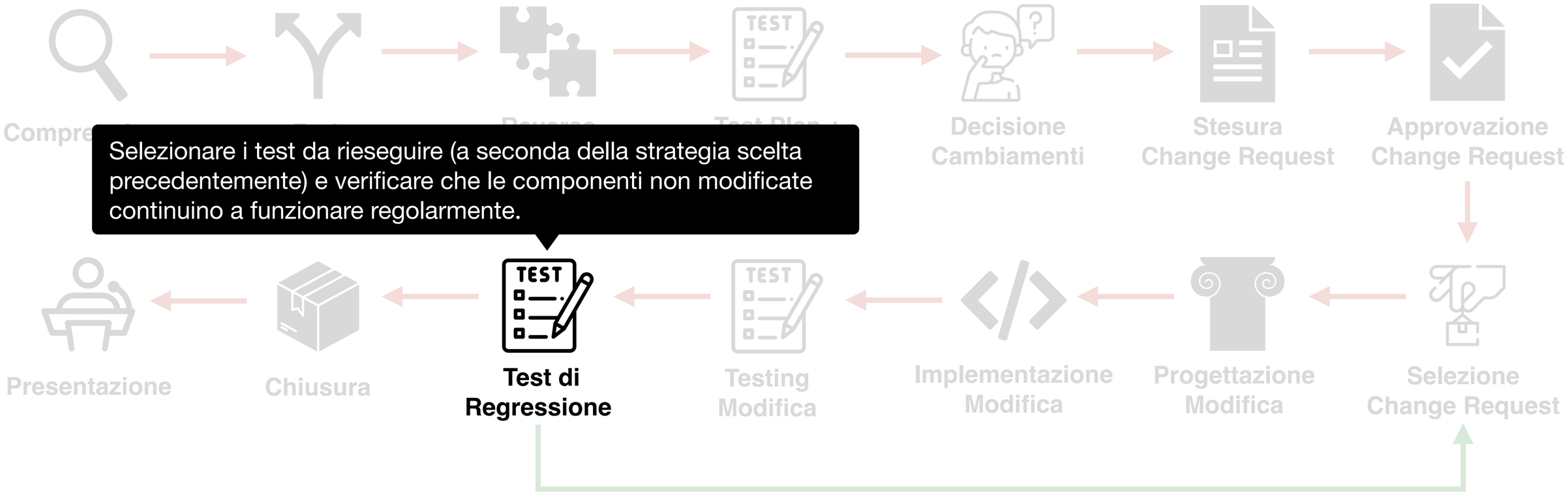
Questo è il processo che si **dovrebbe** seguire per il completamento del progetto:



Regole di Ingaggio

In questa presentazione vedremo alcune possibili idee di progetto, che consistono in manutenzione ed evoluzione di applicativi esistenti. Questi applicativi possono essere tool/script di analisi/manutenzione del codice e modelli predittivi per assistere attività di sviluppo.

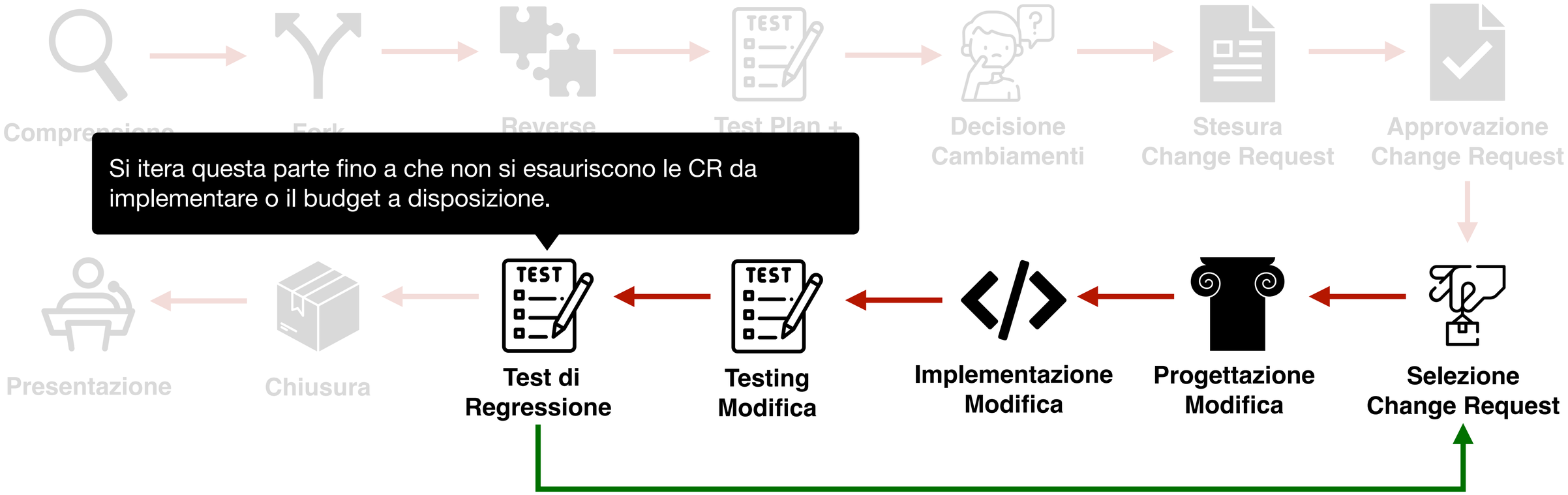
Questo è il processo che si **dovrebbe** seguire per il completamento del progetto:



Regole di Ingaggio

In questa presentazione vedremo alcune possibili idee di progetto, che consistono in manutenzione ed evoluzione di applicativi esistenti. Questi applicativi possono essere tool/script di analisi/manutenzione del codice e modelli predittivi per assistere attività di sviluppo.

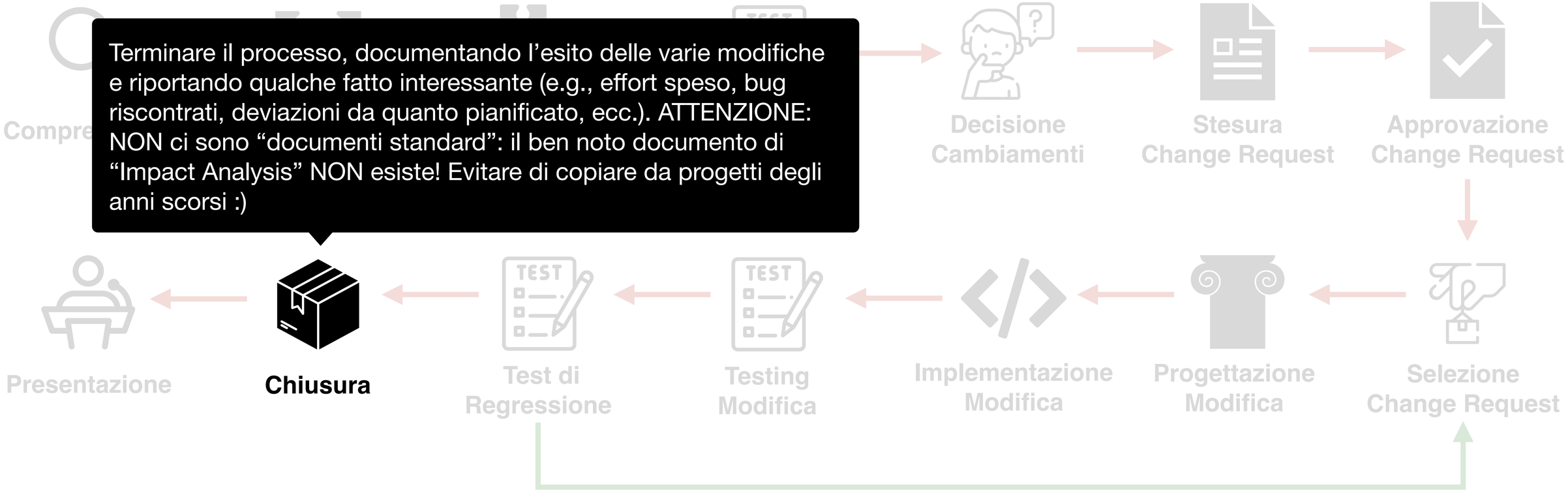
Questo è il processo che si **dovrebbe** seguire per il completamento del progetto:



Regole di Ingaggio

In questa presentazione vedremo alcune possibili idee di progetto, che consistono in manutenzione ed evoluzione di applicativi esistenti. Questi applicativi possono essere tool/script di analisi/manutenzione del codice e modelli predittivi per assistere attività di sviluppo.

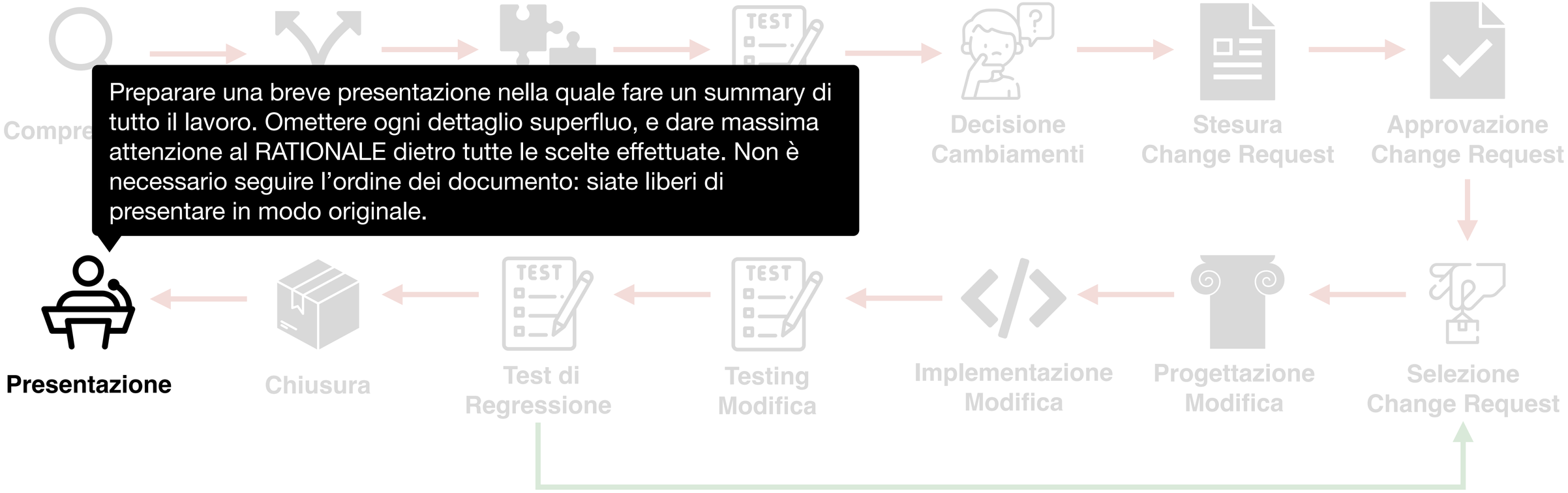
Questo è il processo che si **<u>dovrebbe</u>** seguire per il completamento del progetto:



Regole di Ingaggio

In questa presentazione vedremo alcune possibili idee di progetto, che consistono in manutenzione ed evoluzione di applicativi esistenti. Questi applicativi possono essere tool/script di analisi/manutenzione del codice e modelli predittivi per assistere attività di sviluppo.

Questo è il processo che si **dovrebbe** seguire per il completamento del progetto:



SURFACE

A lightweight tool for measuring the security profile of Java applications

Surface è un tool scritto in Java eseguibile da linea di comando per estrarre metriche di sicurezza di progetti Java. Esso è in grado anche di compiere analisi storiche di repository Git.

Idea #1

Introdurre l'analisi differenziale, ovvero supportare il calcolo di metriche e statistiche per confrontare coppie, set, range di commit e release (attualmente non supportati esplicitamente).

Idea #2

Fare in modo che Surface accetti un file di configurazione che definisca le **regole di detection** invece che usare regole hard-coded. In più, fare in modo che Surface accetti la definizione di metriche custom tramite un "linguaggio" ad hoc.

Idea #3

Separare la logica "core" da quella della command-line interface. Dopodiché, costruire una libreria (si potrebbe distribuire pubblicamente) che espone le classi e i metodi essenziali per lanciare le analisi (i.e., una programming API).

Repository: https://github.com/emaiannone/surface

Paper Security Metrics: https://ieeexplore.ieee.org/document/6004330

Infozilla

Unstructured software engineering data mining tool. It can find and extract source code regions, patches, stack traces, enumerations and itemizations from discussion threads

Infozilla è un tool scritto in Java per estrarre dati non strutturati da bug report riguardanti progetti Java. Esso è in grado di identificare ed estrarre snippet di codice, stack trace, ecc. tramite un approccio basato su keyword ed espressioni regolari. Il tool non è molto aggiornato…

Idea #1

Aggiungere il supporto ad altri linguaggi di programmazione, come C/C++, Python, Javascript. Bisognerebbe anche assicurarsi che il tool funzioni anche per le feature delle ultime versioni di Java.

Idea #2

Introdurre migliorie di vario genere, quali (1) lancio dell'analisi direttamente su un URL fornito in input, (2) export in diversi formati, (3) configurazione custom delle keyword/regex. È possibile identificare altri miglioramenti.

Repository: https://github.com/kuyio/infozilla

Paper Infozilla: https://dl.acm.org/doi/10.1145/1370750.1370757

Vulnerability Prediction Models

Just a “sparse” collection of vulnerability ML/DL prediction models

Modelli di ML/DL implementati in Python con dei semplici script o Jupyter notebook. Nessuno di essi è contenuto in un tool, e replicare ciascuno di essi richiede un effort non indifferente. Oltretutto, per quanto concettualmente simili, non sono eseguibili in modo uniforme (per il training, testing, e il deployment).

Idea

Offrire un'unica interfaccia, tramite l’uso di un wrapper, per configurare, addestrare, validare, importare, esportare, e deployare modelli di vulnerability prediction. Per cominciare, basterebbe offrire un'interfaccia da riga di comando. Nell’ambito di questo progetto, si possono integrare alcuni tool allo stato dell’arte, e.g., ReVeal, IVDetect, Funded, SySeVR, Devign… Il testing dovrà concentrarsi sul wrapper e sulle diverse parti della pipeline dei modelli, e.g., che il metodo di training faccia davvero il training! In alternativa all’interfaccia da riga di comando, si potrebbe esporre una programming API.

Repository Modelli: https://github.com/VulDetProject/ReVeal
https://github.com/vulnerabilitydetection/VulnerabilityDetectionResearch
https://github.com/HuantWang/FUNDED_NISL
https://github.com/SySeVR/SySeVR
https://github.com/epicosy/devign

PySZZ

Open-source implementation of several versions of SZZ for detecting bug-inducing commits

PySZZ è un tool scritto in Python che implementa diversi algoritmi basati su SZZ. SZZ è un algoritmo che permette il recupero di commit che probabilmente hanno originato un certo bug a partire dal suo fixing commit. PySZZ ha già a bordo diverse implementazioni note… ma non tutte!

Idea

Integrare due varianti di SZZ specializzate per il recupero dei Vulnerability Contributing Commit, ovvero: V0Finder e V-SZZ. Il primo è solo una collezione sparsa di script, mentre il secondo è già fornite come un'estensione di PySZZ ma di una sua versione precedente.

Repository PySZZ: https://github.com/grosa1/pyszz

Repository V0Finder: https://github.com/WOOSEUNGHOON/V0Finder-public

Repository V-SZZ: https://github.com/baolingfeng/V-SZZ/tree/main/ICSE2022ReplicationPackage

JDependency

Provides an API to analyse and modify class dependencies. It provides the core to the maven shade plugin for removing unused classes

JDependency è una piccola libreria che permette di analizzare un classpath (i.e., una collezione di JAR) per trovare classi inutilizzate, dipendenze transitive, ecc.

Idea

Aggiungere un nuovo modulo che permetta l'estrazione di tutto il grafo delle dipendenze di un'applicazione Java da esportare in diversi formati—testuale e grafico. Oltre ciò, bisogna anche fornire una piccola interfaccia da riga di comando per effettuare anche analisi su larga scala.

Repository: https://github.com/tcurdt/jdependency

IDEAL

An Open-Source Identifier Name Appraisal Tool

IDEAL è un tool scritto in Java che si occupa della detection dei cosiddetti "linguistic anti-pattern", ovvero cattive scelte negli identificatori di metodi e variabili che possono mandare in confusione lo sviluppatore. Un esempio di linguistic anti-pattern è un metodo "void isGreen()", il cui nome ci fa pensare che il valore di ritorno sia un booleano, e la cui signature ci lascia confusi. IDEAL rileva la presenza di linguistic anti-pattern in codice Java, e fornisce in output un csv con il nome della classe problematica, il tipo di antipattern rilevato, e il punto del codice che presenta il problema.

Idea 1

Il tool analizza codice Java. Il lavoro principale consiste nel permettere la detection di (uno o più tipi di) linguistic anti-pattern in codice Python.

Idea 2

Altri tipi di modifiche coerenti con altri esami (es. SE4AI, SwD). La modifica deve essere attivabile/disattivabile senza compromettere il funzionamento del tool.

Repository: https://github.com/SCANL/ProjectSunshine

Paper: https://ieeexplore.ieee.org/abstract/document/9609217

Readability Tool

A comprehensive model for code readability

La readability del codice sorgente è di fondamentale importanza per la manutenibilità del software. Scalabrino et al. hanno sviluppato un tool per il calcolo di otto metriche di readability.

Idea 1

Il tool è utilizzabile a riga di comando e fornisce l'output direttamente a riga di comando. Non è possibile specificare quali metriche calcolare, e non è possibile ottenere l'output formattato in un file customizzato. Le eccezioni vengono segnalate sullo stderr. Il progetto consiste nello sviluppo di un wrapper che renda il tool utilizzabile in maniera più efficiente.

Idea 2

Altri tipi di modifiche coerenti con altri esami (es. SE4AI, SwD). La modifica deve essere attivabile/disattivabile senza compromettere il funzionamento del tool.

Repository: https://dibt.unimol.it/report/readability/files/readability.zip

Paper: https://onlinelibrary.wiley.com/doi/abs/10.1002/smr.1958

A Conversational Agent for the Detection of Community Smells

CADOCS è un conversational agent (Chatbot) per la piattaforma Slack che si occupa di individuare pattern di comunicazione e collaborazione non ottimali (a.k.a. Community Smells) nelle comunità di sviluppo software su GitHub.

Idea 1

Separare la parte di comunicazione da quella di analisi facendo in modo da avere più moduli per ogni canale di comunicazione (e.g., Slack, Telegram, Discord).

Idea 2

Potenziare CADOCS aggiungendo nuovi moduli basati su ML per la detection di metriche sociali e risolvere i problemi di installazione, mantenendo i moduli come software separati per una attivazione facile.

Idea 3

Altri tipi di modifiche coerenti con altri esami (es. SE4AI, SwD).

Repository: https://github.com/gianwario/CADOCS

Paper: https://stefanolambiase.github.io/assets/papers/ICSME_CADOCS_2022.pdf

DEFUSE

The data annotator and model builder for software defect prediction

Defuse è un tool scritto in Python che raccoglie e classifica automaticamente software failure, consente la correzione di tali classificazioni e crea modelli di machine learning per rilevare i difetti sulla base di tali dati.

Idea 1

Suggerire potenziali refactoring per risolvere difetti

Idea 2

Integrazione con ambienti di CI/CD (attualmente solo GitHub Actions)

Idea 3

Analizzare altri linguaggi (oltre a Python, Ansible e TOSCA)

Repository: https://github.com/radon-h2020/radon-defuse

Paper: https://ieeexplore.ieee.org/document/9978237

DARTS

Detection And Refactoring of Test Smells

DARTS è un plugin per l'IDE IntelliJ capace di eseguire la detection di test smell nei progetti Java.

Idea 1

Aggiungere un modulo ML per la detection di test smell in DARTS e migliorare la documentazione utilizzando i lavori precedenti disponibili su GitHub. Si propone anche un meccanismo di "voto" per unire le precedenti detection, mantenendo la possibilità di attivare/disattivare la modifica senza compromettere il funzionamento del tool.

Idea 2

Integrare la detection di test smell per nuovi linguaggi in DARTS e migliorare la documentazione utilizzando i lavori precedenti disponibili su GitHub. La modifica deve essere attivabile/disattivabile senza compromettere il funzionamento del tool.

Idea 3

Altri tipi di modifiche coerenti con altri esami (es. SE4AI, SwD). La modifica deve essere attivabile/disattivabile senza compromettere il funzionamento del tool.

Repository: https://github.com/StefanoLambiase/DARTS

Paper:https://fabiano-pecorelli.github.io/publications/conferences/C4.pdf