Appunti di processo e sviluppo del software

Daniele Besozzi

Anno accademico 2025/2026

Contents

Premesse

Questi sono appunti realizzati per riassumere e schematizzare tutti i concetti presentati durante il corso di processo e sviluppo del software tenuto presso il corso di laurea magistrale in informatica presso l'università degli studi di Milano Bicocca. Lo scopo di questo documento non è quello di sostituire le lezioni del corso o di essere l'unica fonte di studio, bensì integrare le altri fonti con un documento riassuntivo.
Mi scuso in anticipo per eventuali errori e prego i lettori di segnalarli contattandomi via mail all'indirizzo d.besozzi@campus.unimib.it.

Chapter 1

Requirements Engineering

Per essere sicuri che una soluzione software risolva correttamente un problema del mondo reale dobbiamo prima comprenderlo completamente e definire:

- Quale problema sia da risolvere
- Il contesto da cui il problema parte

Definiamo come **mondo** la parte problematica del mondo reale, composto da componenti umani e componenti fisici. Chiamiamo invece **macchina** (o sistema) ciò che deve essere installato per risolvere il problema, cioè software o soluzione hardware-software. Il requirements engineering (RE) riguarda gli effetti della macchina sul mondo reale, le assunzioni e proprietà rilevanti del mondo.
Il **system as-is** è il sistema allo stato attuale, precedente l'installazione della macchina. Il **system to-be** è il sistema futuro, come sarà una volta installata la macchina.
In una definizione preliminare di RE possiamo dire che è un insieme di attività volto a esplorare, valutare, documentare, consolidare, ripassare e adattare gli obiettivi, capacità, qualità, requisiti e assunzioni di un sistema software-intensive. È basato sui problemi che sorgono nel sistema as-is e le opportunità portate dalle nuove tecnologie. Le difficoltà principali del RE sono:

- Scope ampio: diverse versioni del software (as-is, to-be, to-be-next) e ambienti ibridi (organizzazioni, leggi, politiche, device)
- Considerazioni multiple: funzionali, di qualità, di sviluppo.
- Livelli di astrazione
- Stakeholder multipli: con background diversi, interessi diversi e punti di vista contrastanti.
- Task intersecate durante il processo iterativo di elicitazione, valutazione, specifica e consolidazione.

Tre domande fondamentali che dobbiamo porci sono: perché installare un nuovo sistema? Quali servizi? Chi è responsabile per cosa?

- **Perché?** Identificare, analizzare, rifinire gli obiettivi del sistema to-be per risolvere problematiche rilevate nel sistema as-is, allinearsi con gli obiettivi business e sfruttare nuove tecnologie.
  Le difficoltà principali sono:
  - Acquisire conoscenze del dominio
  - Valutare le varie alternative
  - Identificare e risolvere conflitti tra obiettivi
- **Cosa?** Identificare, definire i servizi funzionali del sistema to-be
  - Soddisfare gli obiettivi identificati
  - In concomitanza con i requisiti di qualità: prestazioni, sicurezza…
  - Basarsi su assunzioni realistiche dell'ambiente.

  Le difficoltà principali sono:

  - Identificare le feature corrette.
  - Specificarle in maniera precisa in maniera che tutti possano capire.
  - Assicurare la tracciabilità tra obiettivi.

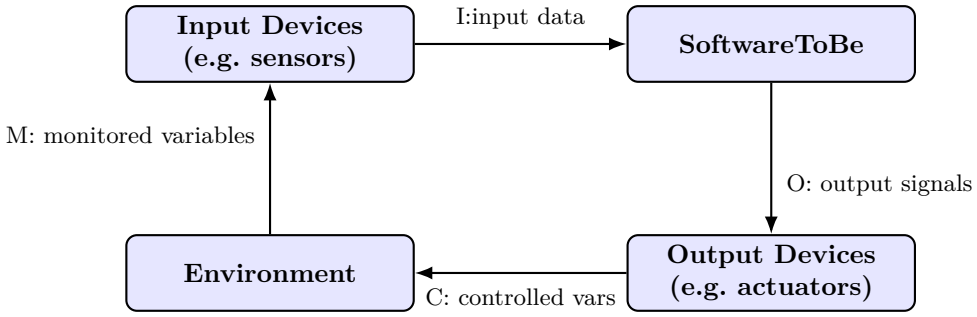
- **Chi?** Assegnare le responsabilità per gli obiettivi, servizi requisiti tra i componenti del sistema to-be.
  - Basandosi sulle capacità e obiettivi del sistema.
  - Definire il limite tra software e ambiente.

  Le difficoltà principali sono:

  - Valutare soluzioni alternative per decidere il giusto livello di automazione.

1.1 Tipi di requisiti

Il modo in cui i requisiti vengono espressi possono essere divisi in **descrittivi** (modo indicativo) e **prescrittivo** (modo ottativo).



1.1.1 Requisiti di sistema

Sistemi prescrittivi che si riferiscono a fenomeni dell'ambiente (non necessariamente condivisi). Vengono soddisfatti dal sistema to-be, possibilmente integrato con altri componenti del sistema. Devono essere comprensibili da tutti gli stakeholder. $SysReq \subseteq M \times C$ relazione tra ambiente monitorato e variabili controllate.

1.1.2 Requisiti software

Affermazioni prescrittive che si riferiscono a fenomeni condivisi tra ambiente e software. Vengono soddisfatti unicamente dal software, formulati nel vocabolario degli sviluppatori. $SofReq \subseteq I \times O$ relazione tra input e output del software.

1.1.3 Proprietà di dominio

Affermazioni descrittive riguardanti i fenomeni del mondo (rimangono veri a prescindere del sistema to-be). $Dom \subseteq M \times C$ leggi che non possono essere violate.

1.1.4 Assunzioni

Affermazioni che l'ambiente del software-to-be deve soddisfare. Formulate in termini di fenomeni di ambiente. Generalmente prescrittive (e.g. sensori e attuatori). $Asm \subseteq M \times C \cup M \times I \cup C \times O$

1.1.5 Definizioni

Affermazioni che forniscono un preciso significato ai concetti di sistemi e termini ausiliari. Non hanno valore di verità, non ha senso contestarle. $sofReq = Map(SysReq, Asm, Dom)$

I requisiti si dividono ulteriormente in:

- **Funzionali**: descrivono quali servizi il sistema deve fornire.
- **Non funzionali**: descrivono come il sistema deve essere (e.g. prestazioni, usabilità, affidabilità…).

1.2 Qualità del software

- Completezza di obiettivi, requisiti e assunzioni
- Coerenza degli elementi del documento dei requisiti (RD)
- Adeguatezza di requisiti, assunzioni e proprietà del dominio
- Non ambiguità degli elementi del documento dei requisiti
- Misurabilità di requisiti e assunzioni
- Pertinenza di requisiti e assunzioni
- Fattibilità dei requisiti
- Comprensibilità degli elementi del documento dei requisiti
- Buona strutturazione del documento dei requisiti
- Modificabilità degli elementi del documento dei requisiti
- Tracciabilità degli elementi del documento dei requisiti

Errori nei documenti dei requisiti:

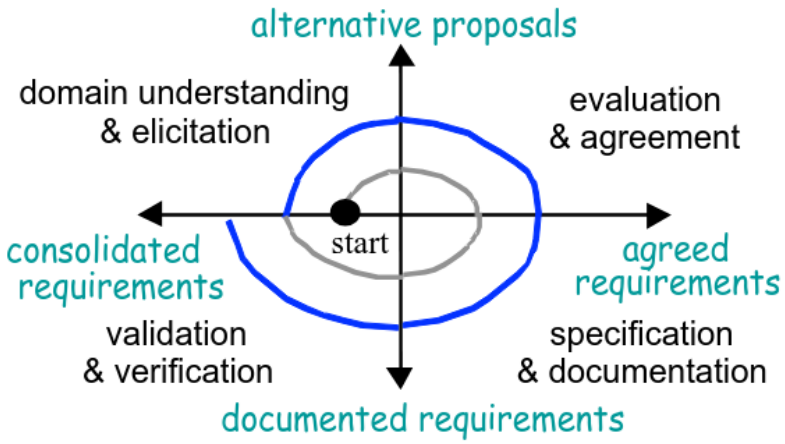
- **Omissione:** caratteristica del mondo del problema non indicata in alcun elemento del documento dei requisiti (RD).
  Esempio: nessun requisito riguardante lo stato delle porte del treno in caso di arresto di emergenza.
- **Contraddizione:** elementi del RD che descrivono una caratteristica del mondo del problema in modo incompatibile.
  Esempio: "Le porte devono essere sempre tenute chiuse tra le piattaforme" e "Le porte devono essere aperte in caso di arresto di emergenza."
- **Inadeguatezza:** elemento del RD che non descrive in modo adeguato una caratteristica del mondo del problema.
  Esempio: "Se un libro non è stato restituito, il prestatore negligente deve essere avvisato che deve pagare una multa."
- **Ambiguità:** elemento del RD che permette di interpretare una caratteristica del mondo del problema in modi diversi.
  Esempio: "Solo le persone che hanno partecipato ad almeno il 75% delle riunioni possono essere premiate alla fine dell'anno."
- **Non misurabilità:** elemento del RD che descrive una caratteristica del mondo del problema in modo tale da impedire il confronto tra opzioni o la verifica delle soluzioni.

Difetti nei documenti dei requisiti:

- **Rumore (Noise):** elemento del RD che non fornisce alcuna informazione su caratteristiche del mondo del problema.
  Variante: ridondanza incontrollata.
  Esempio: "Devono essere affissi cartelli di divieto di fumo sui finestrini del treno."
- **Sovraspecificazione (Overspecification):** elemento del RD che descrive una caratteristica non presente nel mondo del problema, ma nella soluzione della macchina.
  Esempio: "Il metodo setAlarm deve essere invocato al ricevimento di un messaggio Alarm."

- **Non fattibilità (Unfeasibility):** elemento del RD non implementabile entro i vincoli di budget o tempi.
  Esempio: "I pannelli a bordo del treno devono visualizzare tutti i voli in ritardo alla prossima fermata."
- **Inintelleggibilità (Unintelligibility):** elemento del RD incomprensibile per chi deve utilizzarlo.
  Esempio: "Negli Stati Uniti, la nozione di NWO è diventata popolare dopo gli attacchi terroristici al WTC. Tuttavia, i funzionari della NATO e dell'OMC raramente fanno riferimento a un NWO nelle procedure relative al GATT, e si può dire che MVTO, la clausola MFN e gli SRO abbiano poco a che fare con un NOW." (da un comunicato stampa)
- **Scarsa strutturazione (Poor structuring):** elemento del RD non organizzato secondo alcuna regola sensata e visibile di strutturazione.
  Esempio: "Interconnessione dei controllo di accelerazione e problemi al tracking del treno"
- **Riferimento anticipato (Forward reference):** elemento del RD che utilizza caratteristiche del mondo del problema non ancora definite.
  Esempio: uso multiplo del concetto di distanza di arresto nel caso peggiore prima che la sua definizione appaia alcune pagine dopo nel RD.
- **Rimpianto (Remorse):** elemento del RD che definisce una caratteristica del mondo del problema in modo tardivo o incidentale.
  Esempio: dopo molteplici utilizzi del concetto non definito di distanza di arresto nel caso peggiore, l'ultimo uso è seguito direttamente da una definizione incidentale tra parentesi.
- **Scarsa modificabilità (Poor modifiability):** elementi del RD i cui cambiamenti devono essere propagati in tutto il documento.
  Esempio: uso di valori numerici fissi per quantità soggette a variazioni.
- **Opacità (Opacity):** elemento del RD il cui razionale, autore o dipendenze sono invisibili.
  Esempio: "La velocità comandata del treno deve essere sempre almeno 7 mph superiore alla velocità fisica" senza alcuna spiegazione del razionale di questa scelta.

1.3 Processo di RE



Per comprendere il dominio bisogna studiare il sistema as-is, identificare gli stakeholders, così da generare delle prime proposte di prototipi e il glossario dei termini.

1.3.1 Elicitazione dei requisiti

L'**elicitazione** dei requisiti esplora i problemi del mondo facendo ulteriori analisi del problema del sistema as-is, individuandone i sintomi, cause e conseguenze. Vengono identificati:

- Opportunità tecnologiche
- Obiettivi di miglioramento
- Requisiti tecnico-organizzativi del sistema to-be
- Soluzioni alternative per soddisfare gli obiettivi e assegnare le responsabilità
- Scenari ipotetici di interazione software-ambiente
- Requisiti software, assunzioni sull'ambiente

1.3.2 Evaluazione e negoziazione dei requisiti

Presa di decisioni basate sulla negoziazione.

- Identificazione e risoluzione di conflitti
- Identificazione e risoluzione dei rischi del sistema proposto
- Confronto delle alternative tra obiettivi e rischi, selezione della soluzione preferita
- Priorità dei requisiti, per risolvere conflitti, considerare costi e supportare lo sviluppo incrementale.

Vengono prodotti le sezioni finali della proposta di prototipo che documentano gli obbiettivi concordati, i requisiti, le assunzioni e i ragionamenti che hanno portato ad essi.

1.3.3 Specifiche e documentazione

Definizione precisa di tutte le feature del sistema. Obiettivi, proprietà di dominio rilevanti, requisiti, assunzioni, responsabilità. Organizzare questi in una struttura coerente. Documentare in una forma comprensibile a tutti gli interessati. Viene prodotto il Requirements Document (RD).

1.3.4 Consolidazione dei requisiti

Attività per assicurare la qualità del RD. Vengono analizzate l'adeguatezza, la completezza, la mancanza di inconsistenze, vengono poi sistemati gli errori e i difetti. Viene prodotto un RD consolidato.