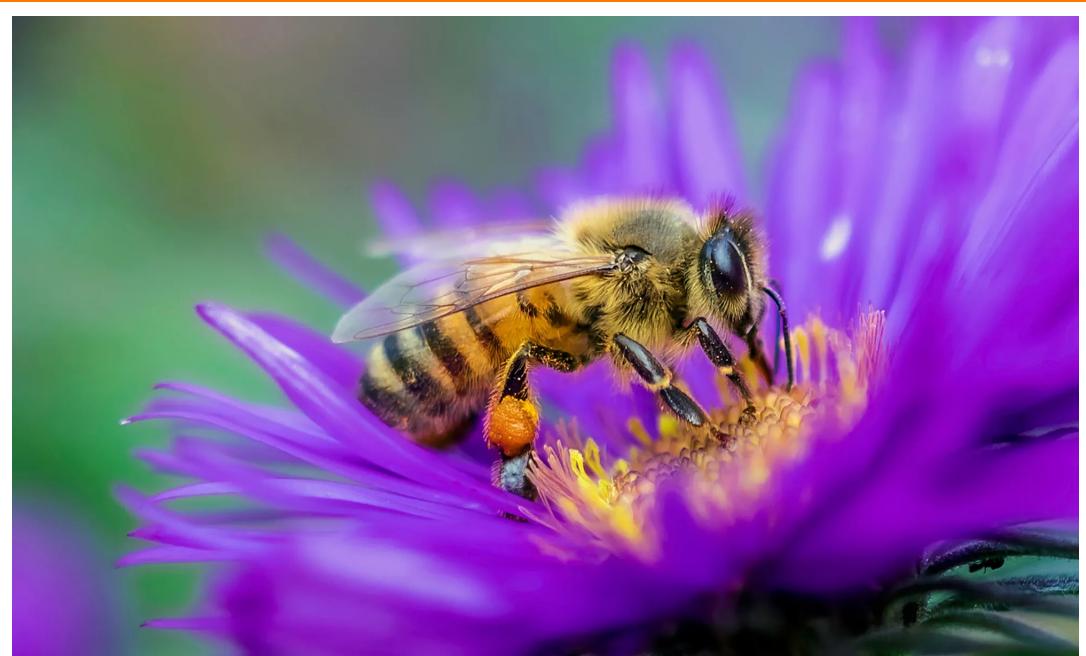


Mattia Robuschi Caprara

Ingegneria del Software



Prefazione

Questi appunti sono una semplice riorganizzazione delle dispense fornite dal professor Agostino Poggi nell'anno accademico 2024/2025, di conseguenza alcune informazioni potrebbero essere troppo riassunte (Ad esempio: alcune sezioni sono solo delle semplici liste che illustrano nomi o temi ma senza approfondirne o spiegarne i significato, al contrario delle dispense) e potrebbe essere consigliato l'utilizzo di questi appunti insieme alle dispense del professore.

Contents

1 Introduzione	4
1.1 Caratteristiche e legami dell'ingegneria del software	4
1.2 Ingegneria del Software e Risoluzione dei Problemi	4
1.3 Relazioni con Altre Discipline	4
1.4 Particolarità del software	4
1.5 Attività principali dell'Ingegneria del software	4
1.6 Proprietà del Software	5
1.7 Capacità necessarie per lo sviluppo di software	5
1.8 Capacità e Conoscenze dell'Ingegnere del software	5
1.9 Ruoli nello sviluppo del software	6
1.10 Responsabilità professionale ed etica	6
1.11 Sistemi software	6
1.12 Stima dei Costi del Software	6
1.13 Dominio applicativo o del problema	6
1.14 Ciclo di vita di un sistema software	7
1.15 Difficoltà dello sviluppo del software	7
1.16 Richieste più frequenti	7
1.17 Problemi tipici	7
1.18 Sfide dell'Ingegneria del software	7
1.19 Stakeholder	8
1.20 Obiettivi degli Stakeholder	8
1.21 Tipi di sistema software	8
2 Attività software di base	10
2.1 Proprietà del Buon Software	10
2.2 Legge sugli Standard di Ambler	10
2.3 Convenzioni di Codifica dei Nomi	10
2.4 Gestione degli Errori	10
2.5 Distribuzione Statistica degli Errori	10
2.6 Fonti degli Errori	10
2.7 Difficoltà nel Riconoscere gli Errori	10
2.8 Passi per Correggere un Errore	10
2.9 Uso di Istruzioni di Stampa	10
2.10 Strumenti a Supporto della Correzione di Errori	10
2.11 Tecniche Principali di Analisi Statica	10
2.12 Testing	10
3 Processi software	10
3.1 Processi e modelli di processi software	10
3.2 Studio di fattibilità	10
3.3 Modello a cascata	10
3.3.1 Problemi del modello a cascata	10
3.3.2 Applicabilità del Modello a Cascata	10
3.3.3 Modello a Cascata – Pro e Contro	10
3.3.4 Modello a cascata con feedback	10
3.3.5 Waterfall V Model	10
3.4 Modello Sviluppo Incrementale - Sviluppo Incrementale	11
3.4.1 Sviluppo e consegna incrementale – Pro e Contro	11
3.4.2 Applicabilità dello sviluppo incrementale	11

3.5	Modello Integrazione e Configurazione	11
3.5.1	Integrazione e Configurazione – Pro e Contro	12
3.6	Distribuzione dei Costi di Sviluppo	12
4	Ingegneria dei requisiti	12
4.1	Definizione di Ingegneria dei Requisiti	12
4.2	Definizione e astrazione dei requisiti	12
4.3	Requisiti dell’Utente e del Sistema	12
4.4	Imprecisione dei Requisiti	12
4.5	Consistenza e Complessità dei Requisiti	12
4.6	Tipi di Requisiti	12
4.7	Classificazione dei Requisiti non Funzionali	12
4.8	Processi dell’Ingegneria dei Requisiti	13
4.9	Raccolta dei Requisiti e relativi problemi	13
4.10	Specifiche dei Requisiti	13
4.11	Specifiche Basate sui Moduli	14
4.12	Specifiche Tabulari	14
4.13	Documento dei Requisiti del Software	14
4.14	Gestione dei Requisiti	14
4.15	Modifica dei Requisiti	14
4.16	Tracciabilità	14
5	Sviluppo di software agile	14
5.1	Metodologia Agile	14
5.2	Extreme programming (XP)	15
5.3	Sviluppo guidato dai test	15
5.4	Programmazione a coppie	15
5.5	Metodi agili e manutenzione del software	15
5.6	Scelta tra metodi agili e guidati dai piani	15
5.7	Coinvolgimento del Cliente	15
5.8	Automazione dei Test	15
5.9	Refactoring	15
5.10	Problemi pratici dei metodi agili	16
5.11	Metodi agili e manutenzione del software	16
5.12	Problemi contrattuali	16
5.13	Problemi organizzativi	16
5.14	Gestione agile dei progetti	16
5.15	Scrum	16
5.16	Scalabilità dei metodi agili	16
6	Modelli dei sistemi e UML	17
6.1	Modellazione del Sistema	17
6.2	Tipi di Modelli	17
6.3	Prospettive dei Sistemi Software	17
6.4	Riuso e attività nell’ambito dei modelli	17
6.5	UML	17
6.5.1	Motivi del Successo di UML	17
6.6	Proprietà ed elementi dei diagrammi dei casi d’uso	18
6.6.1	Elementi dei Casi di Uso	18
6.7	Classi, interfacce e modificatori	18
6.8	Diagramma di Sequenza	18
6.9	Diagramma delle Attività	18
6.10	Diagramma degli Stati	18
6.11	Diagramma degli Oggetti	18
6.12	Diagramma dei Componenti	18
6.13	Diagramma delle Strutture Composite	19
6.14	Diagramma dei Package	19
6.15	Diagramma di Comunicazione	19
6.16	Diagramma di Temporizzazione	19
6.17	Diagramma di Panoramica dell’Interazione	19
6.18	Diagramma di Distribuzione	19

6.19 Diagramma di Robustezza	20
6.20 Tool per il disegno di diagrammi UML	20
7 Modelli e Classi di Analisi	21
7.1 Modello di analisi dei requisiti	21
7.2 Modello di analisi orientati agli oggetti	21
7.3 Classi di analisi	21
7.4 Categorie delle classi di analisi	21
7.5 Tecniche di individuazione delle classi	21
7.6 Errori Comuni di Analisi	22
7.7 Diagramma delle Classi: Descrizione di un appartamento	22
7.8 Diagramma delle Classi: Prenotazioni Online	22
7.9 Diagramma delle Classi: Offerta formativa di una Università	22
8 Scenari e casi d'uso	22
8.1 Storie e scenari	22
8.2 Casi di uso	22
8.3 Casi di Uso - Un Negozio di Vendita Online	23
8.4 Casi di uso - Un altro negozio di vendita online	23
8.5 Descrizione del Caso di Uso: Inserisci l'Ordine	23
8.6 Template per i casi di uso	23
9 Elementi di progettazione	23
9.1 Processo di Progettazione	23
9.2 Concetti e Principi di Progettazione	23
9.3 Modularità	23
9.4 Coesione	23
9.5 Accoppiamento	23
10 Progettazione architetturale	23
10.1 Scopo della Progettazione Architetturale	23
10.2 Requisiti e Progettazione	23
10.3 Diagrammi a Blocchi	23
10.4 Astrazione Architetturale	23
10.5 Progettazione dell'Architettura e Riuso	23
10.6 Pattern Architetturali	23
11 Principi di progettazione	24
12 Testing	24
13 Pattern Software	24
14 Evoluzione del software	24
15 Glossario	24

1 Introduzione

1.1 Caratteristiche e legami dell'ingegneria del software

Si basa su un insieme di tecniche, metodologie e strumenti. Ha l'obiettivo di produrre sistema software di alta qualità che soddisfano: il budget a disposizione, la data di scadenza del rilascio del sistema, e una corretta gestione dei cambiamenti che possono essere necessari durante lo sviluppo.

1.2 Ingegneria del Software e Risoluzione dei Problemi

1. Fase di analisi: in cui si cerca di comprendere la natura del problema e di come suddividerlo in parti.
2. Fase di sintesi: in cui si integrano tra loro le parti per ottenere il sistema finale.

1.3 Relazioni con Altre Discipline

Ha forti legami con:

- I linguaggi di programmazione
- I sistemi operativi
- Le basi di dati
- I modelli teorici
- L'intelligenza artificiale
- Le scienze organizzative
- L'ingegneria dei sistemi

1.4 Particolarità del software

- Il software è immateriale
- Il software richiede alta intensità umana
- Il software è malleabile
- Il software è complesso
- Il software è non Lineare

1.5 Attività principali dell'Ingegneria del software

La realizzazione di sistemi software coinvolge cinque attività principali:

- Lo studio di fattibilità
- La definizione delle specifiche del software
- Lo sviluppo del software
- La verifica del software
- L'evoluzione del software

1.6 Proprietà del Software

Come tutti i tipi di sistemi, un sistema software deve garantire delle buone qualità e un corretto funzionamento. In particolare, il software di un sistema dovrebbe soddisfare un bel numero di proprietà:

- La correttezza
- L'affidabilità
- La robustezza
- Le prestazioni
- L'efficienza
- La scalabilità
- L'usabilità
- La verificabilità
- La manutenibilità
- La riusabilità
- La portabilità
- La comprensibilità
- L'interoperabilità
- La produttività
- La tempestività
- La trasparenza

1.7 Capacità necessarie per lo sviluppo di software

Lo sviluppo di un sistema software richiede le più diverse attività e capacità:

- Risolvere dei problemi
- Acquisire delle conoscenze
- Prendere delle decisioni
- Astrarre oggetti e sistemi
- Definire dei modelli
- etc.

1.8 Capacità e Conoscenze dell'Ingegnere del software

Un ingegnere del software deve essere:

- Un buon programmatore
- Un esperto in strutture dati e algoritmi
- Un esperto in uno o più linguaggi di programmazione
- Familiare con le differenti aree applicative e con più approcci di progettazione
- Capace a muoversi tra i diversi livelli di astrazione di un progetto
- Capace di costruire diverse varietà di modelli
- In grado di fare scelte tra le possibili soluzioni disponibili
- In grado di comunicare e interagire con le persone
- In grado di tradurre descrizioni e risposte (in genere del cliente) anche vaghe in precise specifiche.

1.9 Ruoli nello sviluppo del software

- Il cliente
- Il manager (Gestore del progetto)
- L'analista
- Il programmatore
- Il gestore dei test
- Il consulente
- L'utente

1.10 Responsabilità professionale ed etica

L'ingegneria del software comporta responsabilità più ampie rispetto alla semplice applicazione di competenze tecniche.

Ogni persona coinvolta in un progetto dovrebbe: comportarsi in modo onesto ed eticamente responsabile e offrire competenza e riservatezza.

1.11 Sistemi software

Sono dei sistemi che utilizzano componenti, hardware e software e hanno come obiettivo principale quello di soddisfare le esigenze del cliente.

Un sistema software di successo può rimanere in servizio per parecchi anni e, durante la sua vita, può subire numerose modifiche e estensioni.

1.12 Stima dei Costi del Software

Due tipi di metriche:

- Dimensionali: in cui la stima del costo di base sul numero di linee di codice prodotte, Il numero di istruzioni in “linguaggio macchina” e Il numero di pagine di documentazione del sistema.
- Funzionali: in cui la stima del costo si basa sui numeri di classi, attributi, relazioni, metodi, messaggi, parametri dei messaggi, sorgenti e destinazioni dei messaggi, la percentuale di riuso, il numero di oggetti, la complessità di ciascun oggetto e la produttività.

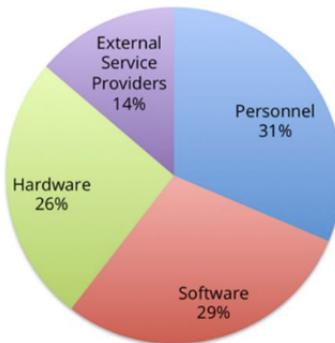


Figure 1: Distribuzione dei costi di sviluppo sistema software

1.13 Dominio applicativo o del problema

In ingegneria del software e in altre discipline informatiche, per dominio applicativo (o dominio del problema) s'intende il contesto in cui opera un'applicazione.

Per limitare i problemi dello sviluppo di un sistema è necessario avere delle buone conoscenze del suo dominio applicativo.

Queste conoscenze possono essere identificate tramite **l'analisi del dominio**.

Quest'analisi ha l'obiettivo di comprendere a fondo i concetti, le dinamiche, le regole generali che definiscono il dominio applicativo in cui il sistema software dovrà essere impiegato.
La qualità dei risultati dipende dal tipo di persone coinvolte, oltre a degli analisti sono coinvolti anche esperti del dominio in questione.

1.14 Ciclo di vita di un sistema software

Il ciclo di vita di un sistema software coinvolge in genere 5 attività principali:

1. L'analisi preliminare
2. La progettazione
3. La realizzazione
4. La gestione
5. La messa fuori servizio del sistema

1.15 Difficoltà dello sviluppo del software

Con il passare del tempo, l'evoluzione delle tecnologie software permette lo sviluppo di sistemi sempre più complessi e quindi i sistemi che si vogliono realizzare richiedono nuove competenze e capacità.

Il processo di sviluppo di un sistema software è difficile da gestire, Questo è dovuto in parte al fatto che il software offre un'**estrema flessibilità** ed è un **sistema discreto**.

Per estremamente flessibile si intende che il software è modificabile con pochissimo sforzo, ma ciò incoraggia un modo di lavorare non pianificato, che può essere un rischio.

La non linearità fa sì che un piccolo cambiamento nel codice può portare a grandi cambiamenti (spesso indesiderati) nel funzionamento di un'applicazione.

Le difficoltà nel realizzare un sistema software sono spesso aumentate dagli obiettivi e dalle richieste degli stakeholder.

1.16 Richieste più frequenti

Sono frequenti le richieste di aggiunta e/o modifica di parti del sistema e le richieste di riduzione dei tempi la consegna.

I risultati dipendono dal peso (tempo in meno e lavora da fare) delle richieste.

1.17 Problemi tipici

- Consegnna del sistema ritardata
- Sforamento del budget
- Bassa affidabilità
- Basse prestazioni del sistema
- Difficoltà nella manutenzione del sistema
- Difficoltà nell'aggiunta di funzionalità al sistema.

Nel caso di sistemi molto complessi, la presenza di diversi risultati negativi può precedere un'elevata probabilità di ritiro del progetto di sviluppo.

1.18 Sfide dell'Ingegneria del software

Il successo dell'ingegneria del software per sviluppare sistemi non solo utili, ma anche molto utilizzati può essere aiutato attraverso la valorizzazioni di alcune proprietà:

- Eterogeneità: la capacità di realizzare sistemi integrando diversi tipi di software e/o hardware
- Cambiamento sociale: comporta un mutamento significativo della società dovuto in gran parte dallo sviluppo tecnologico che offre sempre nuovi interessi
- Sicurezza e Fiducia: due componenti che permettono a un cliente e/o utente di valutare, in modo positivo o negativo, la possibilità di fare delle scelte.
- Evoluzione tecnologica

1.19 Stakeholder

Gli stakeholder sono tutte le persone che in qualche modo sono interessate allo sviluppo di un sistema. Queste persone possono essere:

- Ingegneri
- Programmatori
- Manager aziendali
- Esperti di dominio
- Rappresentanti sindacali
- Utenti finali
- Tutti i membri di un'organizzazione che potrebbero essere interessati all'installazione del sistema software

1.20 Obiettivi degli Stakeholder

Gli stakeholder hanno spesso interessi e obiettivi diversi che in genere dipendono dal ruolo che giocano nel progetto.

Stakeholder	Interesse nel progetto
Sviluppatori	Elevata produttività, prevenzione degli errori, poche rilavorazioni
Venditori	Cifre di vendita elevate, maggiore soddisfazione del cliente
Responsabili del progetto	Riduzione del budget, rispetto del programma
Investitori	Time-to-market più breve, flussi di lavoro più rapidi
Clienti e Utenti	Flusso di lavoro più semplice, buona usabilità

Table 1: Stakeholders e i loro interessi

1.21 Tipi di sistema software

I metodi e gli strumenti di ingegneria del software utilizzati dipendono da: il tipo di applicazione sviluppata, i requisiti del cliente e il background del team di sviluppo.

Tipi di sistema software:

- Sistemi autonomi
- Sistemi interattivi
- Sistemi di controllo integrati (Embedded Control Systems)
- Sistemi di elaborazione batch
- Sistemi di intrattenimento
- Sistemi per la modellazione e simulazione
- Sistemi di raccolta dati (Data Collection Systems)
- Sistemi di sistemi (Systems of Systems)



Figure 2: Livelli di qualità degli stili

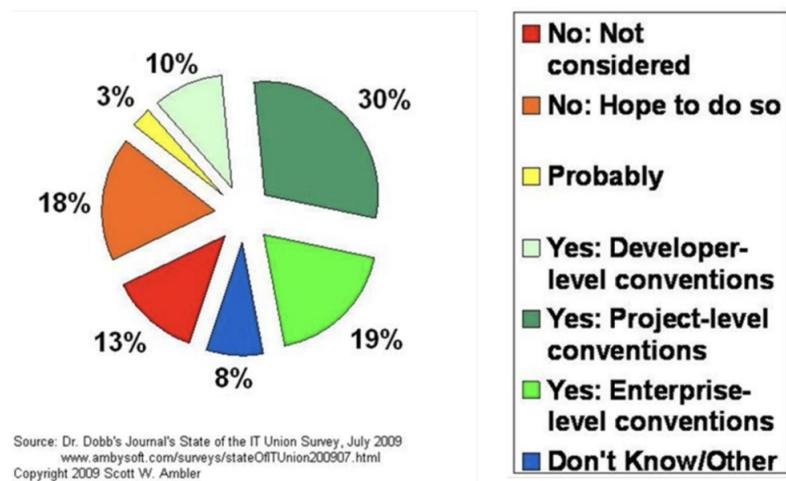


Figure 3: Diffusione dell'uso degli stili

2 Attività software di base

- 2.1 Proprietà del Buon Software
- 2.2 Legge sugli Standard di Ambler
- 2.3 Convenzioni di Codifica dei Nomi
- 2.4 Gestione degli Errori
- 2.5 Distribuzione Statistica degli Errori
- 2.6 Fonti degli Errori
- 2.7 Difficoltà nel Riconoscere gli Errori
- 2.8 Passi per Correggere un Errore
- 2.9 Uso di Istruzioni di Stampa
- 2.10 Strumenti a Supporto della Correzione di Errori
- 2.11 Tecniche Principali di Analisi Statica
- 2.12 Testing

3 Processi software

- 3.1 Processi e modelli di processi software

Plan driven Agili

- 3.2 Studio di fattibilità

- 3.3 Modello a cascata

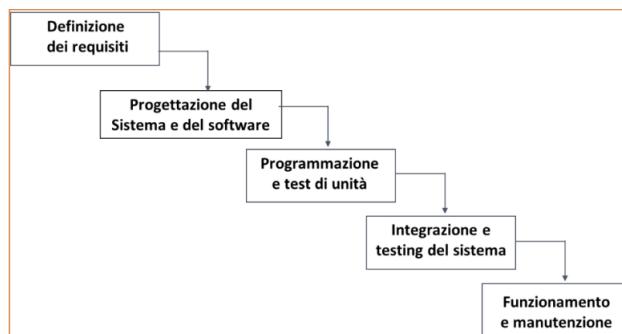


Figure 4: Schema funzionamento modello a cascata

- 3.3.1 Problemi del modello a cascata

- 3.3.2 Applicabilità del Modello a Cascata

- 3.3.3 Modello a Cascata – Pro e Contro

- 3.3.4 Modello a cascata con feedback

Ad ogni step posso tornare ai precedenti

- 3.3.5 Waterfall V Model

Associazione di una fase di testing per ogni fase di sviluppo

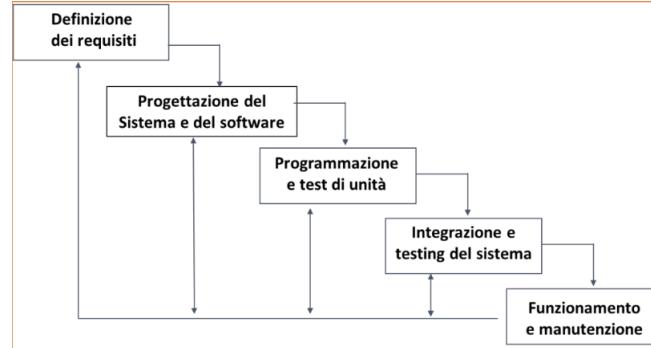


Figure 5: Schema funzionamento modello a cascata con feedback

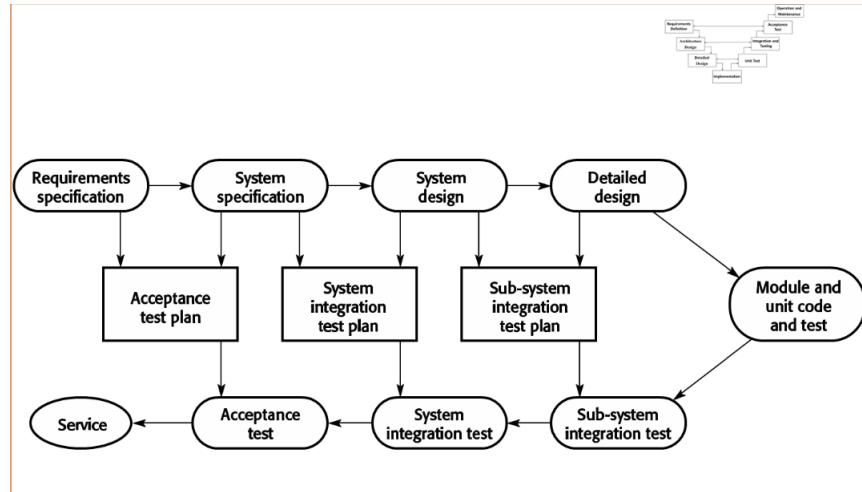


Figure 6: Schema funzionamento modello a cascata V

3.4 Modello Sviluppo Incrementale - Sviluppo Incrementale

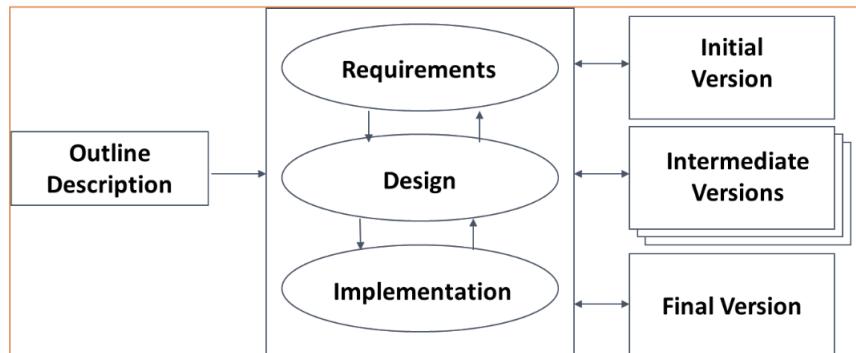


Figure 7: Schema funzionamento sviluppo incrementale

Suddivisione in incrementi

3.4.1 Sviluppo e consegna incrementale – Pro e Contro

3.4.2 Applicabilità dello sviluppo incrementale

3.5 Modello Integrazione e Configurazione

Basato sul riutilizzo del software (componenti, libs, ecc...)

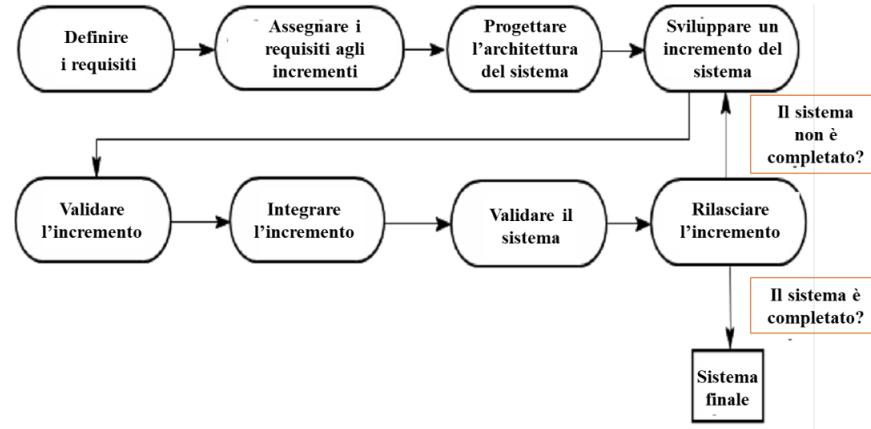


Figure 8: Diagramma funzionamento sviluppo incrementale

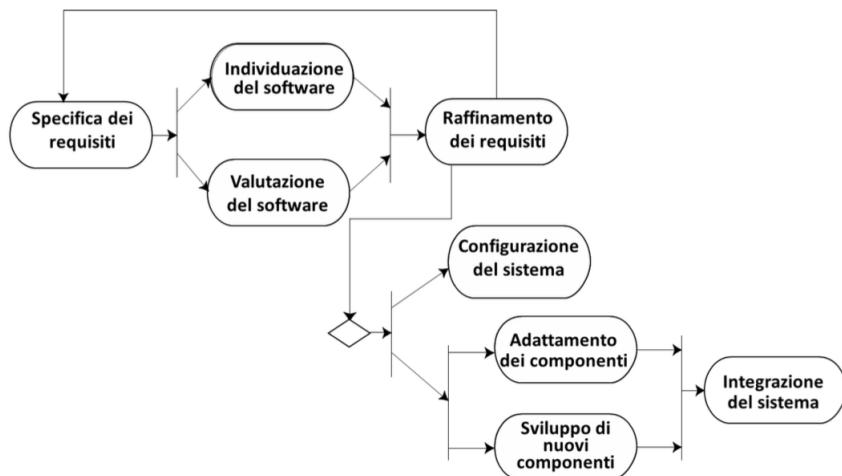


Figure 9: Diagramma funzionamento Integrazione e configurazione

3.5.1 Integrazione e Configurazione – Pro e Contro

3.6 Distribuzione dei Costi di Sviluppo

4 Ingegneria dei requisiti

4.1 Definizione di Ingegneria dei Requisiti

4.2 Definizione e astrazione dei requisiti

4.3 Requisiti dell'Utente e del Sistema

4.4 Imprecisione dei Requisiti

4.5 Consistenza e Complessità dei Requisiti

4.6 Tipi di Requisiti

Funzionali Funzionalità e servizi che il sistema deve fornire Non funzionali Proprietà e vincoli del sistema Di dominio

4.7 Classificazione dei Requisiti non Funzionali

Requisiti del prodotto Requisiti organizzativi Requisiti esterni Obiettivi e Requisiti non Funzionali Verificabili

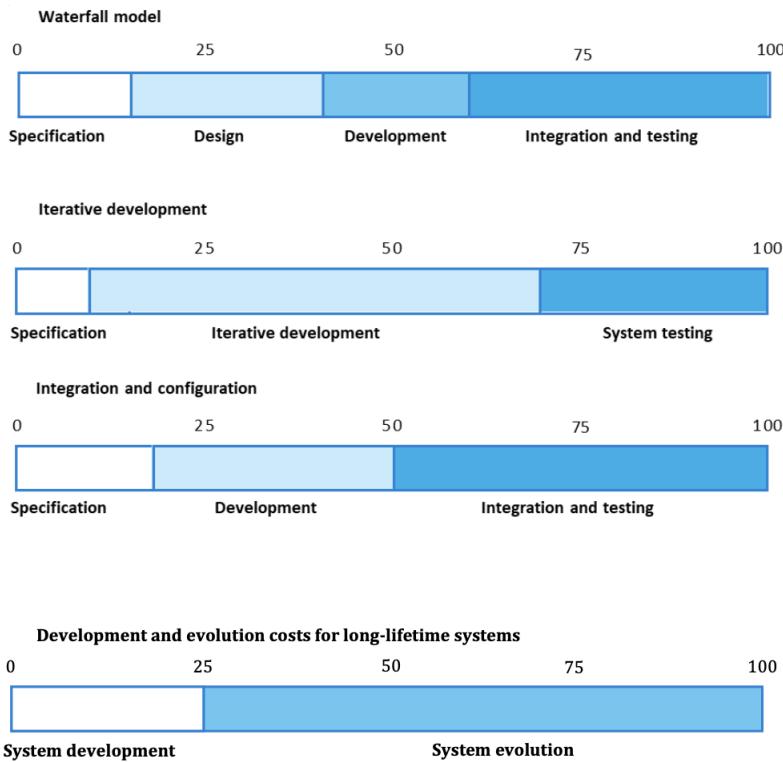


Figure 10: Grafico distribuzione costi di sviluppo

Proprietà	Misure
Velocità	Transazioni elaborate/secondo Tempo di risposta utente/evento Tempo di aggiornamento dello
Dimensioni	Gbyte Numero di chip ROM
Facilità d'uso	Tempo di allenamento Numero di frame di aiuto
Affidabilità	Tempo medio al fallimento Probabilità di indisponibilità Tasso di occorrenza del guasto Disponibilità
Robustezza	Tempo necessario per riavviare dopo il fallimento Percentuale di eventi che causano guasti Probabilità di danneggiamento dei dati in caso di errore
Portabilità	Percentuale di istruzioni dipendenti dall'implementazione Numero di parti/componenti da "studiare"

Table 2: Esempi di metriche per i requisiti non funzionali

4.8 Processi dell'Ingegneria dei Requisiti

Raccolta (Elicitazione) Analisi Convalida Gestione

4.9 Raccolta dei Requisiti e relativi problemi

Tecniche di Raccolta dei Requisiti Interviste Questionari Brain Storming Etnografia (Osservazione degli utenti) Integrazione dell'Etnografia con la Prototipazione

4.10 Specifica dei Requisiti

Linee Guida per la Descrizione dei Requisiti Dettaglia requisiti utente e requisiti di sistema Strumenti per la Descrizione dei Requisiti Requisiti di una Pompa per Insulina

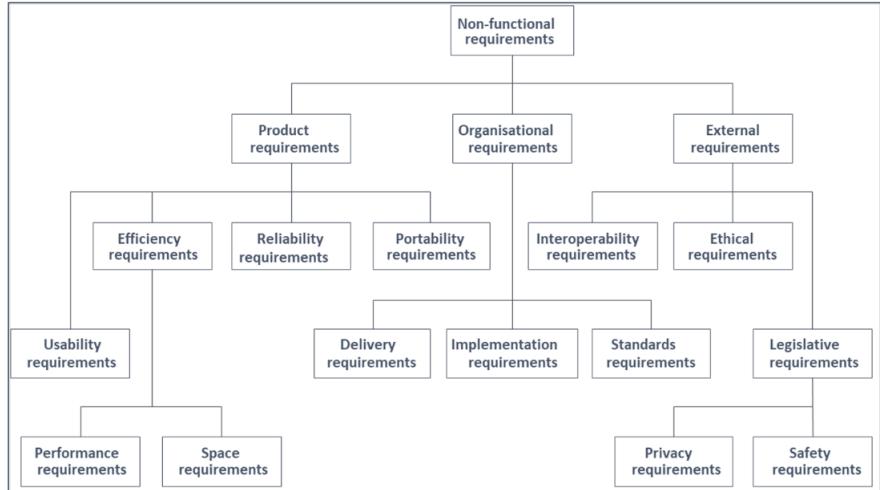


Figure 11: Tipologie dei requisiti non funzionali

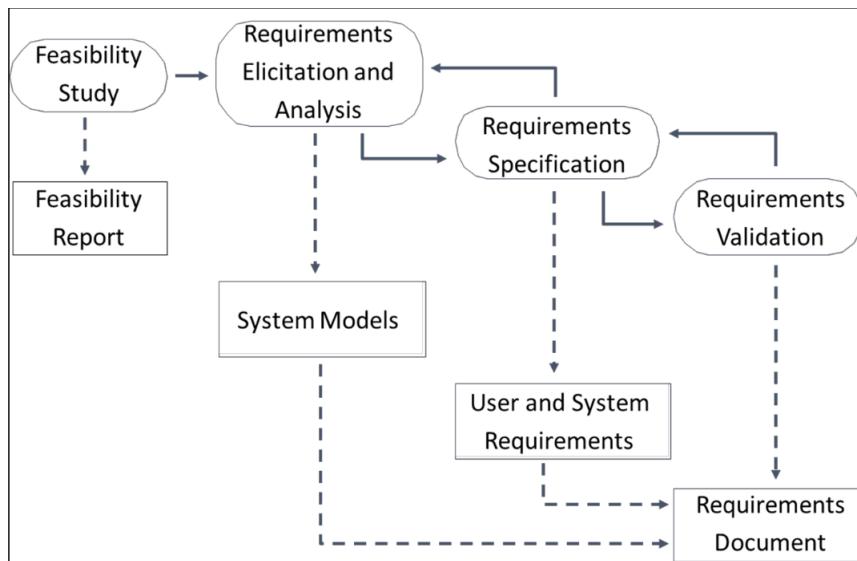


Figure 12: Processo base dell'ingegneria dei Requisiti

4.11 Specifiche Basate sui Moduli

4.12 Specifiche Tabulari

4.13 Documento dei Requisiti del Software

Variabilità e problemi dei Documenti dei Requisiti
Verifica dei Requisiti Tecniche per la Verifica dei Requisiti
Domande di Controllo della Revisione

4.14 Gestione dei Requisiti

Elementi a Supporto della Gestione dei Requisiti

4.15 Modifica dei Requisiti

4.16 Tracciabilità

5 Sviluppo di software agile

5.1 Metodologia Agile

Caratteristiche principali dello sviluppo agile Sviluppo e rilascio rapido del software

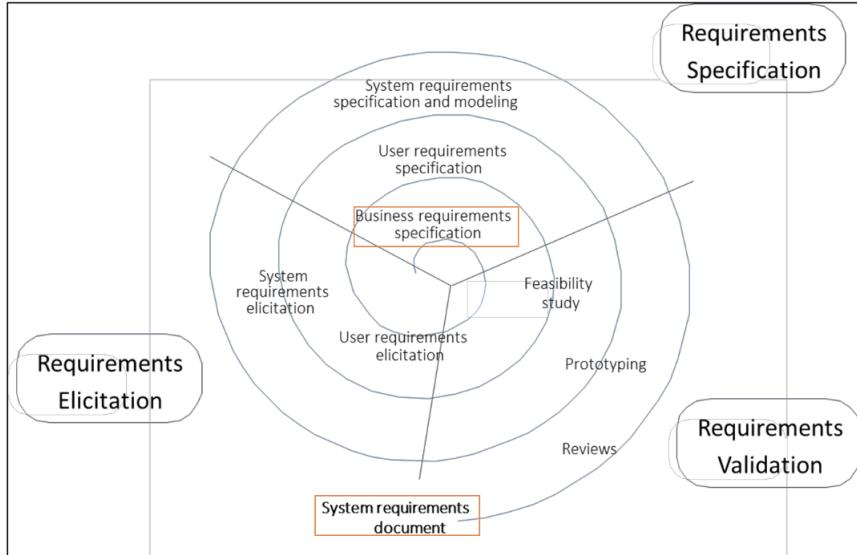


Figure 13: Processo dell'ingegneria dei requisiti a spirale

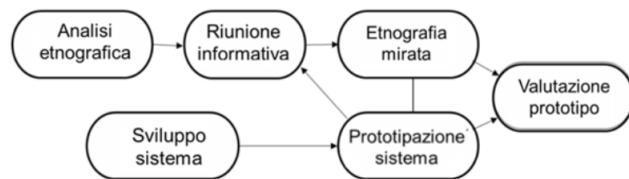


Figure 14: Processo dell'ingegneria dei requisiti a spirale

5.2 Extreme programming (XP)

Requisiti espressi come user stories XP and Principi del Metodo Agile XP & Storie degli Utenti

5.3 Sviluppo guidato dai test

5.4 Programmazione a coppie

5.5 Metodi agili e manutenzione del software

Problemi della manutenzione agile

5.6 Scelta tra metodi agili e guidati dai piani

5.7 Coinvolgimento del Cliente

5.8 Automazione dei Test

Problemi dello Sviluppo basato sui Test 5

5.9 Refactoring

Costante miglioramento del codice Esempi di Refactoring Principi di refactoring

Condizioni	Azioni
Livello di zucchero in calo ($r2 < r1$)	CompDose = 0
Livello di zucchero stabile ($r2 == r1$)	CompDose = 0
Livello di zucchero in crescita e tasso di incremento decrescente ($((r2 - r1) < (r1 - r0))$)	CompDose = 0
Livello di zucchero in aumento e tasso di aumento stabile o in aumento ($((r2 - r1) \geq (r1 - r0))$)	CompDose = rounded result = round $((r2 - r1)/4)$ If rounded result == 0 then CompDose = MinimumDose

Figure 15: Specifica modulare della dose di insulina

<ul style="list-style-type: none"> ◆ Funzione ◆ Calcolare la dose di insulina: livello di zucchero sicuro. 	<ul style="list-style-type: none"> ◆ Azione ◆ CompDose è zero se il livello di zucchero è stabile o in diminuzione o se il livello sta aumentando ma il tasso di aumento sta diminuendo. Se il livello è in aumento e la velocità di aumento è in aumento, CompDose viene calcolato dividendo per 4 la differenza tra il livello di zucchero attuale e il livello precedente e arrotondando il risultato. Se il risultato è arrotondato a zero, allora CompDose è impostato sulla dose minima che può essere erogata.
<ul style="list-style-type: none"> ◆ Descrizione ◆ Calcola la dose di insulina da erogare quando l'attuale livello di zucchero misurato è nella zona sicura tra 3 e 7 unità. 	<ul style="list-style-type: none"> ◆ Requisiti ◆ Due letture precedenti in modo da poter calcolare il tasso di variazione del livello di zucchero.
<ul style="list-style-type: none"> ◆ Ingressi ◆ Lettura attuale dello zucchero ($r2$); le due letture precedenti ($r0$ e $r1$). 	<ul style="list-style-type: none"> ◆ Precondizione ◆ Il serbatoio di insulina contiene almeno la dose singola massima consentita di insulina.
<ul style="list-style-type: none"> ◆ Fonte ◆ Lettura attuale dello zucchero dal sensore. Altre letture a memoria. 	<ul style="list-style-type: none"> ◆ Post-condizione ◆ La post-condizione $r0$ è sostituita da $r1$, quindi $r1$ è sostituito da $r2$.
<ul style="list-style-type: none"> ◆ Uscite ◆ CompDose: la dose di insulina da erogare. 	<ul style="list-style-type: none"> ◆ Effetti collaterali Nessuno.
<ul style="list-style-type: none"> ◆ Destinazione ◆ Ciclo di controllo principale. 	

Figure 16: Specifica tabulare della dose di insulina

5.10 Problemi pratici dei metodi agili

5.11 Metodi agili e manutenzione del software

5.12 Problemi contrattuali

5.13 Problemi organizzativi

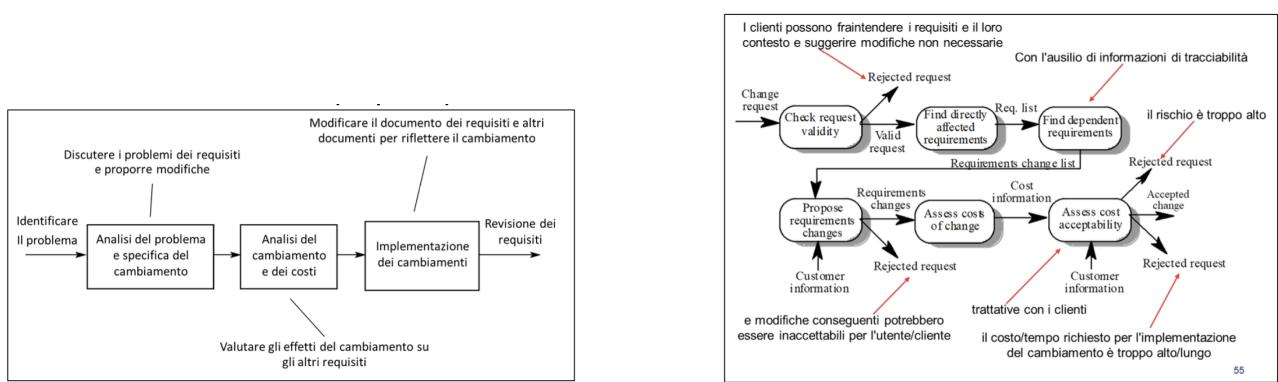
5.14 Gestione agile dei progetti

5.15 Scrum

Basato su sprint, periodi di tempo in cui il team sviluppa e completa una determinata quantità di lavoro Benefici dell'uso di Scrum

5.16 Scalabilità dei metodi agili

Scalabilità verticale e orizzontale Verticale: metodi agili che non possono essere integrati in software di grandi dimensioni Orizzontale: metodi agili che possono essere integrati in una grande organizzazione Sviluppo di sistemi di grossa dimensione Scalabilità verticale di grossi sistemi



(a) Processo di gestione dei cambiamenti Semplificato

(b) Processo di gestione dei cambiamenti Dettagliato

Figure 17: Processo di gestione dei cambiamenti

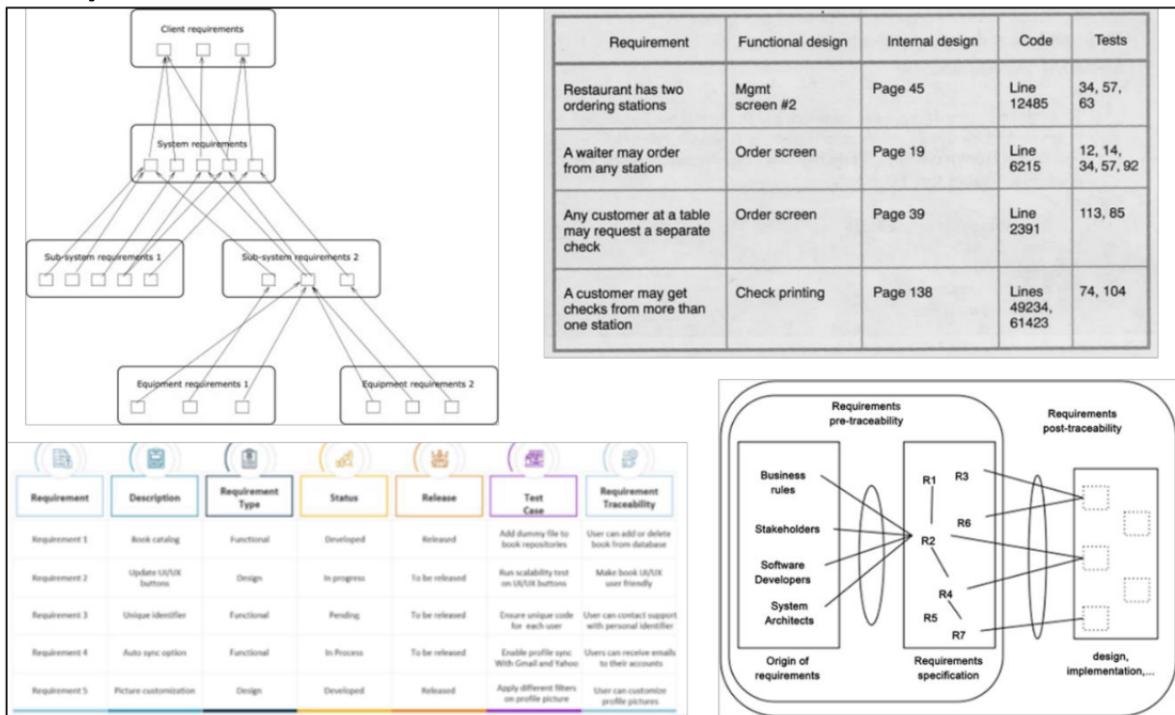


Figure 18: Esempi di documenti di tracciabilità

6 Modelli dei sistemi e UML

6.1 Modellazione del Sistema

6.2 Tipi di Modelli

Modelli pedittivi Modelli estrattivi Modelli prescrittivi

6.3 Prospettive dei Sistemi Software

6.4 Riuso e attività nell'ambito dei modelli

6.5 UML

6.5.1 Motivi del Successo di UML

Diversi studi e ricerche hanno dimostrato l'importanza dell'uso di UML.
Guarda immagine 23 Viste di UML Statistica dell'Uso dei Diagrammi

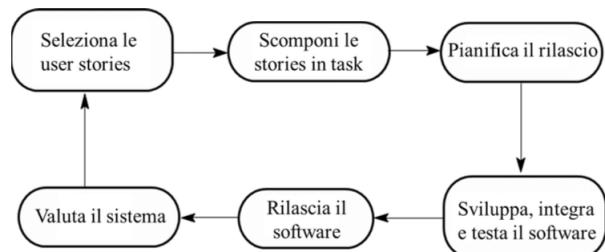


Figure 19: Diagramma ciclo di sviluppo XP

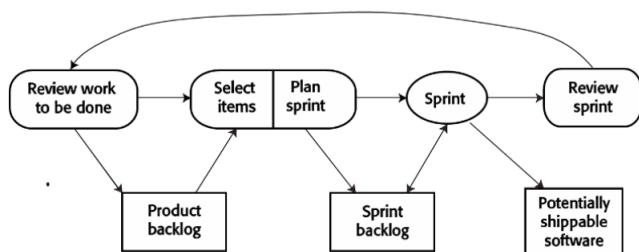


Figure 20: Diagramma del ciclo di sprint

6.6 Proprietà ed elementi dei diagrammi dei casi d'uso

6.6.1 Elementi dei Casi di Uso

Boundary Attori Casi d'uso

6.7 Classi, interfacce e modificatori

Modelli di Classe a Diverso Livello di Dettaglio Associazione Semplice associazione fra classi, definisce le regole per le altre associazioni Generalizzazione, Nesting, Dipendenza e Realizzazione Generalizzazione: Indica ereditarietà, freccia che va dalla SubClass alla SuperClass Nesting: Elemento di origine dentro alla classe Dipendenza: forma debole di relazione Realizzazione: Indica la realizzazione di un'interfaccia Aggregazione e Composizione Aggregazione: La/le classe/i è contenuta all'interno di un'altra classe (Rombo vuoto) Composizione: Le classi compongono una classe più grande (es. albero composto da foglie, rami, tronco, radici, ecc...) (Rombo pieno)

6.8 Diagramma di Sequenza

Lifeline Elementi del diagramma di sequenza Messaggi usati nel diagramma di sequenza

6.9 Diagramma delle Attività

Esempio di Diagramma delle Attività Tipi di Nodi di Controllo Input & Output Pins Decisioni Concorrenza Attività di Raggruppamento Timeout & Signal

6.10 Diagramma degli Stati

Esempio di Diagramma degli Stati Pseudo-stato di Scelta e di Giunzione Stato Composto Storie dello Stato e Regioni Concorrenti

6.11 Diagramma degli Oggetti

Esempio di Diagramma degli Oggetti

6.12 Diagramma dei Componenti

Esempio di Diagramma dei Componenti

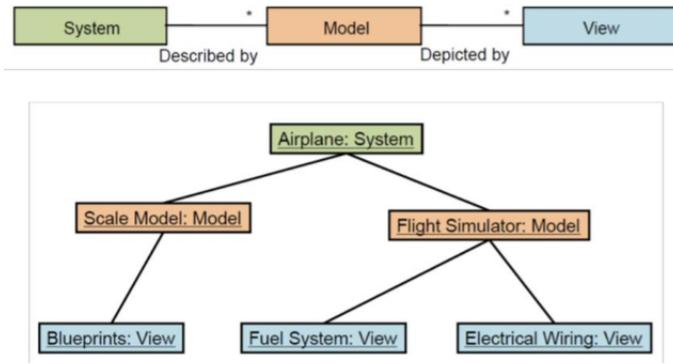


Figure 21: Esempio di modellazione di un sistema

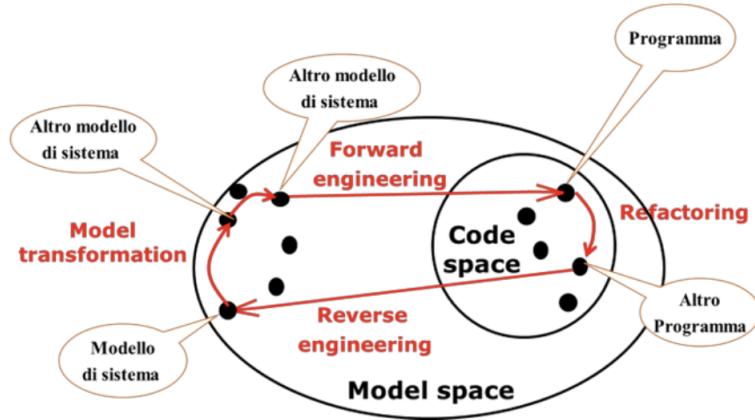


Figure 22: Attività nell'ambito dei modelli

6.13 Diagramma delle Strutture Composite

Esempio di Diagramma delle Strutture Composite

6.14 Diagramma dei Package

Esempio di Diagramma dei Package

6.15 Diagramma di Comunicazione

Esempio di Diagramma di Comunicazione

6.16 Diagramma di Temporizzazione

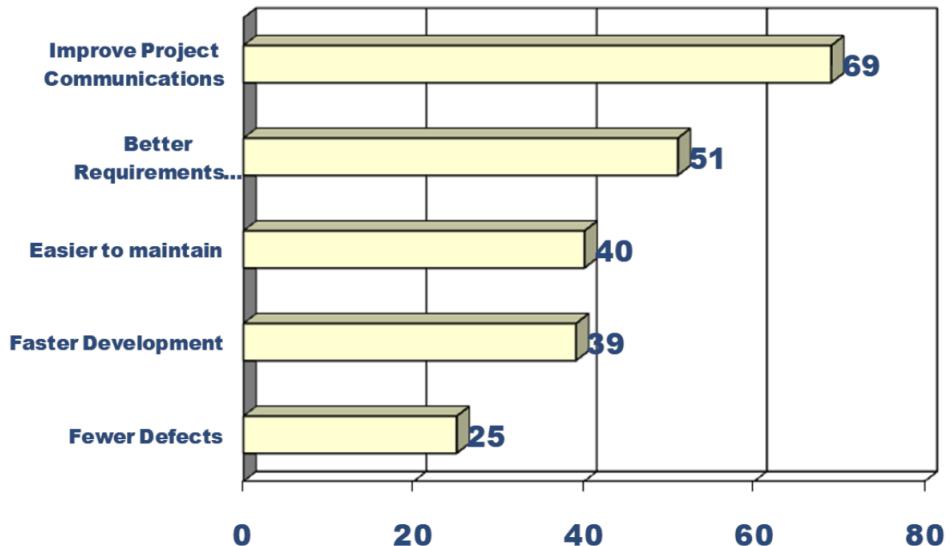
Esempio del Diagramma di Temporizzazione

6.17 Diagramma di Panoramica dell'Interazione

Esempio di Diagramma di Panoramica dell'Interazione

6.18 Diagramma di Distribuzione

Relazioni fra hardware e software Esempio di Diagramma di Distribuzione



Source: BZ Research, August 2004

Figure 23: Motivi del successo di UML

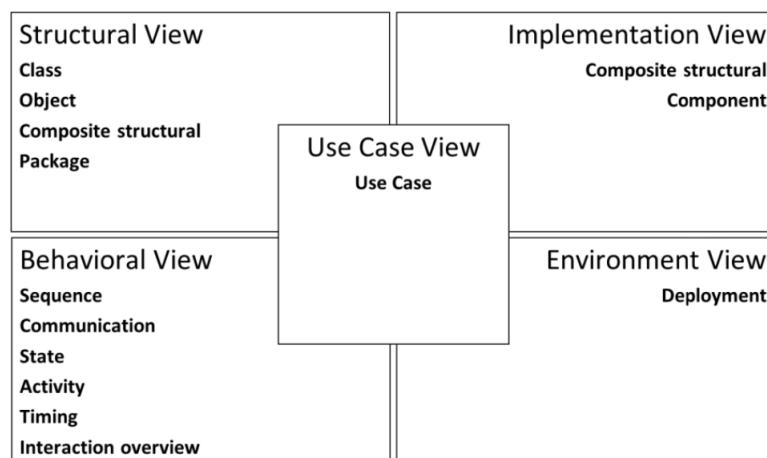


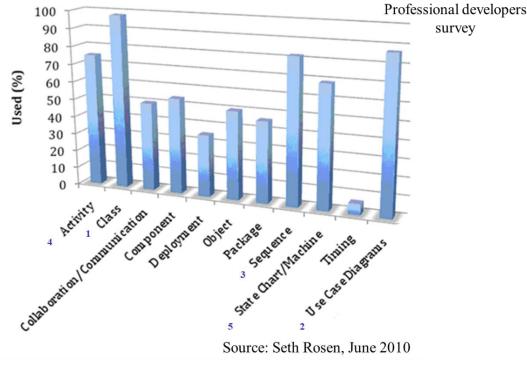
Figure 24: Tipi di diagrammi UML

6.19 Diagramma di Robustezza

Tipi di Nodi Boundary Control Entity Diagramma di robustezza e comunicazione Diagramma di robustezza e sequenza

6.20 Tool per il disegno di diagrammi UML

Eclipse Papyrus e app.diagrams.net

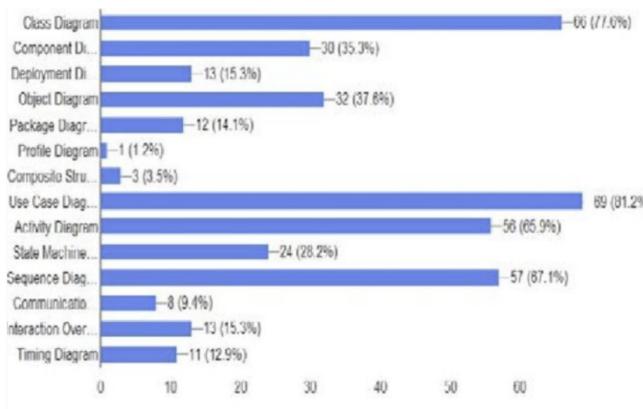


	UML Diagram	Book Guide	Book Spec	Book Tot	Tool	Course	Tutorial	All Sources
1	Class	100%	100%	100%	100%	100%	100%	100%
2	Composite Structure	87%	60%	73%	80%	14%	33%	52%
3	Component	93%	80%	87%	85%	59%	89%	80%
4	Deployment	93%	80%	87%	90%	55%	89%	80%
5	Object	93%	80%	87%	70%	55%	67%	71%
6	Package	100%	79%	89%	65%	52%	67%	70%
7	Activity	100%	93%	97%	100%	95%	100%	98%
8	Sequence	100%	93%	97%	100%	100%	89%	97%
9	Communication	100%	80%	90%	90%	59%	89%	82%
10	Interaction Overview	80%	53%	67%	45%	5%	28%	39%
11	Timing	87%	53%	70%	40%	5%	33%	40%
12	Use Case	100%	93%	97%	100%	95%	89%	96%
13	State Machine	100%	93%	97%	100%	95%	89%	96%

Source: Gianna Reggio, Maurizio Leotta, Filippo Ricca, Diego Clerissi, September 2013

(a)

(b)



From Requirements Engineering to UML using Natural Language Processing – Survey Study - 2017

(c)

Figure 25: Statistiche d'uso dei diagrammi UML

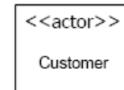


Figure 26: Attore nei casi d'uso

7 Modelli e Classi di Analisi

7.1 Modello di analisi dei requisiti

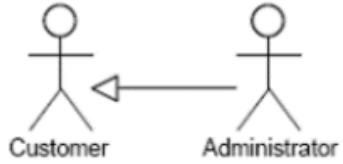
7.2 Modello di analisi orientati agli oggetti

7.3 Classi di analisi

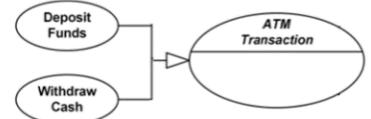
7.4 Categorie delle classi di analisi

7.5 Tecniche di individuazione delle classi

Analisi nome – verbo Approccio guidato dai casi di uso Modelli di classi comuni Classi, responsabilità e collaborazioni (CRC) Approccio Misto



(a) Generalizzazione tra attori



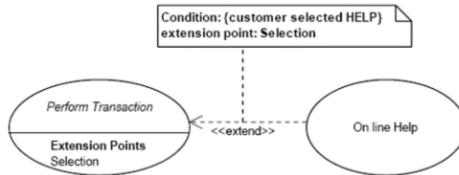
(b) Generalizzazione tra casi d'uso



(c) Inclusione tra casi d'uso



(d) Estensione tra casi d'uso



(e) Estensione condizionata

Figure 27: Operazioni diagramma casi d'uso

Componente del testo	Modello del componente	Esempio
Nome proprio	Istanza	Jim Smith
Nome comune	Classe	Giocattolo, bambola
Verbo di fare	Metodo	Comprare, consigliare
Verbo di classificazione	Eredità	è un
Verbo di possessione	Aggregazione	Ha un
Verbo modale	Vincolo	Deve essere
Aggettivo	Attributo	Di tre anni
Verbo transitivo	Metodo	Entra, modifica
Verbo intransitivo	Metodo (evento)	Dipende da

Table 3: Schema analisi nome-verbo

7.6 Errori Comuni di Analisi

7.7 Diagramma delle Classi: Descrizione di un appartamento

7.8 Diagramma delle Classi: Prenotazioni Online

7.9 Diagramma delle Classi: Offerta formativa di una Università

8 Scenari e casi d'uso

8.1 Storie e scenari

8.2 Casi di uso

Ranking dei casi di uso Operazioni principali dei casi di uso Guide per lo sviluppo dei casi di uso

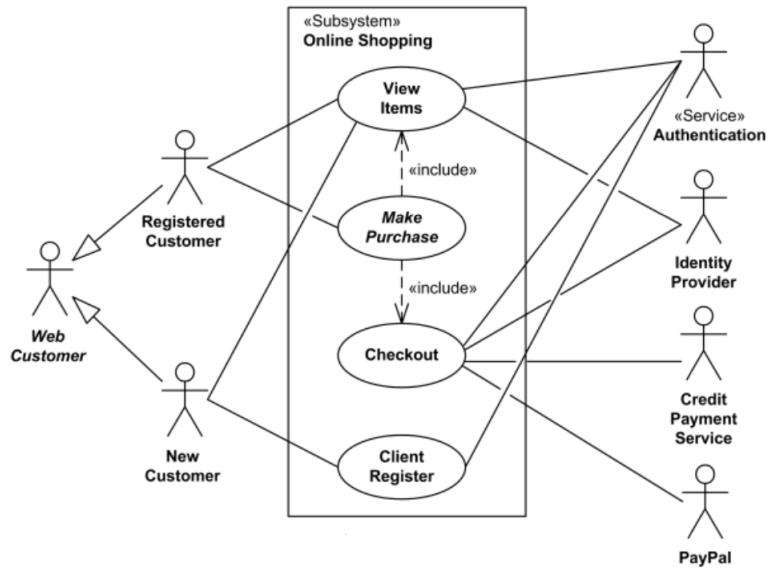


Figure 28: Esempio di caso di uso

8.3 Casi di Uso - Un Negozio di Vendita Online

8.4 Casi di uso - Un altro negozio di vendita online

8.5 Descrizione del Caso di Uso: Inserisci l'Ordine

Attori, Trigger, PreCondizioni e PostCondizioni Flusso Principale Flusso Alternativo 1 Flusso Alternativo 2 Flusso Alternativo 3

8.6 Template per i casi di uso

9 Elementi di progettazione

9.1 Processo di Progettazione

9.2 Concetti e Principi di Progettazione

9.3 Modularità

Difficoltà ad Individuare la Migliore Modularità

9.4 Coesione

9.5 Accoppiamento

10 Progettazione architetturale

10.1 Scopo della Progettazione Architetturale

10.2 Requisiti e Progettazione

10.3 Diagrammi a Blocchi

10.4 Astrazione Architetturale

10.5 Progettazione dell'Architettura e Riuso

10.6 Pattern Architetturali

Architettura - Model-View-Controller (MVC) Architettura - Layered Architettura - Repository Architettura - Pipe & Filter

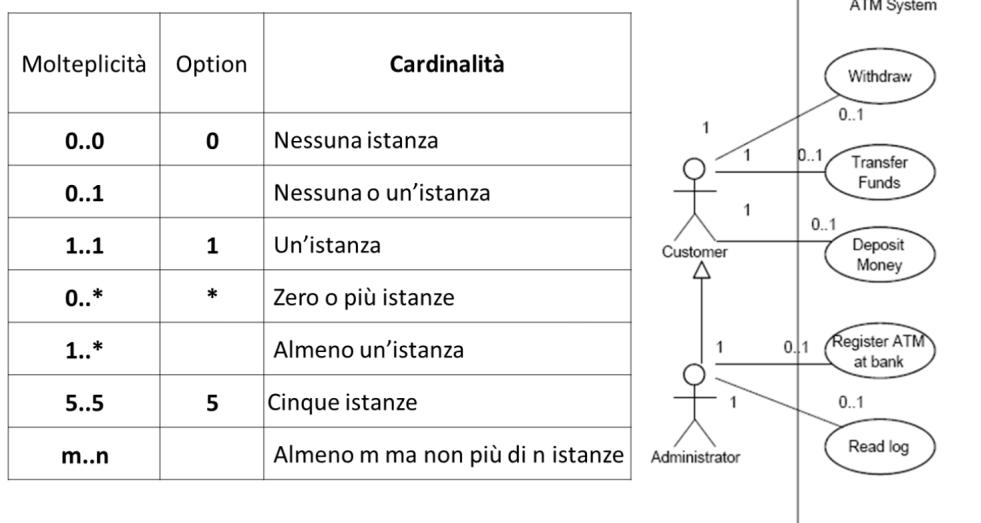


Figure 29: Connettori e Valori di Molteplicità

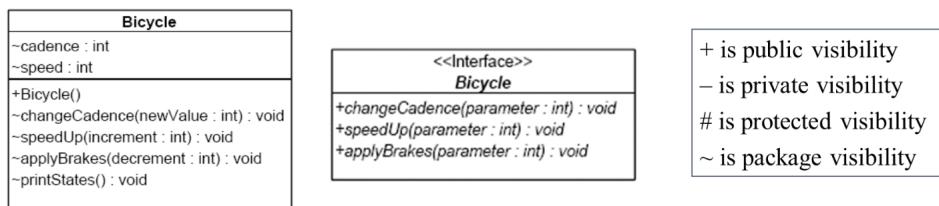


Figure 30: Legenda modificatori di visibilità

11 Principi di progettazione

12 Testing

13 Pattern Software

14 Evoluzione del software

15 Glossario

Dominio applicativo Si intendono, qui, le conoscenze relative ad un determinato ambito. Ad esempio, sviluppando un sistema per Trenitalia, il dominio sarebbe quello della gestione dei treni. Il problema sta nel fatto che il programmatore non ha conoscenze al riguardo dei treni, e deve utilizzare quindi un linguaggio, implicazioni e conoscenze che non gli appartengono.

I sistemi software che si possono realizzare hanno caratteristiche differenti e quindi il loro sviluppo implica la conoscenza delle caratteristiche del sistema da realizzare e dell'ambiente in cui si trova. Queste conoscenze possono essere identificate tramite l'analisi del dominio che ha l'obiettivo di comprendere a fondo i concetti, le dinamiche, le regole generali che definiscono il dominio applicativo in cui il sistema software dovrà essere impiegato, ovvero il contesto in cui il software dovrà agire. L'analisi del dominio deve quindi essere condotta congiuntamente da analisti ed esperti del dominio (e.g., i clienti stessi che commissionano lo sviluppo del sistema, e/o i probabili utenti del sistema stesso).

Funzionalità è un servizio che il sistema dovrebbe fornire. La sua descrizione dovrebbe indicare come il sistema deve reagire a particolari input e come il sistema deve comportarsi in situazioni particolari.

Requisiti sono le descrizioni dei servizi che un sistema software deve fornire e dei vincoli in base ai quali deve operare.

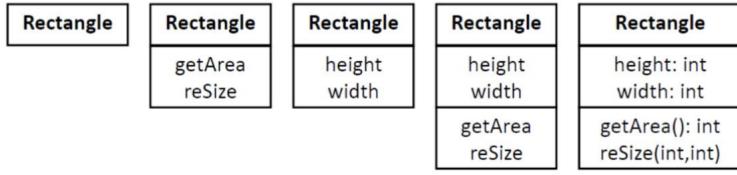


Figure 31: Esempio dei diversi livelli di dettaglio di una classe

Specificità è una descrizione precisa dei requisiti (proprietà o comportamenti richiesti) di un sistema o di una sua parte.

Stakeholder Sono le persone che in qualche modo sono interessate a un sistema. Possono essere ingegneri, programmati che stanno sviluppando il sistema o mantenendo sistemi correlati, manager aziendali, esperti di dominio e rappresentanti sindacali. Possono anche essere gli utenti finali che interagiscono con il sistema e tutti gli altri membri di un'organizzazione che potrebbero essere interessati dalla sua installazione.

User Stories Descrizione generale e informale di una funzionalità software presentata dall'utente finale con lo scopo di articolare il modo in cui questa funzionalità software fornirà valore al cliente.

Scenari Descrizioni in modo narrativo che raccontano la storia dell'interazione di un utente con un prodotto. Forniscono un contesto più dettagliato rispetto alle storie degli utenti, introduce le motivazioni, le azioni e il flusso complessivo di interazione con il prodotto.

CASE Tools Il CASE tool è un software che supporta la progettazione di sistemi software, ad esempio con UML.

Constraint un constraint è un limite che viene posto; restriction, limitation.

Volatility Con RV intendiamo modifiche ai requirements che avvengono durante lo sviluppo del progetto.