

UNIVERSITÀ DEGLI STUDI DI SALERNO
DIPARTIMENTO DI INFORMATICA

Ingegneria del Software Tecniche Avanzate

UniClass - Portfolio Analysis

Team

Lucageneroso Cammarota (Matricola: NF22500053)
Giuseppe Sabetta (Matricola: NF22500155)



ANNO ACCADEMICO 2025/2026

1 Analisi di Portfolio e Valutazione del Reengineering del Modulo Utenti

1.1 Inquadramento metodologico

L'analisi di portfolio del Modulo Utenti è stata condotta con l'obiettivo di valutare in maniera quantitativa il livello di debito tecnico accumulato e di supportare, attraverso un processo decisionale formalizzato, la scelta di un intervento di reengineering architetturale.

La metodologia adottata si ispira ai principi di Software Maintenance e Engineering Decision Making descritti nel Software Engineering Body of Knowledge (SWEBOK), con particolare riferimento alle aree di conoscenza relative alla manutenibilità, alla misurazione del software e all'analisi economica delle decisioni ingegneristiche.

L'analisi è stata formalizzata ex-post al fine di validare la razionalità tecnica della decisione già implementata; tuttavia, tutte le metriche utilizzate erano disponibili ex-ante e pertanto l'intero processo decisionale sarebbe stato applicabile prima dell'intervento.

1.2 Raccolta delle metriche di baseline

La fase iniziale ha previsto la raccolta sistematica delle metriche statiche di manutenibilità e complessità, ottenute tramite analisi automatizzata del codice sorgente.

Metrica	Valore Baseline
Linee di codice (sistema)	7.8 k
Numero di McCabe	582
Complessità ciclomatica modulo utenti	210
Complessità controller utenti	53
Complessità model utenti	50
Complessità service utenti	107
Cognitive Complexity modulo utenti	97
Violazioni architetturali (XMC)	5
Debt Ratio (Sonar)	0.9
Remediation Effort stimato	4 giorni 4 ore (36h)

Table 1: Metriche di baseline del Modulo Utenti

L'elevato valore di Debt Ratio (0.9) e la stima di remediation effort pari a 36 ore evidenziano un significativo accumulo di debito tecnico, con potenziale impatto negativo sulla manutenibilità e sulla velocità evolutiva del sistema.

1.3 Formalizzazione della funzione decisionale

Al fine di oggettivare la classificazione del modulo all'interno del portfolio, è stata definita una funzione di aggregazione pesata delle metriche rilevanti.

Sia definito il Technical Debt Score come:

$$DebtScore = w_1 \cdot CC_n + w_2 \cdot CogC_n + w_3 \cdot XMC_n + w_4 \cdot DebtRatio_n$$

dove le metriche sono state normalizzate come segue:

$$CC_n = \frac{CC_{mod}}{CC_{sys}} = \frac{210}{582} = 0.36$$

$$CogC_n = \frac{97}{120} = 0.81$$

$$XMC_n = \frac{5}{5} = 1$$

$$DebtRatio_n = 0.9$$

I pesi sono stati assegnati privilegiando le dimensioni maggiormente correlate alla manutenibilità strutturale:

$$w_1 = 0.30, \quad w_2 = 0.25, \quad w_3 = 0.20, \quad w_4 = 0.25$$

Ne deriva:

$$DebtScore_{pre} = 0.73$$

Poiché il valore supera la soglia critica di 0.6, il modulo è classificabile come componente ad alto debito tecnico.

1.4 Posizionamento nel Portfolio

Il posizionamento nel quadrante High Value – High Debt evidenzia la natura strategica del modulo, ma al contempo la sua criticità manutentiva. Tale combinazione giustifica la valutazione di un intervento strutturale piuttosto che un approccio puramente correttivo.

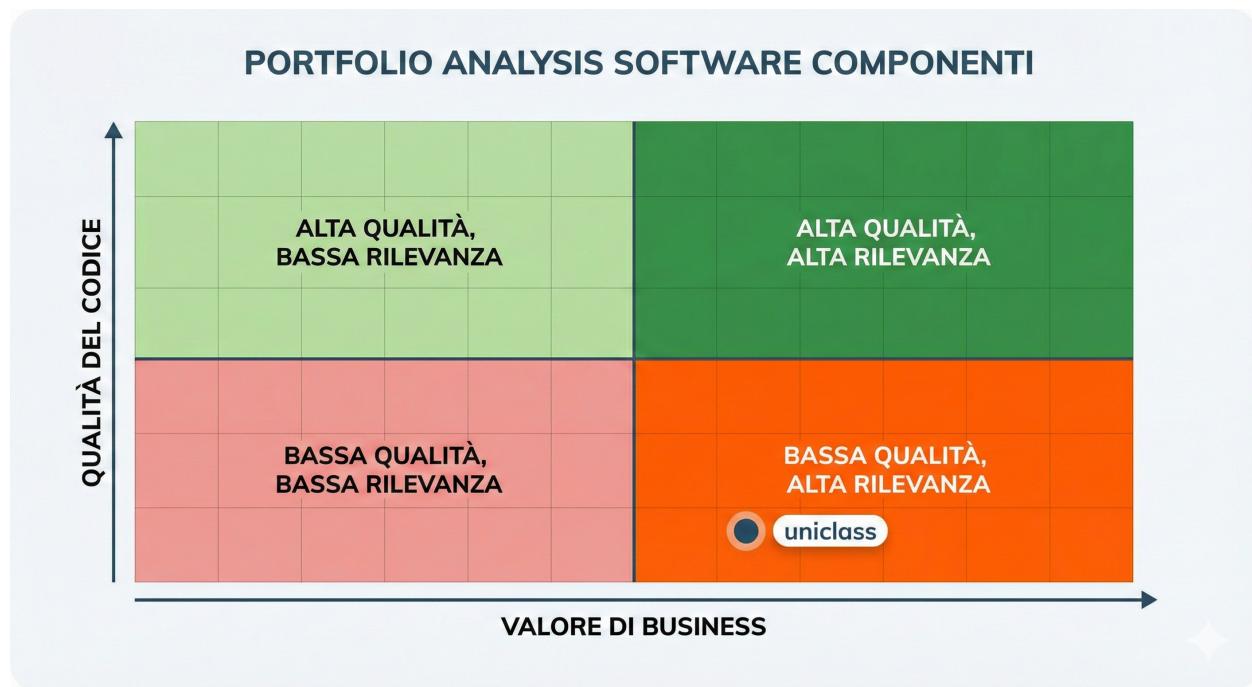


Figure 1: Portfolio Analysis Modulo Utenti

1.5 Analisi delle alternative

Il processo decisionale ha considerato quattro alternative tecniche.

A1 – Manutenzione incrementale correttiva Questa opzione avrebbe previsto la prosecuzione della manutenzione ordinaria senza interventi strutturali, limitandosi alla correzione dei difetti e all'implementazione di nuove funzionalità secondo l'architettura esistente. Tale approccio presenta un costo iniziale minimo ma non riduce il debito tecnico.

A2 – Refactoring locale L'intervento si sarebbe concentrato su porzioni critiche del codice (ad esempio metodi con elevata complessità ciclomatica), senza modificare l'architettura complessiva del modulo. Questa alternativa avrebbe prodotto benefici parziali ma non risolto le violazioni architetturali (XMC).

A3 – Riscrittura completa La completa reimplementazione del modulo avrebbe eliminato il debito tecnico, ma con elevato costo iniziale e significativo rischio operativo, inclusa la perdita di conoscenza implicita nel codice esistente.

A4 – Reengineering architetturale (scelta) L'alternativa selezionata ha previsto la riorganizzazione strutturale delle responsabilità, la riduzione dell'accoppiamento, l'introduzione di pattern

di mediazione e una redistribuzione delle logiche tra controller, service e model. Tale approccio consente una riduzione sostanziale del debito tecnico preservando la continuità funzionale.

Il reengineering architetturale massimizza il valore complessivo.

1.6 Risultati post-intervento

Dopo il reengineering, le metriche risultano:

Metrica	Valore Post
Linee di codice (sistema)	6.7 k
Numero di McCabe	534
Complessità ciclomatica modulo utenti	121
Complessità service utenti	48
Cognitive Complexity modulo utenti	69
Violazioni architetturali (XMC)	3

Table 2: Metriche post-reengineering

Il nuovo DebtScore risulta pari a:

$$DebtScore_{post} = 0.48$$

La riduzione complessiva del debito tecnico è pertanto pari a circa il 34%, mentre la complessità ciclomatica del modulo si riduce del 42%.

1.7 Analisi economica

Lo studio e il processo di manutenzione completato e descritto nei Deliverables citati sono fondamentali per la comprensione di questa sezione.

Dati i Performance Testing, si sono stabiliti condizioni importanti del software post-refactoring, ovvero una baseline prestazionale sistematica con un maggiore impatto sul soddisfacimento dei goal di Software Dependability, fondamentali per una piattaforma accademica (per gli accademici), risolvendo problemi di resilienza e affidabilità, inaccettabili e presenti nella versione precedente del Software.

La nuova release non avrà semplicemente migliori performance, bensì anche una licenza che aiuterà gli sviluppatori e mettere piede nel settore FOSS (Free Open Source Software), tra cui gli studenti del DI (Dipartimento d'Informatica d'Eccellenza dell'Università degli Studi di Salerno) che saranno i primi clienti e manutentori del software preso in considerazione, migliorando la sponsorizzazione e il Feedback Loop di UniClass.

Si considera quindi una prima messa in produzione con analisi della security della piattaforma

ed allineamento al GDPR e ISO 27001 Annex A, per poi introdurre il rilascio del software con interventi di sponsorship alle sole classi di Ingegneria del Software, Ingegneria del Software Tecniche Avanzate e Sicurezza in giornate differenti, potendo gestire sia la totalità delle classi che la loro dispersione nei giorni di sponsorizzazione agli studenti che vorranno partecipare ai processi di manutenzione per mettere mano ad un FOSS con studio In-Depth.

Dopo la divulgazione e il Feedback Loop garantito dagli studenti volenterosi, sarà possibile migliorare la versatilità e velocità dei successivi processi di manutenzione correttiva e perfettiva.

In caso di aumento spropositato degli utenti, sarà necessario utilizzare Kubernetes (oltre a Docker Swarm) e server in locale (per garantire sicurezza e accessibilità), per soddisfare i goal di Software Dependability imposti dalla prima CR di UniClass.

1.8 Conclusioni

L'analisi quantitativa dimostra che il Modulo Utenti presentava un livello critico di debito tecnico, con impatti significativi sulla manutenibilità e sulla sostenibilità evolutiva del sistema.

L'intervento di reengineering architetturale ha prodotto una riduzione sostanziale della complessità strutturale, del debito tecnico stimato e dei costi manutentivi prospettici.

La decisione risulta pertanto giustificata sia dal punto di vista tecnico sia sotto il profilo economico, configurandosi come intervento coerente con le buone pratiche di ingegneria del software.