OFFICINA GENAI
PROPOSTE ARGOMENTI

ACCENTURE

Table of Contents

1 Proposta di tesi 1

Titolo provvisorio
“Test automation agentica da prompt: progettazione e sperimentazione di un agente LLM per l’esecuzione automatica di test”

1.1 Contesto e motivazione

Molte aziende hanno già framework di test automation, ma:

- la scrittura e manutenzione delle suite è costosa e fragile;
- i tester non tecnici faticano a contribuire;
- i test non tengono il passo con il ritmo di rilascio.

L’emergere di **agentic AI** e di **LLM agents** permette di passare da script rigidi a **agenti autonomi** in grado di: capire requisiti in linguaggio naturale, pianificare passi di test, usare strumenti (framework di test, API, file di dati), registrare risultati e segnalare anomalie. (lilianweng.github.io)

1.2 Descrizione estesa del lavoro di tesi

L’obiettivo è progettare e implementare un **prototipo di agente di test** che:

1. riceve dall’utente un **prompt** con la descrizione del test (es. “Verifica che un utente registrato possa fare login e visualizzare il carrello vuoto”) e l’indicazione dei **dati** (inline o tramite file);
2. traduce il prompt in un **piano di test strutturato** (passi, precondizioni, dati, asserzioni);
3. **esegue automaticamente** il test tramite un framework esistente (es. Playwright / Cypress per web, Postman/Newman o pytest per API);
4. raccoglie **log, screenshot, output**, e produce un **report strutturato** con eventuali anomalie (failure tecniche, mismatch rispetto al requisito, step non eseguibili).

L’agente dovrà avere almeno:

- un modulo di **comprensione del prompt** (LLM + schema di test);
- un modulo di **planning/esecuzione** (agent loop che decide quali azioni compiere, quali strumenti richiamare, come reagire ai fallimenti); (promptingguide.ai)
- un modulo di **analisi esiti** (classificazione delle anomalie, sintesi per il tester).

Il lavoro di tesi (5 mesi) si concentra su una **versione MVP solida**, ma progettata per essere estendibile in una roadmap fino a 10 mesi (staffetta con un gruppo successivo).

1.3 Obiettivi principali (5 mesi di tesi)

- **O1 – Progettare l'architettura dell'agente di test** (componenti, flussi, interfacce con gli strumenti di test aziendali). (lilianweng.github.io)
- **O2 – Implementare un MVP funzionante** che, da prompt + dati, generi, esegua e registri almeno una classe di test (es. test end-to-end web o API).
- **O3 – Valutare il prototipo** su un set di casi reali/realistici, misurando copertura, effort di scripting risparmiato, qualità dei report.

1.4 Roadmap estesa (10 mesi, logica "staffetta")

Fase 1 (mesi 1–2) – Analisi e design

- Analisi dello **stato dell'arte su agentic testing, LLM agents, AI test automation**. (UiPath)
- Analisi del **contesto aziendale**: tipo di applicazioni, test esistenti, toolchain (CI/CD, test runner, repository).
- Definizione della **architettura logica** dell'agente e del **formato standard** per i test derivati dal prompt.

Fase 2 (mesi 3–5) – MVP (risultato tipico della prima tesi)

- Implementazione del **modulo di parsing del prompt** e mappatura verso uno schema interno di test (es. Gherkin-like o JSON strutturato). (Medium)
- Integrazione con almeno un **framework di test** (es. Playwright/API testing) per esecuzione end-to-end.
- Implementazione del **reporter** (salvataggio esiti, log, eventuali screenshot o output).

Fase 3 (mesi 6–8) – Estensioni (gruppo successivo)

- Miglioramento della **robustezza agentica**: gestione degli errori, tentativi di recovery, ripianificazione dei passi. (Testsigma Agentic Test Automation Tool)
- Supporto a nuove **tipologie di test** (es. regression suite, smoke test su più ambienti).
- Introduzione di un modulo di **prioritizzazione** automatica dei test o di suggerimento di nuovi casi.

Fase 4 (mesi 9–10) – Valutazione avanzata e industrializzazione

- Esperimento comparativo: agente vs. test automation tradizionale su un sottoinsieme di casi. (journalwjaets.com)

- Definizione di **metriche di qualità** (tasso di failure utili, falsi positivi/negativi, tempo di setup vs. approccio classico).
- Proposta di **linee guida aziendali** per l'adozione di un "Test Agent" in pipeline CI/CD.

1.5 Possibile suddivisione del lavoro tra 2–3 studenti (per i primi 5 mesi)

- **Studente A – Prompt understanding & design del modello di test**
    - Studio LLM/agents; progettazione schema dei casi di test; implementazione del parser da prompt a struttura di test.
- **Studente B – Execution & integrazione con i tool di test**
    - Integrazione con framework (Playwright/Cypress/pytest ecc.); gestione dati di test; orchestrazione dell'esecuzione.
- **Studente C (opzionale) – Reporting, analytics e UX di interazione**
    - Progettazione dell'interfaccia (CLI/web); reportistica; classificazione delle anomalie; piccola dashboard per QA.

1.6 Deliverable attesi (fine tesi – 5 mesi)

- **D1 – Documento di tesi** con analisi, design, dettagli implementativi, risultati sperimentali.
- **D2 – Repository del prototipo** (codice, configurazioni, casi di test di esempio, guida all'uso).
- **D3 – Report di valutazione** con confronto preliminare rispetto ai test attuali o a uno scenario baseline sintetico.

1.7 Tecnologie e stack suggeriti (esempi, adattabili all'azienda)

- **Linguaggi / runtime**: Python o TypeScript/Node.js per l'agente e l'integrazione con framework di test.
- **Framework di test**: Playwright / Cypress (web), pytest + requests / Postman-Newman (API).
- **LLM / agent framework**: API LLM (OpenAI, ecc.) + un semplice framework di orchestrazione agentica (es. definizione di loop di planning/tool-use ispirato alla letteratura su LLM agents). (lilianweng.github.io)

1.8 Bibliografia / riferimenti iniziali

- **Agentic testing e AI per il QA**

- BrowserStack, *Agentic AI in Testing* (articolo introduttivo su agentic AI nel testing, concetti e casi d'uso). (BrowserStack)
- UiPath, *What is Agentic Testing?* (definizione, ciclo di vita del testing agentico). (UiPath)
- XenonStack, *Agentic AI for Software Testing* (benefici, sfide e trend). (XenonStack)

- **LLM agents e architetture agentiche**
  - Lilian Weng, *LLM Powered Autonomous Agents* (blog tecnico sulle componenti chiave: planning, memory, tool use). (lilianweng.github.io)
  - ACM, *Demystifying LLM-Based Software Engineering Agents* (panoramica sul ruolo degli agenti LLM nello sviluppo e testing). (ACM Digital Library)
- **AI-driven test automation / stato dell'arte**
  - S. Arcadinho et al., *Automated test generation to evaluate tool-augmented LLM agents* (dataset e valutazione di agenti LLM in contesti di test). (ACL Anthology)
  - Articoli divulgativi su "How LLMs and AI Agents Are Transforming Test Automation" e "AI and machine learning driven test automation" per statistiche e contesto industriale. (Medium)

2 Proposta di tesi 2

Titolo provvisorio
“DevSecOps guidata da GenAI: progettazione di un copilota di sicurezza lungo la pipeline CI/CD”

(eventuale sottotitolo: “Uso di modelli generativi per automatizzare controlli, remediation e compliance security-by-design”)

2.1 Contesto e motivazione

- Le aziende adottano DevSecOps per integrare la sicurezza nella pipeline CI/CD, ma i team faticano a gestire il numero di tool, alert e policy. (arXiv)
- La GenAI può analizzare codice, dipendenze, IaC e log per suggerire fix, generare test, spiegare vulnerabilità e automatizzare controlli di conformità. (about.gitlab.com)
- Allo stesso tempo l’uso di GenAI introduce **nuovi rischi**: codice generato insicuro, data leakage, prompt injection, mancanza di tracciabilità. (Legit Security)

La tesi mira a progettare e sperimentare un **“GenAI SecOps Copilot”** che assista sviluppatori e security engineer dentro la pipeline esistente, con un approccio pragmatico e misurabile.

2.2 Descrizione estesa del lavoro di tesi

Il gruppo di tesi realizzerà un **prototipo integrato nella pipeline CI/CD** aziendale (o in un ambiente demo) che:

1. **Intercetta eventi chiave** della pipeline (push, merge request, build, deploy).
2. Interroga strumenti di sicurezza già presenti (es. SAST, SCA, container/IaC scanning) e raccoglie i risultati. (Snyk)
3. Usa GenAI per:
    - spiegare in linguaggio naturale le vulnerabilità;
    - suggerire patch o refactoring sicuri;
    - generare checklist e commenti automatici su merge request;
    - supportare la documentazione di compliance (es. mapping a policy interne o standard). (TechRxiv)
4. Integra **guardrail** specifici per GenAI:
    - limitazione e mascheramento dei dati sensibili;
    - policy per l’uso del modello;

- uso di strumenti di red-teaming/valutazione sicurezza LLM (es. garak o linee guida simili). (Wikipedia)

Il risultato dei 5 mesi deve essere un **MVP stabile**, facilmente estendibile nei mesi successivi (staffetta).

2.3 Obiettivi principali (nei 5 mesi di tesi)

- **O1 – Analisi e design**
  Modellare un'architettura DevSecOps + GenAI chiara: punti di integrazione nella pipeline, flussi dati, responsabilità (dev, sec, ops). (AWS Documentation)
- **O2 – Implementazione di un MVP funzionante**
  Un copilota che, su un progetto di riferimento, intercetti almeno:
  - 1 tool di SAST e 1 di SCA o container/IaC scanning;
  - 1 provider GenAI;
  - 1 pipeline CI/CD (GitLab/GitHub/altro).
- **O3 – Valutazione empirica**
  Misurare:
  - riduzione del tempo di triage delle vulnerabilità;
  - qualità delle proposte di remediation;
  - percezione di utilità da parte di dev/security (questionario).

2.4 Roadmap estesa a 10 mesi (logica "staffetta")

Fase 1 (mesi 1–2) – Analisi e modellazione

- Studio di **DevSecOps + GenAI**: casi d'uso, best practice, rischi (codice AI-generated, data leakage, model abuse). (about.gitlab.com)
- Analisi pipeline aziendale o demo (repository, CI/CD, strumenti sicurezza già adottati).
- Definizione dell'**architettura logica** del copilota (ingressi/uscite, API, formati JSON/YAML).

Fase 2 (mesi 3–5) – MVP (output tipico del primo gruppo di tesi)

- Integrazione con pipeline CI/CD (es. GitLab CI o GitHub Actions).
- Collegamento a tool di sicurezza (es. SAST + SCA) e normalizzazione dei risultati. (arXiv)

- Implementazione del modulo GenAI che:
    - riassume le vulnerabilità rilevate per MR/build;
    - propone fix di esempio;
    - genera un mini "security changelog" leggibile.
- Implementazione di controlli base di **sicurezza GenAI** (es. blocco di dati sensibili nei prompt).

Fine Fase 2 = traguardo minimo per la tesi di 5 mesi.

Fase 3 (mesi 6–8) – Estensioni (gruppo successivo)

- Aggiunta di altri use case DevSecOps:
    - suggerimento di test aggiuntivi per vulnerabilità critiche;
    - valutazione di misconfigurazioni su IaC o Kubernetes. (AWS Documentation)
- Introduzione di **policy-as-code** per controllare l'uso della GenAI (es. quali branch/ambienti possono usare il copilota, limiti per i dati).
- Integrazione con strumenti di **LLM red-teaming/vulnerability scanning** (es. garak o framework equivalenti). (Wikipedia)

Fase 4 (mesi 9–10) – Valutazione avanzata e industrializzazione

- Studio comparativo: pipeline con e senza GenAI-copilot su un insieme di MR/cambiamenti. (arXiv)
- Definizione di metriche avanzate (MTTR delle vuln, falsi positivi, adozione da parte dei dev).
- Proposta di **linee guida aziendali** per l'adozione sicura di GenAI in DevSecOps (policy, ruoli, training).

2.5 Possibile suddivisione del lavoro (2–3 studenti)

- **Studente A – Integrazione DevSecOps & pipeline**
    - Analisi pipeline;
    - configurazione CI/CD;
    - integrazione con SAST/SCA/container scan.
- **Studente B – Modulo GenAI & prompt engineering di sicurezza**
    - Definizione prompt e formati input/output;

    - generazione spiegazioni, fix e documentazione di compliance;
    - gestione dei guardrail per i prompt.
- **Studente C (opzionale) – Misurazione, UX e policy**
    - Raccolta metriche;
    - definizione dashboard/reportistica;
    - modellazione di policy e linee guida d'uso.

2.6 Deliverable attesi (a fine tesi – 5 mesi)

- **D1 – Documento di tesi**
    - Stato dell'arte, architettura, scelte progettuali, valutazione.
- **D2 – Repository del prototipo**
    - Codice (pipeline, integrazioni, moduli GenAI), README, esempi di progetto demo.
- **D3 – Report di valutazione**
    - Misure quantitative (tempo di triage, numero vuln trattate) + feedback qualitativo dei potenziali utenti.

2.7 Stack tecnologico suggerito (adattabile al contesto aziendale)

- **CI/CD**: GitLab CI, GitHub Actions, Azure DevOps. (about.gitlab.com)
- **Sicurezza** (esempi):
    - SAST: SonarQube, Semgrep, Snyk Code; (Snyk)
    - SCA: Snyk, OWASP Dependency-Check;
    - Container/IaC: Trivy, Checkov, ecc.
- **GenAI**: API LLM (OpenAI o equivalenti), eventualmente con un livello intermedio di policy (es. soluzioni tipo Prompt Security / piattaforme di controllo per GenAI). (Wikipedia)
- **LLM Security / Red-teaming** (fase avanzata): garak o tool simili. (Wikipedia)

2.8 Riferimenti bibliografici iniziali

- **Linee guida e casi d'uso DevSecOps + GenAI**

- AWS Prescriptive Guidance, *Generative AI use cases for DevSecOps*. (AWS Documentation)
- GitLab, *How to put generative AI to work in your DevSecOps environment*. (about.gitlab.com)
- Softweb Solutions, *How Generative AI transforms DevSecOps*. (Softweb Solutions)
- NextGenSoft, *GenAI in DevSecOps: Enhancing Security and Compliance*. (NextGenSoft)

- **Articoli/Paper accademici**
  - TechRxiv (2025), *Integrating Generative AI into DevSecOps: Principles, Tools, and Practices*. (TechRxiv)
  - Cheenepalli et al. (2025), *Advancing DevSecOps in SMEs: Challenges and Best Practices for Secure CI/CD Pipelines*. (arXiv)
  - Cui (2024), *The Enhancement of Software Delivery Performance through Enterprise DevSecOps and Generative Artificial Intelligence*. (arXiv)
  - Haryanto et al. (2024), *SecGenAI: Enhancing Security of Cloud-based Generative AI Applications*. (arXiv)
- **Rischi e sicurezza nell'uso della GenAI**
  - Black Duck, *Secure AI-generated code with DevSecOps best practices*. (Black Duck)
  - Legit Security, *AI DevSecOps: Updating Current Processes With New Technology*. (Legit Security)
  - TechRadar, *Second-order prompt injection can turn AI into a malicious insider* (esempio di rischio avanzato con agenti GenAI). (TechRadar)

3 Proposta di tesi 3

Titolo provvisorio
"Business Process Outsourcing agentico nel settore Utilities: progettazione di una piattaforma multi-agente per processi di front e back-office"

(eventuale sottotitolo: "Da BPO tradizionale a Agentic Process Automation per energia, gas, acqua e rifiuti")

3.1 Contesto e motivazione

- Le aziende **utilities** (energia, gas, acqua, waste) esternalizzano da anni customer care, billing, credit management, back-office documentale a BPO specializzati. (www.cognizant.com)
- L'AI sta trasformando il BPO: non solo "labour arbitrage", ma **operazioni digitali intelligenti** che automatizzano fino al 90% del back-office. (ardem.com)
- La **Agentic Process Automation** usa **AI agents** (LLM + strumenti + regole) per orchestrare processi end-to-end, superando i limiti di RPA rigida. (Automation Anywhere)
- Le utilities stanno già adottando AI e agenti virtuali per customer care e field operations: è il terreno ideale per sperimentare **architetture agentiche BPO-centriche**. (Baringa)

La tesi mira a definire e testare una **architettura di BPO agentico** specifica per utilities, su 1–2 processi reali (es. reclami fatturazione, volture/subentri, piani di rientro).

3.2 Descrizione estesa del lavoro di tesi

Il gruppo progetterà e realizzerà un **proof of concept** in cui un insieme di **agenti software** coordina attività oggi svolte da operatori BPO e sistemi legacy.

3.2.1 Processi target (esempi tipici utilities)

Si selezionano 1–2 processi, ad esempio:

- **UC1 – Gestione reclami di fatturazione**
    - Apertura ticket (voce/chat/mail), recupero contratto, verifica letture/consumi, eventuale storno, risposta al cliente.
- **UC2 – Voltura/Subentro**
    - Raccolta dati cliente, verifica morosità, controllo POD/PDR, generazione contratto, pianificazione data attivazione.

Entrambi si prestano a un mix di **task strutturati** (query su CRM/Billing, aggiornamento pratiche) e **task cognitivi** (interpretare il problema del cliente, scegliere next best action).

3.2.2 Architettura agentica di riferimento

La tesi definisce un'architettura logica composta da:

- **Orchestrator Agent**
    - Riceve eventi dal canale (nuovo ticket / pratica), consulta le regole di processo, assegna il lavoro ai sotto-agenti, decide escalation verso operatori umani. (IBM)
- **Channel / Interaction Agents**
    - Assistenti per operatore o cliente (chat/console interna): sintetizzano il contesto, propongono azioni, compilano automaticamente form e template di comunicazione. (dialonce.ai)
- **Process Agents** (per dominio)
    - Es. *BillingAgent*, *ContractAgent*, *CreditAgent*, *OutageAgent*: ciascuno incapsula logica di business, workflow e integrazioni per uno specifico sotto-processo.
- **Data & Knowledge Agents**
    - Agenti che dialogano con CRM, Billing, Document Management, Knowledge Base regolatoria/contrattuale; normalizzano i dati e forniscono "risposte pronte" agli altri agenti. (IBM)

Nel PoC, molte integrazioni potranno essere **mockate** o semplificate (API finte, dataset di test), mantenendo però fedele il modello concettuale.

3.3 Obiettivi della tesi (5 mesi)

- **O1 – Modellare i processi BPO utilities selezionati**
    - As-is / To-be, mappa di sistemi, attori, KPI (SLA, FCR, TTR…). (www.cognizant.com)
- **O2 – Progettare l'architettura agentica di BPO**
    - Classi di agenti, responsabilità, flussi di messaggi, interfacce verso sistemi e operatori.
- **O3 – Realizzare un MVP funzionante**
    - Orchestrator + 2–3 Process/Data Agents integrati con un canale (es. console web per operatore) su almeno un processo reale (UC1 o UC2).
- **O4 – Valutare il PoC**

- Su casi realistici: tempo medio di gestione, numero di passi automatizzati, carico residuo per gli operatori.

3.4 Roadmap estesa a 10 mesi (logica "staffetta")

Fase 1 (mesi 1–2) – Analisi e design (gruppo 1)

- Analisi BPO utilities: principali processi esternalizzati, contratti di servizio, KPI. (www.cognizant.com)
- Scelta dei **processi target** (es. reclami fatturazione + volture).
- Modellazione As-is e identificazione delle attività candidabili a automazione agentica vs RPA classica. (Automation Anywhere)
- Design architetturale di piattaforma: tipologie di agenti, canali, flussi.

Fase 2 (mesi 3–5) – MVP (risultato tipico del primo gruppo)

- Implementazione **Orchestrator Agent** + 2–3 Process/Data Agents per UC1.
- Integrazione con:
  - un data store simulato (CRM/Billing fittizio),
  - un'interfaccia per operatore (es. web app) che mostra suggerimenti degli agenti.
- Esecuzione di una **campagna di test** con scenari di ticket standard e complessi; raccolta di metriche di base.

Output atteso per la tesi di 5 mesi: PoC stabile su un processo, architettura documentata, primi numeri su benefici/limiti.

Fase 3 (mesi 6–8) – Estensioni funzionali (gruppo successivo)

- Aggiunta del secondo processo (UC2) o di un nuovo canale (es. voice bot simulato). (startek.com)
- Introduzione di logiche di **AgentOps** (monitoraggio, metriche per singolo agente, alert su colli di bottiglia). (McKinsey & Company)
- Sperimentazione di **politiche di routing intelligente** tra agenti e operatori umani (priorità clienti vulnerabili, reclami complessi). (Baringa)

Fase 4 (mesi 9–10) – Ottimizzazione e valutazione avanzata

- Modellazione del sistema come **multi-agent system** (MAS) e confronto con letteratura MAS in smart grid / resource allocation. (SmythOS)
- Simulazioni "what-if" (più volume di ticket, SLA più stringenti) per valutare scalabilità e robustezza.

- Raccolta requisiti per industrializzazione (integrazione con CRM reale, sicurezza, audit log).

3.5 Suddivisione del lavoro (2–3 studenti)

- **Studente A – Analisi processi & architettura di business**
    - Process mapping As-is/To-be, identificazione use case, requisiti funzionali e non funzionali, KPI.
- **Studente B – Architettura tecnica & agenti**
    - Design agenti, orchestrator, scelte tecnologiche, implementazione PoC, test tecnici.
- **Studente C (opzionale) – Data & Analytics / Valutazione**
    - Disegno dei dataset di scenario, raccolta metriche, simulazioni, confronto con approcci BPO tradizionali o solo-RPA. (ardem.com)

3.6 Deliverable attesi (a fine tesi – 5 mesi)

- **D1 – Elaborato di tesi**
    - Contesto BPO utilities, stato dell'arte su AI BPO/agentic automation, descrizione architettura, risultati sperimentali.
- **D2 – Codice del PoC**
    - Repository (agent framework, orchestrator, mock API, UI operatore, casi di test, istruzioni di deploy).
- **D3 – Report di valutazione**
    - KPI prima/dopo (anche solo stimati su scenari), mappa rischi/limiti, suggerimenti per fasi successive (staffetta).

3.7 Stack tecnologico suggerito (esempi)

(da adattare agli standard aziendali)

- **Linguaggi / runtime**: Python o TypeScript/Node.js per orchestrator e agenti.
- **Architettura agentica**:
    - servizi/"micro-agenti" che comunicano via coda messaggi (es. Kafka/RabbitMQ) o API interne;

    - uso di LLM per comprensione ticket/testi liberi + regole/workflow engine per la parte deterministica. (IBM)

- **Integrazioni demo**:
    - mock CRM/Billing (DB relazionale + API REST);
    - UI web per operatori BPO (dashboard colloquio operatore-agente).

3.8 Riferimenti bibliografici iniziali

BPO & AI nel settore utilities

- Cognizant, *Unlocking the potential of modern business process outsourcing in the utilities sector* (2024). (www.cognizant.com)
- Infosys BPM, *Enhancing utilities BPO with automation* (2024). (Infosys BPM)
- INGO, *Digitizing customer care in utilities: how BPO and AI improve SLA and customer satisfaction* (2025). (ingo.it)

AI-driven BPO / AI agents per BPO

- GoodCall, *AI Agents for BPO: Enhance Service & Cut Costs in 2025*. (goodcall.com)
- Conectys, *Navigating AI BPO: how automation is transforming outsourcing services* (2025). (Conectys)
- Parseq, *AI in business automation – Reimagining Business Processes with AI* (2025). (parseq.com)
- Intalio, *Business Process Outsourcing (BPO) Automation: Trends and Innovations*. (intalio.com)

Agentic automation e architetture agentiche

- IBM, *Orchestrating agentic AI for intelligent business operations* + *What is agentic automation?* (IBM)
- Automation Anywhere, *What is Agentic Process Automation?* (Automation Anywhere)
- UiPath, *What is Agentic AI?* (UiPath)
- McKinsey, *One year of agentic AI: six lessons from the people doing the work* (2025). (McKinsey & Company)

Multi-Agent Systems (per la parte più "accademica")

- Woltmann et al., *Development and implementation of multi-agent systems in smart grids* (2022). (ScienceDirect)

- Izmirlioglu, *A Survey of Multi-Agent Systems for Smartgrids* (2024). (MDPI)
- Mahela et al., *Comprehensive Overview of Multi-agent Systems for Smartgrids* (2022). (SciOpen)

4 Proposta di tesi 4

Titolo provvisorio
“Agentic Workplace Assistant: architettura multi-agente per i servizi di supporto aziendali (IT, HR, Facilities)”

(eventuale sottotitolo: “Dalla FAQ al ticket: un unico assistente agentico per stanze, richieste, HR e helpdesk”)

4.1 Contesto e motivazione

- Nelle aziende moderne i servizi di supporto (IT, HR, facility, prenotazioni, ticket) sono frammentati tra portali, mail, numeri telefonici, form diversi: l’esperienza dell’utente interno è spesso lenta e disomogenea. (servicely.ai)
- Assistenti AI e chatbot gestiscono già fino all’80% delle richieste standard (FAQ, prenotazioni, piccole richieste operative), riducendo tempi e costi di supporto. (signinapp.com)
- Le nuove piattaforme **agentic AI** e **multi-agent systems** permettono di andare oltre il semplice chatbot: più agenti collaborano per gestire ticket, prenotare stanze, leggere policy HR, aggiornare sistemi, ecc. (Automation Anywhere)

La tesi punta a progettare e sperimentare una **piattaforma agentica di “Enterprise Support”** che offra un unico punto di accesso (chat) per: prenotare sale, fare domande HR, aprire richieste IT e gestire ticket.

4.2 Descrizione estesa del lavoro di tesi

Il gruppo realizzerà un **proof of concept** in cui un “Enterprise Support Agent” coordina più sotto-agenti per erogare servizi di supporto interni.

4.2.1 Casi d’uso coperti (esempi minimi)

1. **Prenotazione stanze / spazi**
    - L’utente chiede: “Prenotami una sala per 6 persone domani dalle 15 alle 16 con videoconferenza”.
    - L’agente verifica disponibilità e prenota, proponendo alternative se necessario. (yarooms.com)
2. **Domande HR e policy interne**
    - Es. “Quanti giorni di ferie mi restano?”, “Come funziona lo smart working?”.
    - L’agente cerca nelle policy HR, sintetizza la risposta, se serve apre un ticket HR o avvia un workflow di approvazione. (ebi.ai)

3. **Richieste / ticket IT & servizi generali**
    - Es. “Non funziona la VPN”, “La stampante del 3° piano è bloccata”.
    - L’agente classifica, prioritizza e indirizza la richiesta, aggiornando il sistema di ticketing e proponendo soluzioni rapide. (Rezolve)

4.2.2 Architettura agentica di riferimento

Architettura logica (in gran parte simulabile in lab):

- **Entry / Concierge Agent**
    - Punto di contatto unico (chat/web). Capisce l’intento (HR, IT, stanze, altro), raccoglie il contesto, instrada verso gli altri agenti. (servicely.ai)
- **Domain Agents**
    - *RoomBookingAgent*: dialoga con il sistema di prenotazioni (reale o mock). (yarooms.com)
    - *HRSupportAgent*: consulta knowledge base HR / policy. (siit.io)
    - *ITHelpdeskAgent*: interagisce con sistema ITSM / ticketing (o un suo mock), propone soluzioni standard o escalation. (Rezolve)
- **Knowledge & Integration Layer**
    - Motore di ricerca interna (documenti, FAQ, policy) + connettori (API) ai sistemi aziendali o a versioni demo/mock. (servicely.ai)

Il PoC potrà usare sistemi mock o semplificati ma **mantenendo chiaro il modello enterprise**.

4.3 Obiettivi principali (nei 5 mesi di tesi)

- **O1 – Modellare lo “Enterprise Support” aziendale**
    - Mappare i principali servizi (IT, HR, stanze, ticket), gli attori, i canali attuali, i KPI (tempo risposta, first-contact resolution, soddisfazione). (servicely.ai)
- **O2 – Progettare l’architettura multi-agente**
    - Definire ruoli degli agenti, flussi di messaggi, formati dati, logiche di routing e fallback verso operatori umani. (Automation Anywhere)
- **O3 – Realizzare un MVP funzionante**
    - Concierge + almeno due Domain Agents (es. stanze + IT) integrati con una UI (chat web) e sistemi mock; supporto per almeno 10–15 scenari tipici.

- **O4 – Valutare il PoC**
    - Misurare tempi medi, percentuale di richieste gestite end-to-end dall'agente, qualità percepita (questionario interno).

4.4 Roadmap estesa a 10 mesi (staffetta tra gruppi)

Fase 1 (mesi 1–2) – Analisi e design

- Analisi dell'offerta esistente (Rezolve.ai, Atomicwork, Modo, ecc.) per employee support, ITSM e HR agentici. (Rezolve)
- Mappatura dei processi di supporto interni (As-is) e definizione del modello To-be agentico.
- Design dell'architettura logica (tipi di agenti, canali, integrazioni).

Fase 2 (mesi 3–5) – MVP (risultato tipico della tesi da 5 mesi)

- Implementazione di:
    - **Concierge Agent** (intent classification + routing),
    - *RoomBookingAgent* (interfaccia a un semplice sistema di prenotazioni),
    - *ITHelpdeskAgent* (interazione con un sistema di ticketing o DB simulato).
- Implementazione di una **chat web** o integrazione con Teams/Slack demo. (Microsoft Marketplace)
- Test su scenari reali/realistici e raccolta metriche base.

Fine Fase 2 = obiettivo minimo per il primo gruppo di tesi.

Fase 3 (mesi 6–8) – Estensioni funzionali (gruppo successivo)

- Aggiunta di **HRSupportAgent** con accesso a policy/FAQ HR, onboarding, ferie, permessi. (ebi.ai)
- Introduzione di funzionalità di **triage e auto-risoluzione ticket** (suggerimento soluzione, chiusura automatica di richieste banali). (servicedeskshow.com)
- Logging avanzato per "AgentOps" (metriche per agente, errori, tempi di completamento). (Galileo AI)

Fase 4 (mesi 9–10) – Ottimizzazione e industrializzazione

- Simulazioni con carichi diversi (numero richieste, mix IT/HR/stanze) per valutare la scalabilità del sistema multi-agente. (oneadvanced.com)

- Definizione di **linee guida aziendali** per l'adozione di agenti di supporto (governance, sicurezza, privacy dati dei dipendenti). (BCG Global)
- Analisi di possibili integrazioni con piattaforme agentiche enterprise (es. Salesforce Agentforce, OutSystems Agent Workbench, ecc.). (TechRadar)

4.5 Suddivisione del lavoro (2–3 studenti)

- **Studente A – Process & UX**
  - Analisi processi di supporto;
  - definizione scenari utente e requisiti;
  - design della UX conversazionale (prompt, flussi, messaggi).
- **Studente B – Architettura tecnica & agenti**
  - Progettazione e implementazione degli agenti (Concierge + Domain);
  - orchestrazione, routing, integrazione con sistemi mock.
- **Studente C (opzionale) – Knowledge & Analytics**
  - Modellazione della knowledge base (FAQ, policy, dati stanze);
  - raccolta e analisi metriche;
  - valutazione comparativa con scenari "manuali".

4.6 Deliverable attesi (fine tesi – 5 mesi)

- **D1 – Elaborato di tesi**
  - Contesto, stato dell'arte, architettura, descrizione del PoC, risultati sperimentali.
- **D2 – Codice del PoC**
  - Repository con agenti, UI, mock systems, script di deploy e dataset di test.
- **D3 – Report di valutazione**
  - KPI (tempi, % richieste gestite in autonomia, soddisfazione), rischi, evoluzioni possibili per la staffetta successiva.

4.7 Stack tecnologico suggerito (adattabile)

- **Linguaggi / runtime**:

    - Python o TypeScript/Node.js per agenti e integrazioni.

- **Componenti chiave**:
    - Framework per orchestrazione agentica (o architettura “home-made” con servizi e code messaggi); (Automation Anywhere)
    - LLM per comprensione richieste, generazione risposte, riassunti;
    - motore di ricerca semantica per knowledge base interna. (servicely.ai)
- **Integrazioni demo**:
    - piccolo sistema di room booking (DB + API);
    - sistema ITSM / ticketing simulato;
    - sorgenti HR/FAQ (file, wiki, ecc.).

4.8 Riferimenti bibliografici iniziali

Enterprise Service Management & Agentic AI

- Servicely, *Agentic AI Use Cases in Enterprise Service Management* (ITSM, HR, ESM). (servicely.ai)
- Rezolve.ai, *Agentic AI for ITSM, HR and Shared Services Support*. (Rezolve)
- Atomicwork, *Agentic service management platform*. (Atomicwork)
- Modo Labs, *AI Knowledge Agent for the Modern Workforce*. (Modo Labs)

AI workplace assistant, room booking, scheduling

- Yarooms, *Yarvis – the AI workplace assistant* (room booking e coordinamento attività). (yarooms.com)
- Kadence, *Workplace AI for team scheduling and space booking*. (Kadence)
- SigninApp, *How AI transforms administrative tasks in the modern workplace*. (signinapp.com)

HR & employee support con AI

- EBI.ai, *4 tasks an AI assistant can do for your HR and office management teams*. (ebi.ai)
- Siit.io, *Agentic AI in HR: Automating Employee Support Workflows*. (siit.io)
- AcademyOcean, *Building an AI assistant for internal communications and FAQs*. (academyocean.com)

Multi-agent systems e agentic enterprise

- Automation Anywhere, *Multi-Agent Systems: Building the Autonomous Enterprise*. (Automation Anywhere)
- EdgeVerve, *Multi-Agent AI Systems: A Strategic Framework for Enterprise AI Innovation*. (edgeverve.com)
- Galileo, *Transform Enterprise AI with Multi-Agent Systems*. (Galileo AI)
- BCG, *How Agentic AI is Transforming Enterprise Platforms*. (BCG Global)

5 Proposta di tesi 5

Progettazione e sperimentazione di un agente LLM per il testing randomizzato e la navigazione completa di applicazioni tramite teoria dei grafi".

5.1 Contesto e motivazione

Il Monkey Testing è una tecnica che prevede l'esecuzione di azioni casuali sull'applicazione per individuare crash, errori o comportamenti inattesi. Tuttavia, un approccio completamente randomico rischia da un lato di non esplorare in modo adeguato tutte le possibili interazioni e i percorsi dell'applicazione, lasciando alcune aree non testate, e dall'altro di generare azioni talmente casuali da ridurre l'efficacia stessa del test.

L'integrazione di agenti LLM e l'applicazione della teoria dei grafi permettono di rendere il monkey testing più efficiente. L'agente costruisce una rappresentazione grafica dell'applicazione — in cui i nodi rappresentano schermate o stati e gli archi le possibili azioni — e guida la navigazione in modo controllato, riducendo la casualità. In questo modo massimizza la copertura dei percorsi, evita la ripetizione di azioni già esplorate e identifica in modo sistematico le aree dell'applicazione ancora non visitate.

5.2 Descrizione estesa del lavoro di tesi

La tesi mira a progettare e implementare un prototipo di agente di monkey testing che:

- Genera sequenze di azioni casuali e semi-casuali (evitando che la casualità generi test ridondanti) su applicazioni web/mobile/API.
- Estrae informazioni utili a guidare l'esecuzione dei test, inoltre tramite la teoria dei grafi, costruisce e aggiorna dinamicamente la mappa dell'applicazione.
- Applica algoritmi di ricerca su grafi (es. BFS, DFS, copertura massima) per navigare l'applicazione in modo sistematico e intelligente (es.: evitare di visitare i rami più volte).
- Raccoglie log, screenshot, output e segnala crash/anomalie.
- Produce report strutturati con metriche di robustezza, copertura, costo computazionale, e aree non esplorate.

5.3 Obiettivi principali (5 mesi di tesi)

- **O1** – Progettare l'architettura dell'agente monkey, con modulo di rappresentazione grafica e navigazione intelligente.
- **O2** – Implementare un MVP funzionante che esegua monkey testing su almeno una classe di applicazioni (web o mobile), costruendo e aggiornando il grafo di navigazione.
- **O3** – Valutare il prototipo su casi reali, misurando copertura, numero di crash rilevati, aree non esplorate e confronto con monkey testing tradizionale.

5.4 Roadmap estesa (10 mesi, logica “staffetta”)

- **Fase 1** (mesi 1–2): Analisi stato dell’arte su monkey testing, agentic AI, teoria dei grafi applicata al testing.
- **Fase 2** (mesi 3–5): Implementazione MVP, integrazione con framework di test (es. Cypress, Selenium, Playwright), costruzione dinamica del grafo.
- **Fase 3** (mesi 6–8): Estensione a nuove tipologie di test (API, mobile), ottimizzazione degli algoritmi di navigazione e copertura.
- **Fase 4** (mesi 9–10): Valutazione comparativa con monkey testing classico, definizione metriche di robustezza, costo computazionale e copertura, proposta di linee guida aziendali.

5.5 Possibile suddivisione del lavoro tra 2-3 studenti (primi 5 mesi)

- **Studente A**: Progettazione e implementazione modulo di randomizzazione guidata da LLM e costruzione del grafo.
- **Studente B**: Integrazione con framework di test, raccolta e analisi dei risultati, implementazione degli algoritmi di navigazione.
- **Studente C** (opzionale): Reporting, dashboard, analisi delle metriche di robustezza e copertura.

5.6 Deliverable attesi (fine tesi – 5 mesi)

- **D1** – Documento di tesi con analisi, design, dettagli implementativi, risultati sperimentali.
- **D2** – Repository del prototipo (codice, configurazioni, casi di test di esempio, guida all’uso).
- **D3** – Report di valutazione con confronto rispetto al monkey testing tradizionale e analisi della copertura tramite grafo.

5.7 Tecnologie e stack suggeriti (adattabili all’azienda)

- **Linguaggi**: Python o TypeScript/Node.js.
- **Framework di test**: Cypress, Selenium, Playwright
- **LLM**: API OpenAI o equivalenti, framework agentici per orchestrazione delle azioni.
- **Algoritmi di grafi**: librerie come NetworkX (Python) o Graphlib (Node.js).
- **Dashboard/reportistica**: Streamlit, Dash, Grafana.

5.8 Bibliografia / riferimenti iniziali

GUI Component Detection-Based Automated Software Crash Diagnosis
Seong-Guk Nam -Yeong-Seok Seo
Presenta una tecnica di testing GUI “visiva” (senza necessità di sorgente) per generare test e diagnosticare crash, migliorando la copertura rispetto a monkey testing puro. Utile se la tua tesi guarda a test GUI “black-box” o webapp dove il codice non è sempre “trasparente”.
https://www.mdpi.com/2079-9292/12/11/2382

A comparative analysis of web application test automation tools
Beata Pańczyk
confronta strumenti di test automation – tra cui Playwright – su vari parametri: velocità, efficienza, stabilità e flessibilità.
https://www.researchgate.net/publication/393213361_A_comparative_analysis_of_web_application_test_automation_tools

Artificial Intelligence helps making Quality Assurance processes leaner
Alexander Poth, Quirin Beck, Andreas Riel
Propone un approccio ibrido che combina monkey testing e “difference testing”, con evoluzione tramite algoritmo genetico per ottimizzare i test GUI in termini di coverage. Interessante perché affronta anche il tema del “test oracles” e automatizza regressione su GUI.
https://www.bohrium.com/paper-details/artificial-intelligence-helps-making-quality-assurance-processes-leaner/867753844619083985-108627

Improving random GUI testing with image-based widget detection
White, T., Fraser, G. and Brown, G.
Nell’ambito del test random, propone un approccio al riconoscimento delle widget basate su immagini.
https://eprints.whiterose.ac.uk/id/eprint/145601/

On the effectiveness of random testing for Android: Or how i learned to stop worrying and love the monkey
Priyam Patel, Gokul Srinivasan, Sydur Rahaman, Iulian Neamtiu
Studia con rigore quanto sia efficace il testing randomico su applicazioni mobile reali mostrando metodo ed esempi pratici applicati al mondo Android.
https://digitalcommons.njit.edu/fac_pubs/8654/

Uncertainty-Driven Black-Box Test Data Generation
Neil Walkinshaw, Gordon Fraser
Esplora l'ambito del test generation guidato dall'incertezza, per migliorare l'efficacia rispetto al random testing tradizionale
https://arxiv.org/abs/1608.03181