CAPITOLO 2

SOFTWARE REQUIREMENTS (SWEBOK, 2004)

Ercole Colonese e Benedetto Serra, Consulenti

ACRONIMI

DAG	Directed Acyclic Graph
FSM	Functional Size Measurement
INCOSE	International Council on Systems Engineering
KA	Knowledge Area
SADT	Structured Analysis and Design Technique
UML	Unified Modeling Language

INTRODUZIONE

L'area di conoscenza *Software Requirements Knowledge Area* (*KA*) riguarda la "elicitazione", l'analisi, la specifica e la validazione dei requisiti del software. E' generalmente riconosciuto nell'ambito dell'industria del software che i progetti d'ingegneria del software siano gravemente vulnerabili quando queste attività siano condotte in maniera superficiale.

I requisiti software esprimono i bisogni e i vincoli riguardanti un prodotto software che contribuisce a risolvere un problema del mondo reale. [Kot00]

L'espressione "ingegneria dei requisiti" (*Requirements Engineering*) è largamente usata in questo settore per indicare la gestione sistematica dei requisiti.

Per motive di coerenza, tuttavia, questa espressione non sarà utilizzata in questa Guida, in quanto è stato deciso di non usare il termine "engineering" per attività diverse da quelle del software engineering.

Per lo stesso motivo, non sarà utilizzato nemmeno l'espressione "Requirements Engineer", che compare talvolta nella letteratura specializzata.

Al suo posto sarà utilizzata l'espressione "Software Engineer" oppure, in alcuni casi specifici, "Specialista dei requisiti" (*Requirements Specialist*), quest'ultimo quando il ruolo in questione sia svolto da una persona che non sia un "Software Engineer". Questo non vuol dire però che un "Software Engineer" non possa svolgere questa funzione.

Lo schema della KA è in linea di massima compatibile con le sezioni della IEEE 12207 che si riferiscono alle attività di raccolta dei requisiti. (IEEE12207.1-96)

Uno dei rischi inerenti allo schema proposto consiste nel concepire tale modello come un processo "a cascata". Per evitare questo rischio, la sottoarea 2 "Il processo di analisi dei requisiti" è progettato per fornire una visione d'insieme del processo medesimo, stabilendo le risorse e i vincoli che contribuiscono a configurare il modo in cui operare.

Un modo alternativo di scomporre il processo in attività consiste nell'utilizzare una struttura basata sulla progettazione di prodotto (*System Requirements*, *Software Requirements*, *Prototipi*, *Use case* e così via).

Lo schema basato sul processo riflette il fatto che detto "processo dei requisiti", per avere successo, deve essere trattato come un elemento che comporta attività complesse e strettamente legate tra loro, (condotte sia in modo sequenziale sia in parallelo) piuttosto che un'attività di tipo discreto, on-off, svolta nelle fasi iniziali di un progetto di sviluppo software.

La Software Requirements Knowledge Area è strettamente collegata ad altre KA: Software Design, Software Testing, Software Maintenance, Software Configuration Management, Software Engineering Management, Software Engineering Process, e Software Quality

SCOMPOSIZIONE DEGLI ARGOMENTI RIGUARDANTI I REQUISITI DEL SOFTWARE

1. Fondamenti dei requisiti software

1.1. Definizione di un requisito software

Nella sua essenza, un requisito software è una proprietà che deve essere fornita e dimostrata per risolvere un problema del mondo reale.

La presente Guida fa riferimento ai requisiti del software perché riguarda problematiche che devono essere affrontate dal software. Per questo motivo, un requisito software è una proprietà che deve essere dimostrata dal software che è stato sviluppato o adattato per risolvere un problema particolare. Il problema può riguardare l'automazione di un compito (in tutto o in parte) svolto da qualcuno che farà uso del software, per supportare i processi di business dell'organizzazione che ha commissionato lo sviluppo, per correggere inconvenienti di un software già esistente, per controllare uno strumento e per molte altre cose ancora. Il modo di operare degli utenti, dei processi di business e degli strumenti è generalmente complesso. Per estensione, quindi, i requisiti di un particolare software sono solitamente una combina-

zione complessa di esigenze provenienti da persone diverse, a differenti livelli dell'organizzazione, con vincoli diversi anche per l'ambiente in cui il software dovrà operare.

Una proprietà essenziale di tutti i requisiti software consiste nel loro essere verificabili.

Può essere difficile o costoso verificare certi requisiti. Per esempio, la verifica del requisito di "throughput" di un Call Center può richiedere lo sviluppo di un software di simulazione.

Sia il personale addetto alla raccolta dei requisiti, sia quello addetto alla qualità del software deve assicurarsi che i requisiti possano essere verificati nel contesto delle risorse disponibili.

I requisiti hanno anche altri attributi, oltre alle proprietà comportamentali che esprimono. Tra questi esempi si può includere la necessità di stabilire la gerarchia delle priorità in modo da poter affrontare eventuali problemi legati a risorse limitate e la definizione di una variabile per misurare lo stato in cui si trova il progetto e monitorare l'avanzamento dello stesso. Di solito i requisiti software sono identificati univocamente, in modo che possano essere gestiti con strumenti di "configurazione del software" per tutta la durata del loro ciclo di vita. [Kot00; Pfl01; Som05; Tha97].

1.2. Requisiti di prodotto e di processo.

Una distinzione può essere tracciata tra parametri di prodotto e parametri di processo. I parametri di prodotto sono requisiti richiesti al software da sviluppare, come ad esempio: "Il software deve verificare che uno studente sia in possesso di tutti i requisiti richiesti prima che possa inscriversi ad un corso". I parametri di processo sono essenzialmente vincoli allo sviluppo, ad esempio: "Il software deve essere scritto utilizzando il linguaggio Ada".

1.3. Requisiti funzionali e non funzionali

I *requisiti funzionali* descrivono le funzioni che il software deve eseguire; per esempio, la formattazione di una porzione di testo o la modulazione di un segnale. Essi sono denominati "capabilities".

I *requisiti non funzionali* sono quelli che contribuiscono a imporre vincoli alla soluzione. Essi, infatti, sono chiamati, a volte, anche "vincoli" o "requisiti di qualità".

1.4. Proprietà emergenti

Alcuni requisiti rappresentano proprietà emergenti del software, cioè esigenze che non possono essere affrontati da un singolo componente, ma che dipendono, per essere soddisfatti, da come interagiscono tutti i componenti del software. Il requisito di "troughput" di un Call Center, ad esempio, dipenderà da come il sistema telefonico, il sistema informativo e gli operatori interagiscono tutti insieme nelle reali condizioni operative. Le proprietà emergenti dipendono in modo cruciale dall'architettura del sistema. [Som05]

1.3. Requisiti quantificabili

I requisiti del software devono essere stabiliti nel modo più chiaro e meno ambiguo possibile. Tutte le volte in cui sia appropriato, essi dovrebbero essere anche quantificati. E' importante, dunque, evitare requisiti vaghi e non verificabili, che dipendano per la loro interpretazione da un giudizio soggettivo (es.: "il software dovrà essere affidabile", "il software dovrà essere amichevole").

Ciò è particolarmente importante per i requisiti non funzionali. Ecco due esempi di requisiti quantificati: il software di un Call Center deve incrementare il throughput del 20%; un sistema deve avere una probabilità inferiore a $1*10^{-8}$ di generare un errore fatale in un'ora di operatività. Il requisito di throughput è generalmente definito a un livello molto alto e avrà quindi bisogno di essere dettagliato in un grande numero di requisiti di livello inferiore. Il requisito di affidabilità costituirà un vincolo molto stringente per l'architettura di sistema. [Dav93; Som05]

1.5. Requisiti software e requisiti di sistema

In questa sezione, il termine "sistema" significa "Una combinazione di elementi che interagiscono per raggiungere un obiettivo definito". Esso può includere hardware, software, firmware, personale, informazioni, facilities, servizi e altri elementi di supporto, così come definito dall'International Council on Systems Engineering (INCOSE00).

I requisiti di sistema sono i esigenze che concernono il sistema nel suo insieme. In un sistema che contiene componenti software, tali requisiti (software) derivano da quelli del sistema.

La letteratura sui requisiti chiama solitamente i requisiti di sistema come "requisiti utente". La presente Guida definisce i "requisiti utente" in modo più ristretto, cioè come i requisiti degli utenti finali o dei clienti del sistema. Diversamente, i requisiti di sistema comprendono quelli degli utenti, quelli degli altri stakeholder (come le autorità regolatrici) e i rimanenti requisiti che non abbiano una fonte "umana" ben identificabile.

2. Il processo di raccolta dei requisiti

Questa sezione introduce il processo di raccolta dei requisiti del software, traguardando le cinque rimanenti sottoaree e mostrando come il processo della loro raccolta sia collegato a tutto il processo d'ingegneria del software, nella sua totalità. [Dav93; Som05]

2.1. Modelli di processo

Obiettivo di questa sezione è fornire una spiegazione dei concetti elencati qui di seguito. Il processo di raccolta dei requisiti:

- non è un'attività iniziale o una "una tantum" del ciclo

di vita del software, ma piuttosto costituisce un processo che nasce con l'inizio del progetto e che continua ad essere rifinito per tutta la durata del ciclo di vita del progetto;
- identifica i requisiti del software come elementi di configurazione (*Configuration Item* - CI) e li gestisce utilizzando le stesse pratiche di *Software Configuration Management* utilizzate per gli altri prodotti dei processi del ciclo di vita del software;
- deve essere adattato al contesto dell'organizzazione e del progetto.

In particolare, l'argomento in oggetto riguarda come le attività di elicitazione, analisi, specifica e validazione dei requisiti sono configurate a fronte di differenti tipi di progetti e di vincoli. La trattamento comprende anche attività che forniscono l'input al processo dei requisiti, come il Marketing e gli Studi di fattibilità. [Kot00; Rob99; Som97; Som05]

2.2. Attori del processo

Questa sezione introduce i ruoli delle persone che partecipano al processo di raccolta dei requisiti. Il processo è fondamentalmente inter-disciplinare e lo specialista dei requisiti deve mediare tra il dominio dello stakeholder e quello dell'ingegneria del software. Spesso sono coinvolte molte persone, oltre allo specialista dei requisiti; ciascuna è portatore di un proprio interesse relativamente al software. Gli stakeholder possono variare da un progetto all'altro, ma includono sempre gli utenti, gli operatori e i clienti (che non sono necessariamente le stesse persone). [Gog93]

Esempi tipici di stakeholder del software includono, ma non si limitano a:

- *Utenti*: questa categoria comprende le persone che opereranno con il software. Si tratta spesso di un gruppo eterogeneo di persone che comprendono differenti ruoli ed esigenze.
- *Clienti*: questo gruppo comprende le persone che hanno commissionato il software o che costituiscono il mercato target del software.
- *Mercato* (Analisi di mercato): un prodotto per il grande mercato (Mass-Market) non avrà un cliente specifico che lo commissiona; è quindi richiesto al personale del Marketing di definire quali siano i bisogni del mercato, agendo in rappresentanza dei clienti stessi.
- *Autorità regolatrici*: molti domini applicativi, come ad esempio quello bancario o dei trasporti, sono sottoposti a vincoli regolatori. Il software in questi domini deve essere conforme ai requisiti delle autorità regolatrici.
- *Progettista software:* queste persone hanno un legittimo interesse nell'attività di sviluppo; ad esempio, possono chiedere di riutilizzare alcune componenti in altri prodotti. Se, in questo scenario, un cliente di un particolare prodotto ha requisiti specifici che compromettono il potenziale riutilizzo di una componente, i progettisti software devono pesare attentamente il proprio interesse e confrontarlo con quello del cliente.

Non sarà sempre possibile soddisfare perfettamente i requisiti di ciascun stakeholder; in questi casi, sarà compito del progettista negoziare compromessi che siano accettabili per i principali stakeholder e rispettosi dei limiti di budget, tecnici, regolatori ecc. Prerequisito a tutto ciò è che tutti gli stakeholder risultino identificati e che la natura del loro interesse sia identificata, analizzata e che i loro requisiti siano elicitati. [Dav93; Kot00; Rob99; Som97; You01]

2.3. Supporto al processo e gestione

Questa sezione riguarda le risorse di Project Management richieste e utilizzate dal processo di raccolta dei requisiti. La sezione stabilisce il contesto per la prima sottoarea (*Initiation and Scope Definition*) della KA del "Software Engineering Management". Suo principale obiettivo è stabilire il legame tra le attività identificate in *2.1 Process Planning* e le problematiche riguardanti costi, risorse umane, formazione e strumenti. [Rob99; Som97; You01]

2.4. Processi della qualità e miglioramento

Questa sezione riguarda la valutazione della qualità e il miglioramento del processo di raccolta dei requisiti. L'obiettivo è porre l'accento sull'importanza che il processo di raccolta dei requisiti riveste ai fini del costo, della puntualità di rilascio del prodotto software e della soddisfazione del cliente. [Som97]. La trattazione aiuterà ad orientare il processo di raccolta dei requisiti con gli standard della qualità e con i modelli di miglioramento del processo per il software e i sistemi. La qualità del processo e il suo miglioramento sono in stretta relazione sia con la Knowledge Area *Software Quality* sia con quella *Software Engineering Process*. Di particolare interesse sono le problematiche riguardanti gli attributi di qualità del software, la loro misura e la definizione del processo software. L'argomento include:

- la copertura del processo di raccolta dei requisiti da parte degli standard e dei modelli di miglioramento del processo (*Process Improvement*);
- le misurazioni del processo di raccolta dei requisiti e il Benchmarking;
- la pianificazione del miglioramento e la sua implementazione [Kot00; Som97; You01].

3. Elicitazione dei requisiti

[Dav93; Gog93; Lou95; Pfl01]

L'elicitazione dei requisiti è connessa alla "fonte" dei requisiti software (da dove essi provengono) e al "come" i requisiti possono essere raccolti dal Software Engineer (l'Analista). E' il primo stadio della costruzione e della comprensione del problema che il software è chiamato a risolvere. Si tratta fondamentalmente di un'attività umana, ed è in quest'ambito che gli stakeholder sono identificati e che le relazioni tra il team di sviluppo e il cliente sono stabilite. L'attività è chiamata in diversi modi: "cattura dei requisiti", "scoperta dei requisiti", "raccolta dei requisiti".

Una delle regole fondamentali per una buona ingegneria del software è stabilire una comunicazione efficace tra ingegneri del software e utilizzatori del prodotto[1]. Prima di dar inizio allo sviluppo, gli specialisti della raccolta dei requisiti possono dare forma al "canale" di comunicazione. Essi devono mediare tra il dominio degli utenti (e degli altri stakeholder) e il mondo tecnico dell'ingegnere del software.

3.1 Fonti dei requisiti

[Dav93; Gog93; Pfl01]

I requisiti di un software tipico derivano da diverse fonti; è importante, quindi, che tutte le potenziali fonti siano identificate e valutate dal punto di vista dell'impatto sul software. Questa sezione è disegnata per promuovere la consapevolezza sull'esistenza delle diverse fonti dei requisiti del software e sui modi di gestione. I punti principali della trattazione sono i seguenti:

- *Goal*. Il termine "goal" (in alcuni casi chiamato "Obiettivo di business", "Business Concern" o "Critical Success Factor (CSF)", si riferisce agli obiettivi generali e di alto livello del software: i "goal" forniscono la motivazione per lo sviluppo del software ma spesso vengono formulati in maniera vaga. I progettisti del software devono prestare particolare attenzione alla stima del valore e al costo dei goal (in relazione alla priorità). Lo studio di fattibilità è un modo, relativamente a basso costo, per raggiungere lo scopo. [Lou95].
- *Conoscenza del dominio*. Il progettista software deve acquisire o avere a propria disposizione la conoscenza del dominio applicativo. Questo gli consente di inferire tacitamente la conoscenza che gli stakeholder non esprimono compiutamente e di stimare i compromessi che saranno necessari tra i requisiti in conflitto e, qualche volta, di agire come un "avvocato difensore" degli utenti.
- *Gli stakeholder* (vedi la sezione *2.2 Attori del processo*). Molte volte il software si dimostra incapace di soddisfare il committente perché enfatizza troppo i requisiti di un unico gruppo di stakeholder a spese di quelli degli altri gruppi. Ciò induce a rilasciare software difficoltoso da utilizzare, oppure tale da sovvertire cultura o strutture politiche dell'organizzazione del cliente. Il progettista del software deve identificare, rappresentare e gestire i "punti di vista" di diversi tipi di stakeholder. [Kot00]
- *L'ambiente operativo*. I requisiti sono ricavati dall'ambiente nel quale il software sarà eseguito. Questi possono consistere, per esempio, in vincoli di tempo nel software real-time, oppure in vincoli di interoperabilità di un ambiente di ufficio, ecc. I requisiti devono essere considerati attentamente perché possono influenzare pesantemente la fattibilità e il costo del software e restringere le scelte di progettazione. [Tha97]
- *L'ambiente organizzativo*. Spesso si richiede al software di supportare un processo di business; in questo caso, la scelta può essere condizionata dalla struttura, dalla cultura e dalla politica interna dell'organizzazione. Il progettista software deve essere sensibile a questi fattori, poiché in generale un nuovo software non deve causare cambiamenti non pianificati sui processi di business.

3.2 Tecniche di elicitazione

[Dav93; Kot00; Lou95; Pfl01]

Una volta identificate le fonti dei requisiti, il progettista può iniziare ad elicitare[2] i requisiti da queste. La sezione è dedicata quindi ai diversi modi di supportare gli stakeholder nella formulazione dei propri requisiti. E' un'area irta di difficoltà e il progettista software deve essere sensibile al fatto che, per esempio, molti utenti possono trovarsi in difficoltà nel descrivere i propri compiti e possono quindi lasciare nel vago una grande quantità d'informazioni, oppure possono essere non invogliati o incapaci di fornire collaborazione. E' particolarmente importante capire che la fase di elicitazione non è un'attività passiva e che, anche quando sono disponibili stakeholder collaborativi e variegati; anche in questo caso, il progettista software deve lavorare duramente per raccogliere la giusta informazione. Esiste una grande varietà di tecniche, le principali delle quali sono: [Gog93]

- *Interviste*: sono il mezzo più tradizionale di elicitazione dei requisiti. E' importante capire vantaggi e limitazioni delle interviste e come esse dovrebbero essere condotte.
- *Scenari*: sono un valido mezzo per fornire un contesto alla fase di elicitazione dei requisiti utente. Gli scenari consentono al progettista software di fornire una cornice alle domande che riguardano i compiti dell'utente e permettono di porre domande del tipo "cosa succede se …" e "come si fa a fare questo?". Il tipo più comune di scenario è la tecnica dei "casi d'uso" (*Use Case*). Vi è qui un legame con la sezione *4.2* (*Modello concettuale*) poiché le notazioni di scenario, come i casi d'uso e i diagrammi, sono usati anche per la modellazione del software.
- *Prototipi*: sono uno strumento molto valido per chiarire i requisiti incompleti, ambigui, poco chiari. Si possono utilizzare come gli scenari per fornire agli utenti un contesto nel quale poter capire meglio quale informazione sia necessaria. Vi sono diversi tipi di tecniche prototipali che vanno dallo "schizzo" su carta delle schermate fino alle versio-

[1] Il tema, cruciale per lo sviluppo del software, è indirizzato dalle metodologie "agili" con un impegno degli utenti e degli sviluppatori a collaborare strettamente durante l'intero progetto (n.d.t.).

[2] elicitare v.tr. [dal lat. *elicere* «tirar fuori»] (*io elìcito, ecc.*) – Treccani.it

ni in beta-test dei prodotti software; esiste una forte sovrapposizione nel modo di utilizzare tali tecniche per l'elicitazione dei requisiti e per la loro validazione (vedi la sezione *6.2 Prototipi*).

- *Riunioni assistite*: l'obiettivo dei meeting assistiti (o "facilitati") è di provare a raggiungere un effetto cumulativo, ogni volta che un gruppo di persone può portare più informazioni sui requisiti del software di quanto non possano fare individualmente. Si possono organizzare sessioni di brainstorming e rifinire le idee difficili da portare in superficie con semplici interviste. Un secondo vantaggio sta nel fatto che i requisiti in conflitto vengono a galla da subito, consentendo agli stakeholder di riconoscere, fin dalle prime fasi, le situazioni di conflitto. Se fatta bene, questa tecnica può ottenere requisiti più coerenti e più dettagliati di quelli che si otterrebbero con altre tecniche. In ogni caso le riunioni devono essere gestite con molta attenzione, (da qui nasce la necessità di un "facilitatore") per prevenire le situazioni nelle quali le capacità critiche del gruppo sono ostacolate da fenomeni di "lealtà di gruppo", oppure quando, i requisiti che riflettono le preoccupazioni di pochi interlocutori più "franchi" (o forse più senior) sono trattati in modo più favorevole a detrimento degli altri.
- *Osservazione*: l'importanza del software nel contesto organizzativo ha richiesto l'introduzione di tecniche di osservazione del comportamento nell'elicitazione dei requisiti. I progettisti software studiano i compiti svolti dagli utenti immergendosi in prima persona nel loro contesto operativo e osservando come essi interagiscono con il loro software e l'uno con l'altro. Queste tecniche sono relativamente dispendiose, ma sono molto istruttive perché rendono evidente ai tanti come molti task dell'utente e processi di business siano complessi perché possano essere descritti agevolmente da chi li esegue.

4. Analisi dei requisiti

[Som05]

Questa sezione è dedicata al processo di analisi dei requisiti che serve a:

- rilevare e risolvere i conflitti tra requisiti;
- scoprire i confini del software e come esso debba operare nel proprio ambiente;
- elaborare i requisiti di sistema per ricavarne quelli del software.

Il modo tradizionale di vedere l'analisi dei requisiti è sempre stata quella di ridurla ad una modellazione concettuale, utilizzando una tra le numerose tecniche di analisi quale, ad esempio, SADT (*Structured Analysis and Design Technique*). Pur essendo importante la modellazione concettuale, anche la classificazione dei requisiti aiuta a dare forma ai compromessi tra i requisiti (classificazione dei requisiti) e a mettere in atto questi compromessi (negoziazione dei requisiti). [Dav93]

Bisogna porre attenzione nel descrivere con sufficiente precisione i requisiti in modo da poter validare i requisiti, validare la loro implementazione e stimare il loro costo.

4.1 Classificazione dei requisiti

[Dav93; Kot00; Som05]

I requisiti possono essere classificati secondo certe dimensioni. Seguono alcuni esempi.

- *Tipologia*: il requisito è funzionale o non funzionale (vedi la sezione *1.3 Requisiti funzionali e non funzionali*).
- *Gerarchia*: il requisito è derivato da uno o più requisiti di livello più alto oppure da una proprietà emergente (vedi la sezione *1.4 Proprietà emergenti*), oppure è stato imposto al software direttamente da uno stakeholder o da qualche altra fonte.
- *Riferimento*: il requisito riguarda il *prodotto* oppure il *processo*. I requisiti relativi al processo, per esempio, possono vincolare la scelta del contraente; possono condizionare la scelta del processo da adottare per la progettazione del software; oppure possono determinare gli standard ai quali aderire.
- *Priorità*: i requisiti hanno una loro priorità. In genere, più alta è la priorità più importante è il requisito per raggiungere gli obiettivi (*goal*) complessivi e di alto livello del software. Spesso la priorità è classificata secondo una scala fissa: "obbligatorio", "altamente desiderabile", "desiderabile" oppure "opzionale". La priorità di un requisito deve essere spesso bilanciata con il costo dello sviluppo e dell'implementazione.
- *Ambito* (*scope*): l'ambito si riferisce all'estensione del software e delle sue componenti influenzate dal requisito. Alcuni requisiti, ed in particolare alcuni di quelli non funzionali, influenzano un ambito globale, in quanto la loro soddisfazione non può essere allocata in una singola componente. Di conseguenza un requisito che riguarda l'ambito globale può influenzare in modo rilevante l'architettura del software e la progettazione di molte sue componenti, mentre un requisito con un ambito più ristretto può offrire una molteplicità di scelte di progettazione e può avere un impatto basso sulla soddisfazione di altri requisiti.
- *Volatilità/stabilità*. Alcuni requisiti cambiano durante il ciclo di vita del software o perfino durante una stessa fase di sviluppo. E' utile dunque, quando possibile, eseguire una stima della probabilità che il requisito possa cambiare. In un'applicazione bancaria, ad esempio, i requisiti riguardanti le funzioni per il calcolo e l'accredito degli interessi sui conti dei clienti sono probabilmente più stabili di un requisito che realizzi un particolare tipo di conto esente da imposte. Il primo riflette una funzione fondamentale del dominio applicativo bancario (cioè che i conti possono maturare interessi), mentre il secondo può essere reso obsoleto da una modifica alla legislazione in vigore. Contrassegnare alcuni requisiti come potenzialmente volatili può aiutare il progettista software a realizzare un progetto con una tolleranza maggiore a fronte dei cambiamenti.

Altri tipi di classificazione possono risultare appropriati in funzione delle pratiche di lavoro dell'organizzazione e dell'applicazione stessa.

Esiste una forte sovrapposizione tra la classificazione dei requisiti e suoi attributi (si veda la sezione *7.3 Attributi dei requisiti*).

4.2 Modellazione concettuale

[Dav93; Kot00; Som05]

Lo sviluppo di modelli per la rappresentazione dei problemi del mondo reale è basilare per l'analisi dei requisiti del software. Lo scopo è di aiutare a comprendere il problema e non a iniziare la progettazione della soluzione da subito. Di conseguenza, quelli concettuali comprendono modelli delle entità appartenenti al dominio del problema, configurati in modo tale da riflettere le loro relazioni e le loro dipendenze nel mondo reale.

Si possono sviluppare diversi tipi di modelli. Tra questi sono compresi i flussi di dati, i flussi di controllo, i modelli di stato, il tracciamento degli eventi, le interazioni degli utenti, i modelli degli oggetti, i modelli dei dati e numerosi altri. I fattori che influenzano la scelta di un modello includono:

- *Natura del problema*. Alcuni tipi di software richiedono che certi aspetti siano analizzati in modo particolarmente rigoroso. Per esempio, i flussi di controllo (*Control Flow*) e i modelli di stato (*Status Diagram*) sono probabilmente più importanti per il software real-time che per il software di gestione delle informazioni, mentre normalmente accade l'opposto per i modelli dei dati (*Data Model*).
- *Esperienza del progettista software*. Spesso è più produttivo adottare un metodo o una notazione per la modellazione con cui il progettista software ha maggiore familiarità.
- *Requisiti di processo del cliente*. I clienti possono imporre i loro metodi e tecniche di notazione, oppure proibire atri con cui hanno minore familiarità. Questo fattore può entrare in conflitto con quello precedente.
- *Disponibilità di metodi e di strumenti*. Notazioni o metodi scarsamente supportati da strumenti e da formazione adeguata possono non raggiungere una larga accettazione, anche quando siano particolarmente adatti per certi tipi di problematiche.

Spesso è utile partire con la costruzione di un modello del contesto software (*Contest Diagram*). Esso fornisce una connessione tra il software che s'intende sviluppare e il suo ambiente esterno. E' un fattore di estrema importanza per comprendere il software nel suo ambiente operativo e identificare le interfacce con l'esterno.

Il tema della modellazione è strettamente connesso a quello dei metodi. Ai fini pratici, un metodo è una notazione (o un insieme di notazioni) supportata da un processo che guida il modo di applicare le notazioni. Non ci sono molte evidenze soggettive che possano sostenere la pretesa superiorità di una notazione rispetto ad un'altra. Comunque, l'accettazione su larga scale di un particolare metodo o di una particolare notazione può portare come beneficio l'aumento del bacino di esperienze e di conoscenze, allargato a più settori di industria. Questa situazione si è verificata, per esempio, per l'UML (*Unified Modeling Language*). [UML04]

La modellazione formale facente uso di notazioni basate sulle matematiche, tracciabili fino al ragionamento logico, ha avuto un impatto in alcuni domini specializzati. Tali notazioni possono essere imposte dai clienti o da standard, oppure possono offrire vantaggi importanti per l'analisi di certe funzioni o componenti critiche.

La presente sezione non ha il fine di "insegnare" un particolare stile di modellazione o di notazione, ma piuttosto quello di fornire una guida sui fini e sugli intenti della modellazione.

Due standard forniscono notazioni utili per la modellazione concettuale: IEEE 1320.1 - IDEF0 - per la modellazione funzionale; IEEE 1320.2 - IDEF1X97 (IDEFOobject) - per la modellazione dell'informazione (*Information Modeling*).

4.3 Il disegno architetturale e l'allocazione dei requisiti

[Dav93; Som05]

Giunti a un certo punto del progetto, l'architettura della soluzione deve essere esplicitata. La progettazione architetturale è il punto nel quale il processo di raccolta dei requisiti si sovrappone alla progettazione del software e dei sistemi e costituisce la dimostrazione di quanto sia impossibile disaccoppiare nettamente i due task. [Som01] Questa sezione è strettamente legata alla sottoarea *Struttura del software e architettura* nella Knowledge Area *Software Design*. In molti casi, il progettista software agisce come un architetto software poiché il processo di analisi ed elaborazione dei requisiti richiede che siano identificate le componenti responsabili di soddisfare ciascun richiesta. Questo è quanto fa l'allocazione dei requisiti, cioè l'assegnazione alle componenti della responsabilità di soddisfare le richieste.

L'allocazione è importante per assicurare un'analisi dettagliata dei requisiti. Ad esempio, una volta che un insieme di requisiti sia stato allocato ad una componente, le singole esigenze possono essere ulteriormente analizzate per scoprire nuovi requisiti sul come le componenti debbano interagire tra loro per soddisfare i requisiti allocati. Nei progetti di grandi dimensioni l'allocazione è di stimolo per un nuovo ciclo di analisi di ogni sottosistema. Per esempio, nel campo automobilistico, i requisiti relativi ad una frenata di particolare efficacia (lunghezza di frenata, sicurezza in cattive condizioni della strada, uniformità della frenata, pressione richiesta sul pedale, ecc.) possono essere allocati all'hardware del sistema frenante (strutture meccaniche e idrauliche) e ad un sistema anti-bloccaggio (ABS). Solo quando un requisito per un sistema anti-bloccaggio sia stato identificato e i relativi requisiti siano stati allocati ad esso, le capacità

dell'ABS e dell'hardware del sistema frenante e le proprietà emergenti (quali il peso della vettura) possono essere usate per identificare i requisiti di dettaglio del software per l'ABS.

La progettazione dell'architettura (*Architectural Design*) s'identifica strettamente con la modellazione concettuale. La mappatura delle entità di dominio del mondo reale sulle componenti software non è sempre ovvia; tant'è che la progettazione dell'architettura viene considerata come un argomento a sé stante. I requisiti delle notazioni e dei metodi sono largamente gli stessi sia per la modellazione concettuale sia per la progettazione dell'architettura.

Lo Standard IEEE 1471-2000, *Recommended Practice for Architectural Description of Software Intensive Systems*, suggerisce un approccio (a partire da una pluralità di punti di vista) alla descrizione di un'architettura di sistemi e delle loro componenti software. (IEEE 1471-00)

4.4 La negoziazione dei requisiti

Un altro termine comunemente utilizzato per questo tema è "risoluzione dei conflitti". Il termine indica la risoluzione dei problemi riguardanti i requisiti (esempio, conflitti tra due stakeholder che richiedono caratteristiche incompatibili tra di loro, oppure conflitti tra i requisiti e le risorse disponibili, o tra requisiti funzionali e non funzionali). [Kot00, Som97] In molti casi non è prudente da parte del progettista prendere decisioni unilaterali; diventa necessario consultarsi con lo stakeholder (o gli stakeholder) per raggiungere un consenso o un compromesso. E' spesso importante, per ragioni contrattuali, che queste decisioni siano tracciabili all'indietro fino a risalire alla fonte (cliente). Abbiamo classificato questo argomento tra gli elementi dell'analisi dei requisiti software, poiché i problemi sorgono proprio come risultato dell'attività di analisi. Vi sono però anche buone motivazioni per considerarlo un argomento da trattare nell'ambito delle attività di validazione dei requisiti.

5. Specifica dei requisiti

Per molte professioni del settore ingegneristico, il termine "specifica" si riferisce all'assegnazione di valori numerici o di limiti agli obiettivi di progettazione di un prodotto. [Vin90]. Tipicamente, i sistemi fisici presentano un numero relativamente piccolo di tali valori. Il software, invece, ha normalmente un gran numero di requisiti e l'attenzione si divide tra la quantificazione numerica e la gestione della complessità dell'interazione tra questo grande numero di requisiti. Così, nel gergo del software engineering, "specifica dei requisiti del software" si riferisce di solito alla produzione di un documento, o del suo equivalente elettronico, che possa essere sistematicamente sottoposto a revisione, valutato e approvato. Per i sistemi complessi, ed in particolare quelli che comprendono importanti componenti non-software, sono prodotti non meno di tre differenti tipi di documento: Definizione del sistema (*System Definition*), Requisiti di sistema (*System Requirements*) e Requisiti software (*Software Requirements*). Per i prodotti software più semplici, è richiesto solo il terzo di questi documenti. Tutti e tre i documenti sono descritti nel seguito, con l'avvertenza che essi possono essere combinati di volta in volta nel modo più appropriato. Una descrizione dell'ingegneria dei sistemi può essere reperita nel *Capitolo 12 Le discipline correlate all'ingegneria del software*.

5.1 Documento di definizione del sistema (System Definition)

Questo documento (noto anche come "documento dei requisiti utente" o "Concept of Operations") raccoglie i requisiti di sistema. I requisiti di sistema sono definiti ad alto livello dal punto di vista del dominio applicativo. Il documento è destinato ai rappresentanti degli utenti/clienti del sistema (il marketing può interpretare il ruolo dei clienti per il software destinato al mercato, quello detto "market-driven"), così che il suo contenuto possa essere espresso nei termini del dominio applicativo. Esso elenca quindi i requisiti di sistema insieme alle informazioni di base che riguardano gli obiettivi generali del sistema, il suo ambiente di destinazione di riferimento e un'enunciazione dei vincoli, delle assunzioni e dei requisiti non funzionali. La definizione dei requisiti può includere modelli concettuali progettati per illustrare il contesto di sistema, gli scenari di utilizzo e le principali entità del dominio applicativo, così come dati, informazioni e flussi di processo. Lo standard IEEE 1362, *Concept of Operations Document*, offre suggerimenti per la predisposizione e sui contenuti di questo documento. [IEEE1362-98]

5.2 Specifica dei requisiti di sistema

[Dav93; Kot00; Rob99; Tha97]

Gli sviluppatori dei sistemi che comprendono parti importanti sia di software sia non-software (per esempio, un aviogetto moderno) spesso separano la descrizione del sistema da quella del software. In quest'ottica sono prima specificati i requisiti di sistema; i requisiti del software sono ricavati da quelli di sistema; in ultimo, sono espressi i requisiti per le componenti software. In senso stretto, la specifica dei requisiti di sistema è un'attività d'ingegneria dei sistemi e ricade quindi al di fuori dell'ambito di questa Guida. Lo standard IEEE 1233 è una guida allo sviluppo dei requisiti di sistema. [IEEE1233-98]

5.3 Specifica dei requisiti del software

[Kot00; Rob99]

La specifica dei requisiti software stabilisce la base di un accordo tra clienti e contraenti o fornitori (nei progetti market-driven questi ruoli possono essere interpretati dalle divisioni marketing e sviluppo) circa quello che il prodotto software deve fare e quello che non è richiesto che faccia. Per i lettori non tecnici, il documento di spe-

cifica dei requisiti software è spesso accompagnato da un documento di "definizione dei requisiti software".

La specifica dei requisiti consente una valutazione rigorosa delle esigenze, condotta prima che possa iniziare la progettazione, e riduce il rischio di una successiva riprogettazione. Inoltre, fornisce una base realistica per la stima dei costi del prodotto, dei rischi e dei tempi.

Le organizzazioni possono utilizzare il documento di specifiche dei requisiti software anche per sviluppare in maniera più produttiva i propri piani di validazione e di verifica.

La specifica dei requisiti software fornisce una base informativa per il trasferimento di un prodotto software a nuovi utenti o su nuove macchine. Infine, essa fornisce una base anche per il miglioramento del software.

I requisiti software sono spesso espressi in linguaggio naturale, ma, nel documento di specifica, possono essere corredati da descrizioni formali o semi-formali.

Una selezione di notazioni appropriate consente di descrivere con maggiore precisione e concisione requisiti e aspetti particolari dell'architettura software, rispetto al linguaggio naturale. La regola è di utilizzare notazioni che consentano una descrizione dei requisiti più accurata possibile. Ciò è particolarmente importante per il software critico per la sicurezza e per altri tipi con requisiti di massima affidabilità. In ogni caso, la scelta della notazione è spesso vincolata alla formazione, all'esperienza e alle preferenze degli autori e dei lettori.

Sono stati sviluppati molti indicatori per mettere in relazione la qualità delle specifiche dei requisiti software con altre variabili di progetto, come il costo, l'accettazione, le prestazioni (performance), la tempistica, la riproducibilità ecc. Gli indicatori di qualità per i singoli statement delle specifiche comprendono imperativi, direttive, frasi deboli (*Weak Phrases*), opzioni, rinvii (*Continuances*). Gli indicatori riguardanti il documento di specifiche dei requisiti software comprendono le sue dimensioni, la leggibilità, le specifiche, il grado di approfondimento e la struttura del testo. [Dav93; Tha97] [Ros98]

Lo IEEE ha un proprio standard, lo IEEE 830 [IEEE830-98], riguardante la produzione e il contenuto delle specifiche dei requisiti software. Anche lo IEEE 1465 (simile all'ISO/IEC 12119) è uno standard che tratta i requisiti di qualità nei pacchetti software (*Software Packages*). [IEEE1465-98]

6. Validazione dei requisiti

[Dav93]

I documenti riguardanti i requisiti possono essere soggetti a procedure di validazione e di verifica. I requisiti possono essere validati per assicurarsi che il progettista abbia compreso i requisiti. E' anche importante verificare che un documento di requisiti sia conforme ai modelli delle società che non utilizzano software, come ad esempio, i modelli realizzati con lavagne a fogli mobili. I prototipi possono essere costosi da sviluppare. Comunque, se consentono di evitare lo spreco di risorse causato dai tentativi di soddisfare requisiti erronei, il loro costo può essere giustificato più facilmente.

6.1 Validazione del modello

[Dav93; Kot00; Tha97]

Solitamente è necessario validare la qualità dei modelli che sono stati sviluppati in fase di analisi. Ad esempio, nei modelli a oggetti, è utile eseguire un'analisi statica per verificare che esistano modi di comunicazione tra gli oggetti che, nel dominio applicativo dello stakeholder, si scambiano dati. Se sono utilizzate notazioni formali per le specifiche, è possibile avvalersi delle tecniche del ragionamento formale per sottoporre a dimostrazione le proprietà delle specifiche.

6.2 Test di accettazione

[Dav93]

Una proprietà essenziale di un requisito software consiste nel rendere possibile la verifica che il prodotto finito lo soddisfi. I requisiti che non possono essere validati sono soltanto "desideri". Un compito importante è quindi la pianificazione delle modalità con cui verificare ciascun requisito. In molti casi la progettazione dei test di accettazione ha proprio questo obiettivo.

L'identificazione e la progettazione dei test di accettazione può rivelarsi difficoltosa per i requisiti non funzionali (si veda la sezione *1.3 Requisiti funzionali e non funzionali*). Per essere validati questi requisiti devono in primo luogo essere analizzati fino al punto di poterli esprimere anche quantitativamente.

Informazioni aggiuntive possono essere trovate nel capitolo sulla Knowledge Area *Software Testing*, nella sezione *2.2.4 Test di conformità*.

7. Considerazioni pratiche

Il primo livello di scomposizione delle sotto-aree che è stato presentato in questa Knowledge Area può far sembrare che si tratti di una sequenza lineare di attività. Se così fosse, si tratterebbe di un modo troppo semplificato di vedere il processo. [Dav93]

Il processo dei requisiti copre l'intero ciclo di vita del software. Il processo di *Change Management* e l'attività di mantenere i requisiti in uno stato che rifletta accuratamente il software che deve essere sviluppato, o che è stato assemblato, sono attività fondamentali per il buon esito del processo di ingegneria del software. [Kot00; Lou95]

Non tutte le organizzazioni hanno una cultura sul come documentare e gestire i requisiti. E' frequente nelle società start-up, che sono più dinamiche, caratterizzate da un'altrettanta "visione del prodotto" e da risorse limitate, constatare che la documentazione dei requisiti sia vi-

sta come un sovraccarico non necessario. Il più delle volte, comunque, man mano che l'organizzazione cresce con la loro base di clienti, il loro prodotto comincia ad evolversi e scoprono di aver bisogno di recuperare i requisiti iniziali che hanno motivato all'origine le caratteristiche del loro prodotto per valutare l'impatto dei cambiamenti proposti. Perciò, la documentazione dei requisiti e la gestione delle modifiche sono fattori chiave per assicurare il successo di un qualsiasi processo di raccolta dei requisiti.

7.1 La natura iterativa del processo di raccolta dei requisiti

[Kot00; You01]

Si assiste a una pressione generale da parte dell'industria del software per disporre di cicli di sviluppo sempre più brevi; questo fenomeno è particolarmente acuto nei settori altamente competitivi del software cosiddetto market-driven. Inoltre molti progetti sono vincolati in qualche maniera dal loro ambiente e molti di essi consistono in adattamenti o revisioni di software esistente per il quale l'architettura è un dato di fatto. In pratica, quindi, si rivela quasi sempre poco pratica l'implementazione di un processo dei requisiti lineare e deterministico nel quale i requisiti software siano elicitati dagli stakeholder, congelati, allocati e consegnati al team di sviluppo. E' un "mito" considerare che i requisiti di un progetto di grandi dimensioni siano sempre perfettamente compresi o perfettamente specificati. [Som97]

Al contrario, i requisiti sono tipicamente sottoposti ad un processo iterativo per giungere ad un livello di qualità e di dettaglio che permetta di eseguire la progettazione e prendere decisioni circa l'acquisto dall'esterno. In alcuni progetti questo può dare origine a requisiti che sono congelati prima che tutte le loro proprietà siano state pienamente comprese. Questo fatto è causa di costose rilavorazioni se i problemi emergono tardi nel processo d'ingegnerizzazione del software. In ogni caso, i progettisti software sono necessariamente costretti a rispettare i piani di progetto e devono di conseguenza prendere provvedimenti per assicurarsi che la "qualità" dei requisiti sia la più alta possibile a fronte delle risorse disponibili. Essi dovrebbero, ad esempio, rendere esplicite tutte le assunzioni che sono alla base dei requisiti, così come tutti i problemi noti.

Nella maggior parte dei casi, la comprensione dei requisiti continua ad evolvere insieme al procedere della progettazione e dello sviluppo. Ciò richiede spesso una revisione dei requisiti in una fase più tarda del ciclo di vita. Forse, il punto più delicato nel capire l'ingegneria dei requisiti sta nel fatto che una porzione rilevante dei requisiti è destinata a cambiare. In alcuni casi ciò è dovuto ad errori di analisi, ma molto spesso è una conseguenza inevitabile del cambiamento del "contesto". Ad esempio, può cambiare l'ambiente operativo del cliente, il business, il mercato nel quale il software deve essere venduto. Qualunque sia la causa, è importante riconoscere l'inevitabilità del cambiamento e prendere provvedimenti per mitigarne gli effetti. Il cambiamento deve essere gestito assicurandosi che i cambiamenti proposti passino attraverso un processo definito di verifica e di approvazione e applicando tecniche accurate di tracciamento dell'analisi d'impatto e di tecniche di gestione della configurazione software (si veda la Knowledge Area *Software Configuration Management*). Si capisce, dunque, che il processo dei requisiti non è semplicemente un task iniziale dello sviluppo del software, ma copre l'intero ciclo di vita. In un tipico progetto di sviluppo, le attività di raccolta dei requisiti evolvono nel tempo dalla loro elicitazione fino alla gestione delle modifiche.

7.2 Change Management

[Kot00]

La gestione delle modifiche (*Change Management*) riveste un'importanza centrale nella gestione dei requisiti. La sezione descrive il ruolo del change management, le procedure da mettere in opera e l'analisi da condurre sui cambiamenti proposti. Essa ha forti correlazioni con la Knowledge Area *Software Configuration Management*.

7.3 Attributi dei requisiti

[Kot00]

I requisiti non dovrebbero consistere solo di specifiche su ciò che è richiesto, ma anche di tutte le informazioni ausiliarie che aiutano a gestire e a interpretare i requisiti. Questo dovrebbe includere le diverse classificazioni del requisito (si veda la sezione *4.1 Classificazione dei requisiti*), il metodo di verifica e il piano dei test di accettazione. Può anche includere informazioni aggiuntive come, ad esempio, una sintesi riassuntiva di ciascun requisito, la fonte e una storia dei cambiamenti subiti. L'attributo più importante rimane comunque il codice che consente ai requisiti di essere identificati in modo univoco e non ambiguo.

7.4 Tracciamento dei requisiti

[Kot00]

Il tracciamento dei requisiti riguarda il recupero della fonte da cui sono tratti e la capacità di poter prevedere i loro effetti. Il tracciamento è fondamentale per eseguire l'analisi d'impatto quando un requisito cambia. Un requisito dovrebbe essere tracciabile all'indietro fino ai requisiti di livello superiore e allo stakeholder che ne ha motivata la necessità (es.: da un requisito software si torna indietro fino al requisito di sistema che esso contribuisce a soddisfare). Al contrario, un requisito dovrebbe poter essere tracciato in avanti fino agli altri requisiti e alle entità software che lo soddisfano (ad esempio, da un requisito di sistema fino ai requisiti software che sono stati elaborati da esso e fino ai moduli di codice che lo implementano).

Il tracciamento dei requisiti per un tipico progetto software formerà un complesso grafo di requisiti, diretto e aciclico (*Direct Acyclic Graph* - DAG)

7.5 Misurazione dei requisiti

Nella pratica quotidiana è utile avere qualche concetto sul "volume" dei requisiti per un particolare prodotto software. Questo numero è utile per valutare le "dimensioni" di un cambiamento di requisiti, nello stimare il costo di uno sviluppo o di un intervento di manutenzione o semplicemente per un suo utilizzo come denominatore in altre misure. Il *Functional Size Measurement* (FSM) è una tecnica per la valutazione delle dimensioni di un insieme di requisiti funzionali. Lo standard IEEE 14143.1 definisce il concetto di FSM. [IEEE 14143.1-00].

Altri standard ISO/IEC e altre fonti descrivono nuovi metodi di FSM.

Un'informazione aggiuntiva sulla misura delle dimensioni e sugli standard può essere recuperata nella Knowledge Area *Software Engineering Process*.

RIFERIMENTI RACCOMANDATI PER IL SOFTWARE REQUIREMENTS

[Dav93] A.M. Davis, *Software Requirements: Objects, Functions and States*, Prentice Hall, 1993.

[Gog93] J. Goguen and C. Linde, *"Techniques for Requirements Elicitation," presented at International Symposium on Requirements Engineering*, 1993.

[IEEE830-98] IEEE Std 830-1998, IEEE, *Recommended Practice for Software Requirements Specifications*, IEEE, 1998.

[IEEE14143.1-00] IEEE Std 14143.1-2000//ISO/IEC14143-1:1998, *Information Technology—Software Measurement—Functional Size Measurement—Part 1: Definitions of Concepts*, IEEE, 2000.

[Kot00] G. Kotonya and I. Sommerville, *Requirements Engineering: Processes and Techniques*, John Wiley & Sons, 2000.

[Lou95] P. Loucopulos and V. Karakostas, *Systems Requirements Engineering*, McGraw-Hill, 1995.

[Pfl01] S.L. Pfleeger, *"Software Engineering: Theory and Practice"*, second ed., Prentice Hall, 2001, Chap. 4.
[Rob99] S. Robertson and J. Robertson, Mastering the Requirements Process, Addison-Wesley, 1999.

[Som97] I. Sommerville and P. Sawyer, *Requirements Engineering: A Good Practice Guide*, John Wiley & Sons, 1997, Chap. 1-2.

[Som05] I. Sommerville, *Software Engineering*, seventh ed., Addison-Wesley, 2005.

[Tha97] R.H. Thayer and M. Dorfman, eds., *Software Requirements Engineering*, IEEE Computer Society Press, 1997, pp. 176-205, 389-404.

[You01] R.R. You, *Effective Requirements Practices*, Addison-Wesley, 2001.

BIBLIOGRAFIA

Di seguito è riportata la bibliografia essenziale sui temi generali del Software Engineering e del Requirements Engineering in particolare.

[1] IEEE *Standard Glossary of Software Engineering Terminology*, 1990.

[2] *Guide to the Software Engineering Body of Knowledge, SWEBOK, A Project of the Software Engineering Coordinating Committee*. IEEE-Version 2 (2004).

[3] Sommerville, Ian. *Ingegneria del Software*. Addison Wesley, 7ª Edizione (2005). pp. 11-136.

[4] Pressman, Roger S. *Principi di ingegneria del software.* McGram-Hill (2005). pp. 159-194.

[5] Ghezzi, Jazayeri, Mandrioli. *Ingegneria del software*. Prentice Hall 2ª Edizione (2004). pp 177- 291.

[6] Binato, Fuggetta, Sfardini. *Ingegneria del software.* Addison Wesley (2006). pp 67-108.

[7] Arlow J., Neustadt, I. *UML 2 e Unfied Process.* McGraw-Hill (2006). pp 45-121.

[8] Zuser, W., Biffl, S., Kohle, M. *Ingegnria del software con UML e Unified Process.* McGraw-Hill (2004). pp 69-105.

[9] Kruchten, Philippe. *Rational Unified Process - Introduzione.* Addison-Wesley (2000). pp 161-174.

[10] Colonese, Ercole. *Gestione dei requisiti*. www.colonese.it/pubblicazioni/. (2007).

REVISIONI

Il documento è stato sottoposto a revisione interna non formale.

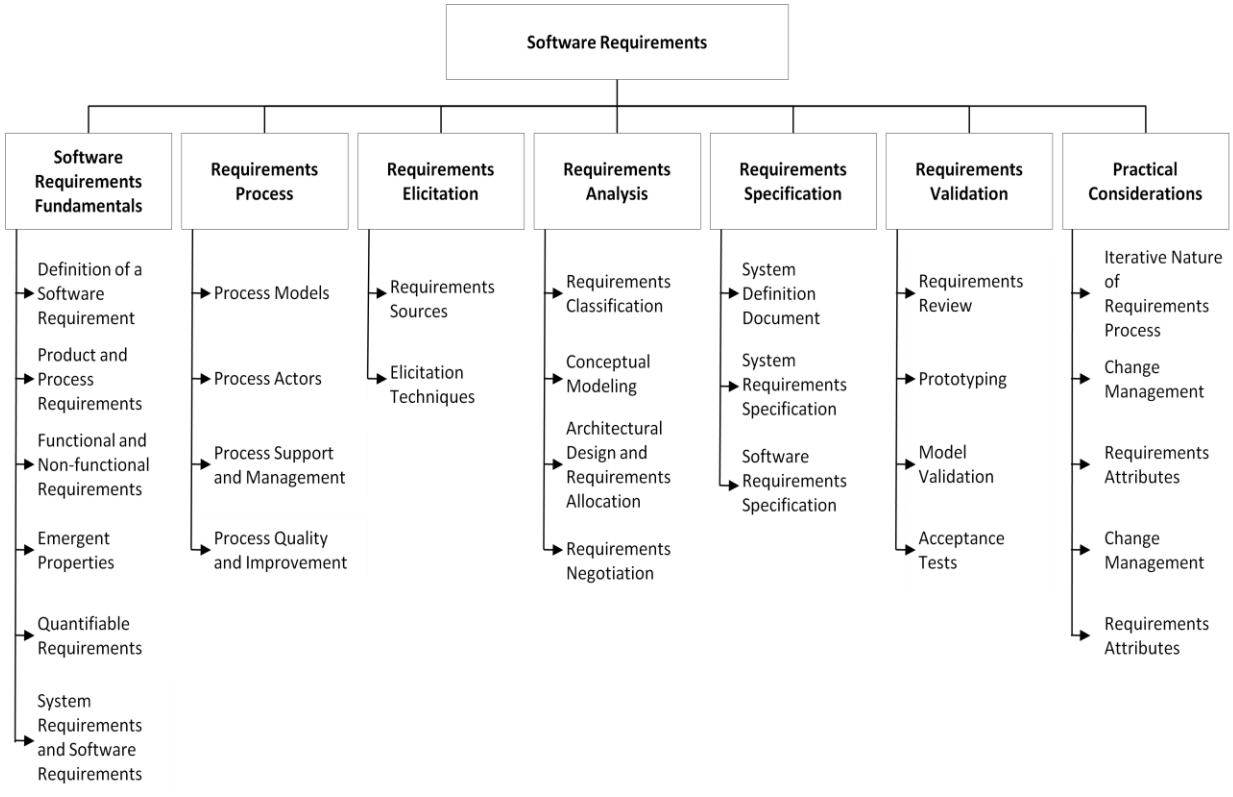

Figura 1 Scomposizione degli argomenti dei Requisiti software