Politecnico di Milano
Scuola di Ingegneria Industriale e dell'Informazione
Corso di Laurea Magistrale in Ingegneria Informatica
Dipartimento di Elettronica, Informazione e Bioingegneria

Titolo

Una metodologia basata su logica fuzzy per la valutazione dei progetti informatici

Relatore: Ing. Cinzia Cappiello

Tesi di laurea di:
Mattia Depoli Matr. 783159

Anno Accademico 2013–2014

Alla mia famiglia...

Ringraziamenti

In conclusione a questi cinque anni di studi di ingegneria informatica presso il Politecnico di Milano vorrei ringraziare le persone che mi sono state accanto in questi anni e che hanno creduto in me.

Vorrei ringraziare innanzitutto la mia famiglia: mia mamma, mio papà e mia sorella per essermi stati sempre vicino, per avermi permesso di intraprendere questi studi universitari senza mai farmi mancare nulla e per avermi supportato e compreso soprattutto durante tutte le mie ansie nei momenti difficili di studio.

Vorrei ringraziare la mia relatrice Ing. Cinzia Cappiello per la disponibilità e la professionalità dimostratami nei mesi che ho passato a redigere la tesi.

Vorrei ringraziare inoltre l'azienda Zucchetti S.p.A per avermi dato l'opportunità di entrare nel mondo del lavoro concedendomi l'occasione di poter esprimere al meglio tutte le nozioni che ho acquisito durante questi anni universitari.

Sommario

La qualità di un software può essere misurata in modo oggettivo o in modo soggettivo attraverso delle metriche che dovrebbero essere definite non solo per il prodotto finale ma anche per tutti i processi intermedi all'interno del ciclo di vita. Pertanto, se si vuole prendere in considerazione la qualità di un prodotto, si devono applicare delle metriche adeguate alle specifiche dei requisiti, al progetto architetturale ed al codice.

Il problema è quello di avere a disposizione una metodologia applicabile alla fase di valutazione sia ex-ante (nella fase iniziale di un progetto) sia ex-post (nella verifica dei risultati finali) di un progetto con l'obiettivo di ottenere dei risultati oggettivi riguardo il prodotto sviluppato.

Lo scopo di questa tesi è proprio quello di definire con oggettività un meccanismo basato su logica fuzzy che utilizza delle metriche standard utili a tracciare una valutazione ed un monitoraggio del prodotto o del prototipo sviluppato.

L'attività svolta in questa tesi è stata quella di stabilire delle guideline utili e un framework che permetta a chiunque di stabilire quanto sia di qualità il proprio lavoro svolto.

Verranno proposti nuovi spunti di riflessione inerenti l'attività di valutazione con le neural fuzzy network.

I risultati ottenuti in seguito a questo sviluppo e le riflessioni riportate in conclusione potranno essere di spunto per dei lavori futuri in questo campo di applicazione.

Abstract

The software quality could be gauged in an objective or subjective way through metrics which could be considered not only for the final product but also for all intermediate processes which characterise the project lifecycle. Thus, if you want to consider the product quality, you must use adequate metrics for specific features, architectural project and code.

The problem is to have a methodology to use during the evaluation phase both ex-ante (during the initial phase of a project) and ex-post (during the final results examination) of a project with the aim to obtain objective results about developed product.

The aim of this thesis is to define, in an objective way, a mechanism based on a fuzzy logic that use standard metrics suitable to do an evaluation and a monitoring of the product or prototype developed.

This thesis aims to establish suitable guidelines and a framework which allow to anybody to establish the quality of her/his own work.

New remarks regarding the evaluation activity with fuzzy logic will be proposed.

Indice

Elenco delle figure

Elenco delle tabelle

Capitolo 1

Introduzione

La gestione di un progetto software include la progettazione, la pianificazione, l'allocazione delle risorse e la gestione dei cambiamenti all'interno di un processo tecnologico.

Il project management permette alle aziende di controllare i costi e i budget gestendo la qualità di un prodotto o di un prototipo.

Un prodotto o un prototipo si può definire «di qualità» se soddisfa una serie di requisiti con lo scopo di misurare o di prevedere i tempi di consegna, i costi di sviluppo, l'effort impiegato e le caratteristiche del prodotto.

«Misurare» vuol dire mappare un insieme di oggetti in un altro insieme di oggetti (numeri o simboli); in altre parole, per misurare un software è necessario stabilire delle metriche che permettano di mappare una serie di caratteritiche e di valutare oggettivamente il software stesso.

Vi sono diverse fasi del ciclo di vita che richiedono una valutazione delle caratteristiche del software, ad esempio:

1. la fase di progettazione: per prevedere la facilità di eseguire interventi di manutenzione dopo il rilascio, per valutare l'impatto ed il costo che avranno nel contesto di utilizzo, per prevenire problemi nel software rilasciato;

2. la fase di testing: per controllare di essere allineati con quanto pianificato e per scoprire problemi non considerati in fase di progettazione;

3. la fase di deployment: per controllare la relativa soddisfazione dei clienti.

Lo scopo della tesi è quello di tracciare una metodologia applicabile alla fase di valutazione di un progetto sia in fase di testing di piccole funzionalità sia in fase di deployment con l'obiettivo di avere a disposizione dei risultati oggettivi riguardo il prodotto sviluppato (anche solo a livello prototipale).

Questa metodologia può essere utilizzata per la valutazione di un prodotto nuovo o di un prototipo del prodotto stesso; il meccanismo presentato quindi può essere

applicato all'interno di un progetto condotto sia attraverso uno schema a cascata (dall'inizio alla fine del prodotto) sia a spirale (ovvero in qualsiasi punto dello sviluppo, in fase prototipale).

Il meccanismo di valutazione presentato in questa tesi si basa principalmente su logica fuzzy che, tramite la loro caratteristica di incertezza, sono in grado di fornire una valutazione completa ed efficace basata principalmente sulla memorizzazione e sulla reazione agli stimoli (tipici del cervello umano).

Infatti, all'interno di un processo di sviluppo è molto utile (se non indispensabile) porsi la seguente domanda: «come sta procedendo il lavoro?» ovvero, data l'elevata incertezza e i continui cambiamenti in ambito tecnologico, si è veramente sicuri di soddisfare dei requisiti oggettivi di qualità del prodotto?

La valutazione è quella fase che deve stimolare la crescita ed il miglioramento ed ad un perfezionamento delle pratiche [21] affinchè si possa giudicare se un programma è «di qualità».

Si può definire «di qualità» un programma se, e solo se, i suoi obiettivi sono stati raggiunti («goal based») oppure se ha soddisfatto gli standard di qualità prefissati («goal free»); di coseguenza il valore dell'obiettivo e la stilazione di standard sono parti fondamentali in un PM per essere in grado di realizzare un prodotto ottimo (o ottimale).

L'idea che ha portato allo sviluppo di questa metodologia è nata proprio in risposta alla domanda posta precedentemente, partendo dall'analisi di cosa esiste già nella letteratura della valutazione dei progetti e cercando di compensare ciò che si ha come limite in altre tecniche.

A questa domanda si possono trarre diverse conclusioni sotto diversi punti di vista; trovare una risposta completa ed oggettiva è il compito che questa tesi si propone di compiere.

Partendo da standard di modellizzazione (come i* modeling e feature model) uniti a metriche già collaudate o provenienti da approcci di metrica del software (come il GQM), si è cercato di definire delle guideline di valutazione attraverso la costruzione di fuzzy rule che permettano ai fornitori di un servizio software di intraprendere eventuali decisioni future inerenti un prodotto o un prototipo.

La metodologia Neural Fuzzy Netwok (NFN) presentata in questo elaborato è stata applicata al caso di Zucchetti (più precisamente attraverso il prodotto ZTimesheet), una realtà aziendale leader del settore ICT italiano (e non solo).

Le guideline presentate, utili alla costruzione del framework, permettono di arrivare ad una valutazione efficace indipendentemente dalle caratteristiche del fornitore stesso del servizio; il protagonista in questa metodologia NFN è «l'esperienza» e non il livello di competività aziendale.

Il concetto di qualità è un concetto ampio che può variare a seconda dei punti di vista; nel mondo ICT per esempio ci si può trovare di fronte ad un concetto di

qualità secondo i customer e secondo i fornitori.

Non è detto che ciò che è ottimo per il fornitore lo sia anche per il customer, viceversa non è detto che ciò che è ottimo per il customer lo sia anche per il fornitore del servizio: tipicamente si tende ad affermare che se «il customer è soddisfatto, anche il fornitore è soddisfatto», ma ci si può chiedere quanto il fornitore sia soddisfatto del proprio lavoro e se si possano avere dei margini di miglioramento.

Attraverso il confronto con altri meccanismi di valutazione, come per esempio il Quality Function Deployment (QFD), si faranno dei confronti e delle riflessioni in merito a questo argomento, cercando di capire se serve maggiormente valorizzare solo il cliente o se è importante valorizzare anche il fornitore in relazione al cliente.

Si noteranno delle discrepanze e dei limiti tra QFD e NFN, che saranno poi spunto di riflessione per un lavoro futuro e per un eventuale miglioria di questo meccanismo.

Questo elaborato farà capire come non sia semplice definire la fase di valutazione all'interno di un processo e quanto è fondamentale attuarla in maniera precisa ed oggettiva considerando anche la soggettività intrinseca che appartiene a questa fase.

La tesi sarà strutturata nel modo seguente:

1. Capitolo 1: parte introduttiva della tesi dove viene illustrata l'area generale in cui si svolge il lavoro e l'attività sperimentale svolta illustrando lo scopo che ha portato alla realizzazione della tesi.

2. Capitolo 2: dove verrà presentato lo stato dell'arte, ovvero ciò che si ha già in letteratura e gli strumenti che si hanno a disposizione utili alla nuova metodologia.

3. Capitolo 3: dove verrà presentata la metodologia NFN per il processo di valutazione mostrando passo dopo passo il flusso informativo, la definizione di tutte le metriche utilizzate e la costruzione delle fuzzy rule.

4. Capitolo 4: dove verrà sottoposta la metodologia NFN al caso di studio proposto, ovvero ZTimesheet. In questo capitolo si seguiranno i meccanismi presentati nel capitolo 2 e si arriverà al processo di valutazione.

5. Capitolo 5: dove verranno raccolti i risultati ottenuti e si procederà con il vero e proprio processo di valutazione del prodotto.

6. Capitolo 6: conclusioni della tesi dove viene fatto un riepilogo del lavoro svolto e si mostrano le prospettive future di ricerca nell'area dove si è svolto il lavoro.

7. Appendice A: dove verrà presentata una proposta di come realizzare un tool automatico che implementi il meccanismo NFN di questa tesi.

Capitolo 2

Stato dell'arte

2.1 Project Management

2.1.1 Note storiche e definizione

L'attività del Project Management nacque moltissimi anni fa, si presume addirittura nel periodo di realizzazione delle grandi costruzioni dell'antichità, soprattuttto con le piramidi (XXVI secolo a.C.) e le grandi opere degli antichi romani (acquedotti, strade, ponti).

Già in quegli anni si conducevano attività di Project Management non utilizzando però presumibilmente tecniche di programmazione e di rappresentazione di tipo scientifico del processo produttivo.

I primi sviluppi scientifici del Project Management si sono avuti infatti moltissimi anni dopo con gli studi di F.Taylor (1865-1915) e di H.Gantt (1861-1919) il quale proponevano l'esistenza di una "one-best way" per compiere una qualsiasi operazione utilizzando metodi scientifici e introducendo quello che oggi è definito come il "diagramma di Gantt". I loro studi hanno contribuito all'evoluzione del management in varie funzioni ben distinte in diversi ambiti: dall'economia al sociale, dall'edilizia alle grandi industrie fino arrivare al XX secolo nel mondo dell'informatica.

Nel corso degli anni ci sono state svariate definizioni di Project Management:

"Per Project Management si intende l'applicazione dell'approccio sistemico alla gestione di attività tecnologicamente complesse o di progetti i cui obiettivi sono esplicitamente fissati in termini di parametri di tempo, costo e performance" (Cleland & King 1988)

"Pianificare, organizzare, dirigere e controllare le risorse dell'azienda per un obiettivo relativamente di breve termine, che è stato fissato per portare a termine traguardi ed obiettivi specifici. Inoltre il Project Management utilizza l'approccio sistemico

alla gestione, mediante l'assegnazione di personale di funzione (gerarchia verticale) ad uno specifico progetto (gerarchia orizzontale)" (Kerzner 1989)

"Il Project Management è un insieme di persone e di altre risorse temporaneamente riunite per raggiungere uno specifico obbiettivo, di solito con un budget determinato ed entro un periodo stabilito" (Graham, University of Chicago, 1990)"

"Il Project Management è un sistema gestionale orientato ai risultati" (Miscia, Italia, 1994)

"Un insieme di attività tra loro correlate e interdipendenti, volte al raggiungimento di un obiettivo preciso, con un limite di tempo determinato, un budget di risorse stabilite, che vengono avviate alla ricerca di un aumento di valore per l'azienda o per il soddisfacimento delle esigenze del cliente" (SDA Bocconi - Scuola di Direzione aziendale – Div. Ricerche 1999)

"Il Project Management è l'applicazione di conoscenze, attitudini, tecniche e strumenti alle attività di un progetto al fine di conseguirne gli obiettivi" (IEEE Standard Association,2011) [34]

Raggruppando tutte queste definizioni, si può definire il Project Management come l' abilità, gli strumenti, le tecniche e i processi per:

1. definire il progetto determinandone la visione globale, gli obiettivi, le responsabilità e i risultati;

2. pianificare e organizzare il lavoro, ovvero determinare i passi necessari all'esecuzione del progetto, assegnare le responsabilità ed identificare le date di inizio e fine, la pianificazione delle attività, nonché lo sviluppo di metodi di valutazione;

3. "Control&Monitoring" ovvero accertarsi in modo continuativo che il progetto stia raggiungendo i suoi obiettivi nei tempi e nei costi previsti;

4. chiudere il progetto, in modo efficiente ed efficace, per la realizzazione di statistiche e la preparazione della documentazione relativa a ciò che si è appreso durante la sua esecuzione;

I concetti chiave del Project Management (PM) sono quindi: definire un obiettivo, definire le risorse disponibili/necessarie, pianificare il modo in cui ottenere il risultato, definire i criteri di valutazione, controllare il lavoro correggendo eventuali differenze dagli obiettivi ed infine valutare il risultato raggiunto.

Dalle parole e dalle definizioni poste nel tempo, sì può notare che i concetti ricorrenti sono: “controllo”, “valutazione” e “monitoraggio”. Questi tre concetti risiedono in una fase specifica del PM chiamata “Monitoring and Controlling” (anche detta “Analyze”).

Il controllo, la valutazione e il monitoraggio in un PM sostanzialmente danno un valore aggiunto al progetto per la verifica degli obiettivi preposti.

La grande sfida è quella di raggiungere gli obiettivi restando all’interno del perimetro costituito dai classici vincoli del “Project Management Triangle”(PMT): costo, tempo, scopo/qualità (ogni cambiamento in un lato del triangolo avrà un effetto sugli altri due e tutti insieme produrranno una differente configurazione qualitativa).

Figura 2.1: PM Triangle

Seguendo le caratteristiche del PM, ogni prodotto dovrebbe rispettare questi vincoli in ogni fase del suo «ciclo di vita».

2.1.2 Ciclo di vita: modellizzazione e gestione

L’aumentare di complessità dei sistemi informatici ha spinto le organizzazioni a cercare metodi sempre più strutturati per la gestione delle infrastrutture allo scopo di evitare di intraprendere strade tortuose che impedirebbero il raggiungimento degli obiettivi aziendali.

Sempre più organizzazioni stanno mettendo in atto una rivoluzione interna che porta inevitabilmente a un ridimensionamento e riposizionamento dei singoli ruoli nei processi; da un’istituzione gerarchica si sta lentamente passando a una gestione più orizzontale grazie alla quale è possibile guadagnare in flessibilità e chiarezza.

Un processo di PM è strutturato in attività e da un insieme di azioni che vengono compiute dalle parti coinvolte (che si tratti di una macchina, di un software o di una persona non ha importanza) al fine di ottenere un certo output indispensabile al completamento del processo e quindi al raggiungimento dell’obiettivo finale.

Un servizio IT è progettato, implementato e continuamente sviluppato attraverso il suo «ciclo di vita» che rappresenta un modello organizzativo che ne definisce il metodo di gestione e le tecniche di miglioramento e mantenimento durante il periodo di produzione.

2.1.2.1 Ciclo di vita del software

Il «ciclo di vita» di un software (in inglese «software lifecycle») designa tutte le varie tappe dello sviluppo di un software, dalla sua concezione alla sua scomparsa.

L'obiettivo di una divisione simile è quello di permettere di definire degli stadi intermedi che permettano la validazione dello sviluppo del software, cioè la conformità secondo i bisogni espressi e la verifica del processo di sviluppo.

L'origine di questa divisione proviene dalla constatazione che gli errori hanno un costo tanto più elevato quanto la loro rilevazione avviene tardivamente nel processo di realizzazione.

Il ciclo di vita permette di rilevare gli errori il prima possibile e quindi di controllare la qualità del software, dei tempi di realizzazione e i costi associati. Esso comprende generalmente almeno le seguenti attività:

1. «Raccolta dei requisiti» (Requirement analysis), che consiste nel definire la finalità del progetto, la sua iscrizione in una strategia globale, la raccolta e la formalizzazione dei bisogni del richiedente (il cliente) e dell'insieme dei limiti;

2. «Design, sviluppo e testing» (Prototype & Feedback), ossia la traduzione in un linguaggio di programmazione delle funzionalità definite e il test delle stesse;

3. «Produzione e utilizzo» (Production & Operation), ovvero la messa in opera effettiva del software;

4. «Verifica e manutenzione» (Maintenance & Support), che comprende tutte le azioni correttive o evolutive sul software.

Figura 2.2: Ciclo di vita di un software

Dal punto di vista del profitto e dell'offerta del prodotto software, il ciclo di vita può essere descritto attraverso un grafico che mette in relazione il profitto e l'offerta con il tempo. Le fasi salienti che si possono notare sono [20]:

1. Introduzione (Introduction): il prodotto non è ancora ben conosciuto dai potenziali customer, cosicchè il volume di produzione è ancora basso e di conseguenza si ha un basso profitto.

2. Crescita (Growth): il prodotto diventa più conosciuto nel mercato, i profitti e le offerte salgono velocemente. Solitamente questa fase è la fase in cui entrano in gioco i competitor.

3. Maturità (Maturity): rapida crescita dell'offerta, il prezzo di vendita in questa fase deve essere competitivo quindi anche spesse volte tagliato causando generalemente una perdita di profitto.

4. Saturazione (Saturation): l'offerta ha raggiunto i suoi livelli massimi e i competitor aumentano sempre più causando un declino sempre più forte del profitto.

5. Degenerazione (Degeneration): il prodotto è sostituito generalmente da un altro prodotto più innovativo, la domanda e l'offerta decrescono in maniera esponenziale e per far in modo di non perdere il mercato il prodotto dovrebbe essere rilanciato o sostituito.

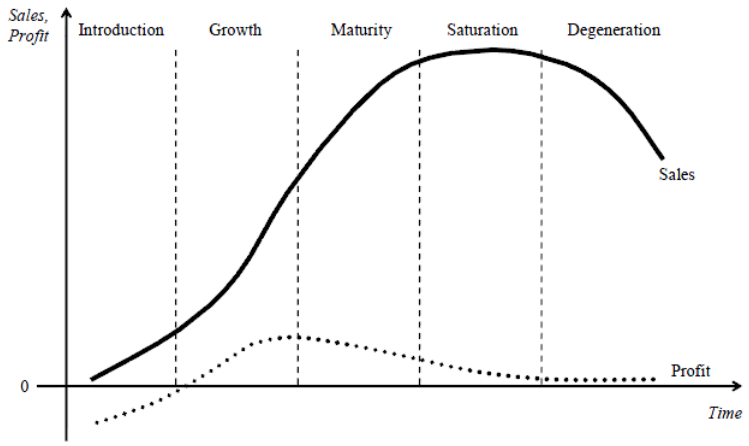

Figura 2.3: Curva del ciclo di vita

Attraverso la versione presentata sia nella Fig.2.2 sia nella Fig.2.3, l'intera implementazione di un ciclo di vita di un software è gestita sotto forma di progetto all'interno di un PM tramite metodologie di gestione e modelli di cicli di vita che verranno mostrati nei seguenti sottoparagrafi.

2.1.2.2 Modelli ciclo di vita

Per essere in grado di avere una metodologia comune fra il cliente e la società che si occupa dello sviluppo, sono stati elaborati dei cicli di vita che definiscono tutte le

tappe dello sviluppo nonché i documenti da produrre per validare ogni tappa prima di passare alla successiva.

Esistono tre tipi di modelli di ciclo di vita:

1. «Cascata» che era popolare negli anni '70 dove ogni fase è caratterizzata da attività (tasks), prodotti di tali attività (deliverables) e controlli di qualità/stato (quality control measures). La fine di ogni fase è un punto rilevante del processo (milestone). Gli output di una fase sono input alla fase successiva. I prodotti di una fase vengono "congelati", ovvero non sono più modificabili se non innescando un processo formale e sistematico.

2. «Prototipato» dove l'obiettivo è capire i requisiti del sistema e quindi sviluppare una definizione migliore dei requisiti. Il prototipo sperimenta le parti del sistema che non sono ancora ben comprese. Realizzazione di una prima implementazione (prototipo), più o meno incompleta da considerare come una «prova», con lo scopo di accertare la fattibilità del prodotto e di validare i requisiti. Il prototipo è uno strumento per validare i requisiti e/o validarne la fattibilità.

3. «Prototipazione Evolutiva» che cerca di superare i limiti principali del modello a cascata, introducendo il concetto di prototipizzazione evolutiva che possiede le seguenti fasi:

   (a) realizzazione di un artefatto,

   (b) consegna al cliente,

   (c) ottenimento dei feedback,

   (d) modifica del progetto in base a tali feedback.

2.1.2.3 Gestione di progetto in un PM: ITIL

In un Project Management, ITIL rappresenta una risposta al problema della gestione di un progetto IT.

Nato negli anni 80 su richiesta del governo del Regno Unito per migliorare il livello dei servizi IT, attualmente è divenuto lo standard per eccellenza grazie anche al suo approccio strutturato all'erogazione di servizi. E' un framework scritto dall'OGC (Office of Government Commerce del Regno Unito) liberamente accessibile ma non aperto; le specifiche infatti sono protette da copyright pur essendo completamente disponibili agli utenti.

ITIL si divide in tre versioni:

1. ITIL v1 che era stata intitolata "Government Information Technology Infrastructure Method" (GITM) e negli anni si è espansa fino a 31 volumi in un

progetto inizialmente diretto da Peter Skinner e John Stewart presso il CCTA. Naturalmente, questa era abbastanza diversa dall'ITIL che si conosce oggi, anche se concettualmente si avvicinava molto e si concentra sulle due aree di service support e service delivery.

2. ITIL v2 usci' nel 2001 e le parti di Service Delivery e Service Support furono riassunte in due concisi volumi.

3. ITIL v3 nasce il 1° Giugno 2007 e la pubblicazione iniziale di ITIL v3 è composta da cinque testi principali denominati: «Service Strategy», «Service Design», «Service Transition», «Service Operation» e «Continual Service Improvement» consolidando così molte delle pratiche della versione v2 attraverso il ciclo di vita del servizio.

Nelle cinque fasi di ITIL v3 si affronta la gestione del servizio partendo dalla definizione degli obiettivi e delle strategie, passando dalla progettazione ed implementazione delle risorse necessarie, concludendo con il test e l'immissione del servizio in produzione.

Lo schema sottostante delinea chiaramente come si pongono le varie fasi rispetto alle altre senza tuttavia dare un'informazione temporale sulla loro esecuzione.

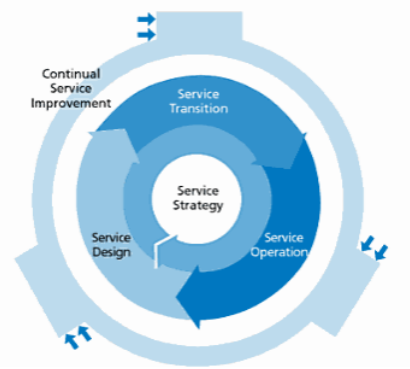

Figura 2.4: Fasi ITIL v3

La fase di Strategy è la prima ad essere eseguita ed è seguita dalle altre i cui processi però si vanno a sovrapporre reciprocamente. La fase di CSI invece inizia al momento del primo rilascio del servizio e prosegue per tutta la sua vita coinvolgendo tutte le fasi precedenti.

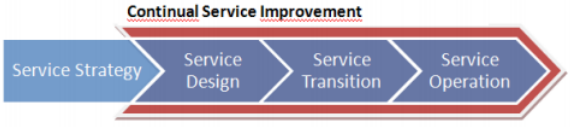

Figura 2.5: ITIL: Continual Service Improvement (CSI)

Le cinque fasi del ciclo di vita sono descritte in ognuno dei volumi di ITIL, di seguito se ne riporta una breve definizione:

1. «Service Strategy»: è la fase di progettazione, sviluppo ed implementazione della gestione del servizio quale risorsa strategica,
2. «Service Design»: è la fase di definizione dello sviluppo dei servizi IT, inclusi l'architettura, i processi, le politiche e la documentazione. Lo scopo è di aderire ai requisiti di business attuali e futuri,
3. «Service Transition»: è la fase di sviluppo e di miglioramento delle capacità per la transizione dei servizi nuovi e modificati, nell'ambiente di produzione,
4. «Service Operations»: è la fase che si occupa di raggiungere l'efficacia e l'efficienza nel fornire e supportare i servizi, in modo da garantire valore per il cliente ed il fornitore di servizi,
5. «Continual Service Improvement»: la fase che si occupa di creare e mantenere il valore per il cliente attraverso il miglioramento della progettazione, l'introduzione e l'esercizio del servizio.

Lo schema dominante è dato dal passaggio dal «Service Strategy» agli altri processi per concludere con il «Continual Service Improvement» per poi tornare al «Service Strategy» qualora si decida di implementare nuove funzioni del servizio o direttamente nuovi servizi.

2.1.2.4 Gestione di progetto in un PM: PRINCE2

L'implementazione di ITIL e' generalmente gestita come un progetto.

Esiste una metodologia di gestione di progetti "preferita" in ITIL, ed e' la PRINCE2.

PRINCE2 è un framework process-based. La metodologia fornita permette un approccio scalabile in ragione dei requisiti richiesti, sia in termini di complessità di progetto sia in termini di rischi connessi. In pratica PRINCE2 permette di suddividere il progetto in piccole fasi più semplici da monitorare e che consentano un'oculata gestione delle risorse disponibili.

L'ambito in cui PRINCE2 potrebbe essere più facilmente applicabile riguarda il processo di Change Management nella fase di Service Transition: nel paragrafo 8.1.3 del libro di ITIL [18], si afferma "In order to be able to define clear boundaries, dependencies and rules, Change Management should be integrated with processes used to control very large organizational programs or projects".

In pratica per definire confini chiari e precisi, le dipendenze dei processi e le regole di gestione, il Change Management deve essere integrato con processi dedicati atti a controllare l'organizzazione dei progetti.

Esempio:

Un'organizzazione già basata su ITIL ha creato un nuovo servizio che per qualche anno aggiorna e mantiene attraverso release minori. Una nuova versione del sistema operativo sul quale gira il servizio viene rilasciata integrando nuove funzionalità che potrebbero migliorare notevolmente il servizio. Nasce l'idea di creare una nuova versione del servizio completamente incentrata sul sistema operativo rilasciato, tuttavia il fornitore vuole introdurre gradualmente le novità senza interrompere il ciclo di vita del servizio vecchio.

Il Change Management decide così di nominare un responsabile di progetto il quale si dovrà occupare di definire un piano di implementazione appropriato. Qui subentra PRINCE2. Si crea il Project Board (comitato di progetto) adibito ad approvare il progetto e tutte le varie fasi. Una volta avuta l'approvazione si inizierà la stesura e la successiva implementazione del nuovo progetto secondo gli otto processi di PRINCE2. Al termine del progetto il prodotto è inviato in qualità di "recommendations" (in stile W3C) al Change Manager il quale provvederà ad avviare la fase di Service Transition di ITIL per il test e la messa in produzione del nuovo servizio.

2.1.2.5 Gestione di progetto in un PM: COBIT

COBIT (Control Objectives for Information and related Technology) invece è un framework che è nato nel 1992 ed ha il compito di fornire una struttura dei processi e una serie di strumenti teorici e pratici con lo scopo di valutare l'efficacia di requisiti di qualità, affidabilità e di sicurezza dell'organizzazione IT.

COBIT fornisce alle organizzazioni un riferimento strutturato, organizzato come ITIL in processi oltre a una serie di strumenti pratici e teorici ad essi collegati ed è uno dei tre standard, assieme ad ISO 17799 e BSI, internazionalmente riconosciuto dall'Unione Europea [16] e utilizzabile per garantire la sicurezza dei sistemi informativi COBIT 4.1 divide la gestione IT in quattro domini: Plan and Organise, Acquire and Implement, Deliver and Support, Monitor and Evaluate.

Nei quattro domini sono collocati 34 processi, ai quali fanno capo un totale di 210 obiettivi di controllo; questi ultimi rivestono un'importanza centrale nel COBIT, al punto di dare il nome al modello stesso.

Nel corso del XX secolo ci sono stati diversi studi inerenti il miglioramento dei processi IT che hanno proposto una svariata gamma di metodologie per poter realizzare un progetto di qualità, avendo come punto di partenza il criterio primario di valutazione che si vuole monitorare: "Goal-Attainment", "Program Improvement", "accreditation of program".

Per ogni processo sono definiti degli obiettivi di controllo specifici raggruppati in due insiemi di obiettivi di controllo "generali": il primo insieme riguarda i processi medesimi (si tratta di controlli identificati con la sigla PCn - Process Control

n) mentre il secondo riguarda i dati in ingresso e in uscita dai processi (controlli identificati con la sigla ACn - Application Control n). Il rispetto degli obiettivi di controllo garantisce il raggiungimento delle finalità ultime dell'azienda assicurando che i processi vengano eseguiti correttamente prevenendo inoltre i rischi associati. COBIT non impone un rispetto rigido delle regole descritte, l'organizzazione può decidere quali sono gli obiettivi di controllo da conseguire e in che misura anche in base a come sono implementati i processi interni.

2.1.2.6 ITIL e COBIT a confronto

COBIT e ITIL non sono framework concorrenti ma complementari. Tuttavia mentre COBIT descrive cosa fare e come controllarlo in termini di obiettivi e processi da attivare, ITIL fornisce indicazioni e strumenti per implementare i requisiti espressi da COBIT.

COBIT si rivolge principalmente ai manager delle organizzazioni, nella pratica infatti i fruitori reali sono i responsabili IT che possono applicare COBIT per conseguire gli obiettivi dell'organizzazione.

ITIL invece amplia il proprio target andando a coinvolgere non solo ciò che provvede ad erogare ogni giorno i servizi IT ma anche le società stesse fornitrici dei servizi e tutte quelle entità necessarie al supporto dell'esecuzione dei processi. Di fatto quindi, ITIL coinvolge l'interesse di tutti i gestori di infrastrutture IT e non implica l'uso di COBIT mentre, viceversa, l'adozione di COBIT implica necessariamente lo studio di ITIL.

Quest'ultimo infine, grazie alla sua forte adattabilità, può essere applicato anche a processi completamente slegati dal mondo IT anche se ovviamente non è lo scopo originario.

2.1.2.7 PRINCE2 e ITIL a confronto

Entrambi i framework provengono dalla stessa fonte, la Office Government Commerce (OGC). A prima vista i libri di OGC non entrano molto nel dettaglio su questo tema. In ITIL viene tuttavia riportata una frase: "Within ITIL when we discuss project management if we need to draw on a particular method we will use PRINCE" [18] che fa capire come i metodi siano collegati, ma non esiste purtroppo una esplicita spiegazione sul come gestirli.

Sia ITIL che PRINCE2 possiedono un meccanismo di valutazione e gestione dei progetti, tuttavia l'integrazione di due standard dà la possibilità non solo di gestire un ambiente di produzione stabile ma anche accettare importanti modifiche all'ambiente stesso.

2.2 Metodologie di valutazione

H.H.Kempfer, un politico americano, nel 1955 affermava: "Scopo basilare della valutazione è stimolare la crescita e il miglioramento. Tutte le altre finalità, pur rispettabili, sono solo sfaccettature dello sforzo generale che consiste nel valutare le condizioni presenti come base per migliorare. Una valutazione che non porti a un perfezionamento delle pratiche è sterile" [21].

Partendo da questo spunto si vuole ora descrivere il compito del processo di valutazione all'interno del PM, come pratica di miglioramento e di perfezionamento di un progetto.

Come detto precedentemente,la valutazione (o in inglese «evaluation») è quella pratica presente nella fase di "Monitoring and Controlling" all' interno del Product Management.

Molto spesso infatti quando si parla di "evaluation" si parla anche di "monitoring", "appraisal", "review" [32] come se fossero sinonimi, costruendo dei modelli che ne caratterizzano il loro studio.

La fase di monitoring viene eseguita mentre un progetto è in fase di realizzazione con l'obiettivo di migliorarne la progettazione; la fase di evaluation invece viene eseguita a progetto terminato con l'obiettivo di informare la progettazione per futuri progetti.

Nel corso del XX secolo ci sono stati diversi studi inerenti il miglioramento dei processi IT che hanno proposto una svariata gamma di metodologie per poter realizzare un progetto di qualità avendo come punto di partenza il criterio primario di valutazione che si vuole monitorare: «Goal Attainment», «Program improvement» e «accreditation of program».

2.2.1 Goal Model

Quando si parla di evaluation, si intende quel meccanismo di confronto rispetto a qualcosa di oggettivo, di misurabile, che si possa appunto valutare in base ad un obiettivo prestabilito; tra le varie metodologie di modello troviamo il "Goal-Attainment".

Il modello Goal-Attainment concepisce l'evaluation come la determinazione del livello raggiunto rispetto agli obiettivi imposti. Il modello nasce dall'idea di Ralph.W.Tyler negli anni '30 e alla sua ricerca nota come "Eight Year Study".

Tyler (1950) affermò che "la valutazione dovrebbe giudicare che un programma è buono se, e solo se, i suoi obiettivi sono stati raggiunti".

Da questa affermazione nascono tutti gli i modelli "Goal Based", il quale mostrano la relazione tra i cosiddetti "means" e "goal" [36], rispettivamente tra un prodotto (o un servizio o un comportamento o un requisito) e l'obiettivo da raggiungere.

Il "Goal Based Model" si basa sull'assunzione che i "consumers" valutano il prodotto seguendo una semplice formula [36]:

$$v_i = \sum_j (g_j \text{ x } p_{ij})$$

che esprime "la valutazione di un prodotto i (v_i) dipende dalla misura in cui i aiuta o ostacola il raggiungimento di un obiettivo j (p_{ij} che rappresenta la percezione di consumo di i per il raggiungimento di un goal j) pesato dall'importanza del goal (g_j), sommato su tutti gli obiettivi j" .

L'obiettivo prefissato da raggiungere è l'elemento fondamentale nel "Goal Attainment", seguendo una logica SMART (Specific, Measurable, Attainable, Relevant, and Timely). Gli obiettivi sono solitamente impostati da responsabili e persone qualificate il quale utilizzano: dati storici temporali, risorse (per cercare di capire per esempio se le risorse impiegate sono in numero adatto al progetto svolto) e strumentazione specifica.

I valutatori devono attenersi solo ai livelli degli obiettivi prestabiliti all'inizio del progetto, senza mai essere cambiati.

Il valutatore per il "Goal based Model" è colui che sa cosa deve raggiungere e ne verifica il suo raggiungimento.

L'analisi "Goal Oriented" è stata proposta come una sorta di intenzionalità a soddisfare i requisiti software all'interno della Requirements Engineering (RE). I goal sono visti come l'astrazione dei bisogni degli stakeholder uniti alle loro aspettative [28].

Questa ideologia è stata molto criticata con il tempo soprattutto da Scriven (1972), a tal punto che lanciò un nuovo modello : il "Goal Free Evaluation".

La critica di Scriven faceva riferimento alla mancanza di una valutazione libera da obiettivi nella quale il valutatore possa comprendere lo sviluppo e l'evoluzione del progetto, senza in alcun modo farsi condizionare dagli obiettivi di partenza. Il valutatore, secondo Scriven, può avere una funzione interna e/o esterna alla valutazione, è attento alla qualità degli obiettivi del programma ma anche al livello che essi raggiungono.

Scriven divide la valutazione in:

1. formativa: la valutazione di un programma durante la sua fase di attuazione, allo scopo di apporvi parziali adattamenti e verificare il reale contributo delle attività che si stanno svolgendo

2. sommativa: la valutazione di un programma giunto alla sua conclusione, dopo gli eventuali aggiustamenti determinati dalla valutazione formativa, in cui si valuta la totalità del programma e degli effetti ottenuti

Scriven esorta i valutatori a cercare risultati voluti e non voluti non concentrandosi solo sugli obiettivi di partenza. Questa metodologia è usata per default soprattutto in programmi dove gli obiettivi non sono stati prestabiliti inizialmente nella fase di planning [6].

Nel goal free evaluation chi valuta non può essere neutrale in quanto la valutazione stessa è un giudizio di un valore, quindi per Scriven la valutazione non si doveva riferire all'obiettivo ma allo standard di qualità. Le fasi di valutazione devono essere:

1. stabilire criteri di merito e standard di qualità,
2. misurare le performance dei singoli programmi,
3. creare un graduatoria dei programmi,
4. giudizio finale di valore.

Si usa quindi in questo caso un modello prospettista della qualità («il programma è di buona qualità?») piuttosto che un modello positivista («il programma ha conseguito gli obiettivi?»).

Ricapitolando:

1. Goal Attainment/Goal Based:
   (a) esistono obiettivi prestabiliti da essere misurati,
   (b) specifica una serie di preformance che indicano se si è raggiunto un goal,
   (c) indica quali sono le performance che se raggiunte pongono critico il raggiungimento del goal,
   (d) indica un valore di valutazione del goal,
   (e) si basa su un modello positivista.
2. Goal Free
   (a) utilizzato spesso in valutazioni cercando risultati anche al di fuori degli obiettivi prestabiliti,
   (b) identifica effetti positivi e negativi dei risultati ottenuti,
   (c) si basa su un modello prospettista della qualità.

2.2.2 Decision Making

Oltre all'approccio del Goal Attainment e del Goal Free, l'evaluation propone altre tecniche che si focalizzano su altre tipologie di approcci, tra questi troviamo il Decision Making.

Il «Decision Making approch» è caratterizzato fondalmentalmente da 3 fasi: focalizzazione dell'informazione (delineating), collezione e organizzazione dell'informazione (obtaining) e sintetizzazione dei risultati ottenuti (providing).

Il decision maker solitamente è un manager o un amministratore che vuole e che ha bisogno di informazioni necessarie per rispondere a domande importanti sullo svolgimento di una determinata attività o progetto che sia.

Le quattro fasi di valutazione a cui solitamente si fa rifermento sono: «context», «input», «process» e «product». Essi rappresentano:

1. Il "context" descrive la condizione e l'ambiente,
2. "l'input" identifica come usare risorse per raggiungere un goal,
3. "process" fornisce risposte ai responsabili che attuano il programma,
4. "product" indica invece la misurazione del raggiungimento durante e dopo la realizzazione del programma.

Da queste quattro fasi nasce il «Context – Input – Process - Product» (CIPP) , uno dei metodi di decision making sviluppato da Stufflebeam (Di cui sotto vengono mostrate tutte le fasi di stage) [22].

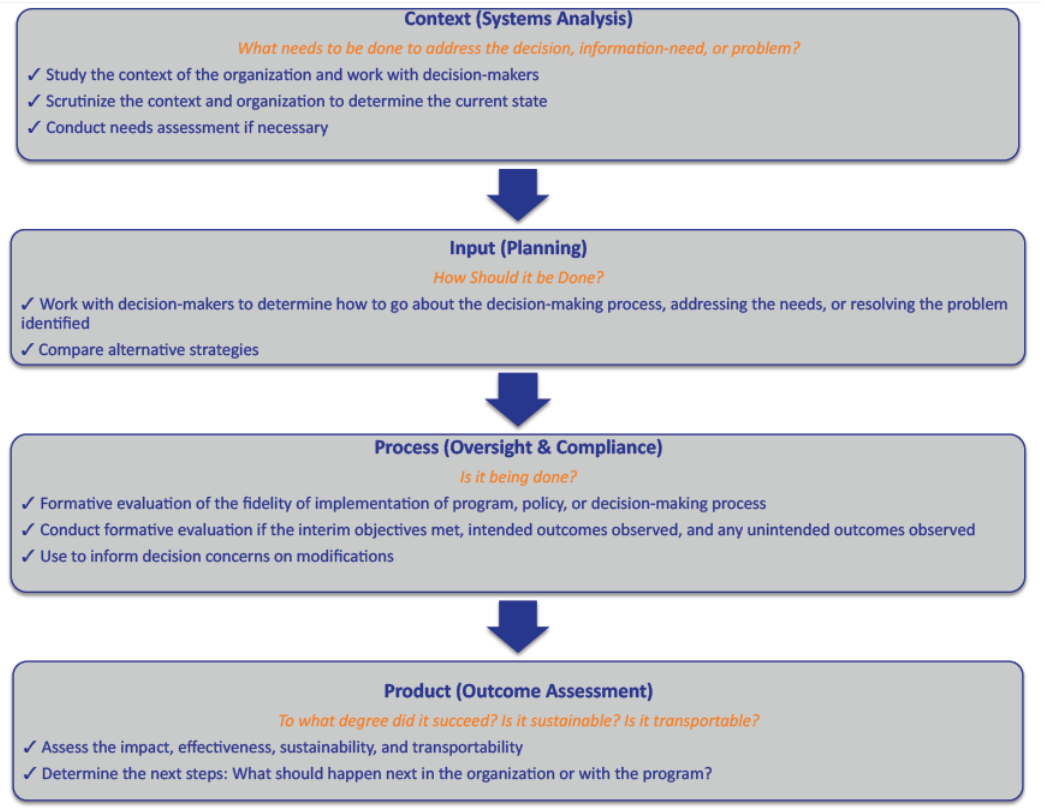

Figura 2.6: CIPP Stage

Stufflebeam nel 2007 pubblicò una checklist dove fece un riassunto di tutte queste fasi di stage di valutazione indicando il momento esatto di un processo (anche in un PM) in cui applicare il CIPP model.

Questo documento inoltre fornisce una serie di checklist di azioni comuni che l'evaluator deve compiere all'interno di uno stage. Questo meccanismo possiede diversi vantaggi:

1. sottolinea l'importanza delle informazioni,
2. la valutazione può essere fatta mentre si sta eseguendo il progetto,
3. è un grande stumento oggettivo di aiuto per il valutatore per generare domande importanti inerenti il controllo,
4. supporta l'operazione, la crescita ed il cambiamento del progetto,
5. sottilinea l'uso tempestivo del feedback nelle varie fasi.

Ma possiede anche svantaggi:

1. occasionalmente inabilità del valutatore di rispondere alle domande o problemi che possono essere significativi,
2. i programmi che non hanno una leadership decisiva non possono beneficiare al meglio di questo approccio,
3. può provocare valutazioni costose e complesse anche in termini di tempo.

Nel Decision Making la definizione del problema è il primo punto cruciale per il raggiungimento di una decisione valida. Questo processo deve perlomeno identificare le cause principali restringendo il numero di condizioni, i limiti, gli aspetti del sistema, le connessioni e gli elementi relativi agli stakeholder.

L'Analisi Decisionale non descrive come e perché un individuo prende una decisione, bensì prevede una procedura di decisione che sia compatibile con le preferenze e il comportamento di un individuo.

Alcune osservazioni:

1. la maggior parte delle decisioni prese a livello aziendale non richiedono (e solitamente non utilizzano) un'analisi formale. Ma per quella decisione dalla quale "dipende tutto", è molto utile avere una procedura logica e sistematica per decidere;
2. La maggior parte delle persone ha pochissima esperienza su come gestire dati statistici ed esemplificativi, con i quali è probabile ci si debba confrontare durante un processo di Decision Making. Di conseguenza, è preferibile affidarsi all'approccio meccanico dell'Analisi Decisionale piuttosto che alle poco attendibili capacità di valutazione del nostro intuito;

Oltre al metodo CIPP, si ricorda anche il metodo Kepner-Tregoe (K-T) che è una comparazione quantitativa nella quale un team di esperti ordina numericamente i criteri e le alternative in base al giudizio individuale di ciascuno di loro. Nel metodo K-T, ogni criterio di valutazione (o anche un goal) ha un valore, a seconda della sua importanza rispetto agli altri criteri (1 = «meno importante», 10 = «più importante»). Questi punteggi diventano il valore del criterio.

Il punteggio totale per ogni alternativa è poi determinato moltiplicando il suo punteggio per il valore di ogni criterio (fattore di valore relativo per ogni criterio) e poi sommando tutti i valori ottenuti. L'alternativa migliore avrà il punteggio totale maggiore.

Il metodo di analisi K-T è valido per decisioni mediamente complesse con pochi criteri. Il metodo richiede solo la conoscenza di elementi di base di aritmetica.

Lo svantaggio maggiore consiste, ad esempio, nel fatto che esso potrebbe non spiegare come possa essere meglio un punteggio di 10 piuttosto che un punteggio di 8; inoltre, il punteggio totale delle varie alternative potrebbe essere molto simile, quindi rendere difficile la scelta.

Anche in questo caso si è di fronte a forte incertezza e poca esperienza.

2.3 Fasi del processo di valutazione

Il monitoraggio e la valutazione di progetto all'interno di un PM è diretto a osservare e misurare l'esecuzione del progetto (tramite gli indicatori definiti nelle metriche di progetto) in modo da identificarne per tempo i rischi e i potenziali problemi intraprendendo così le azioni correttive volte a rimettere il progetto in linea con i propri obiettivi.

Gli strumenti di project management possono essere intesi sia come le tecniche utilizzate per supportare la realizzazione delle attività di project management, sia come i prodotti software che implementano tali strumenti e li forniscono contestualmente ad un insieme integrato di servizi e/o funzionalità.

Le fasi importanti del processo di valutazione vengono proposte di seguito e verranno dettagliatamente spiegate nei paragrafi successivi:

1. l'identificazione e la modellizzazione degli obiettivi,
2. riorganizzazione,
3. parametrizzazione,
4. valutazione.

2.3.1 Identificazione e modellizzazione degli obiettivi

Il punto di partenza di un PM è sempre la fase di identificazione degli obiettivi.

Nella fase di "Planning" di un PM vengono stilati in un documento i requisiti di progetto in seguito ad uno studio un'analisi di mercato o di una scelta manageriale o dall'analisi di documenti specifici (come il Product RoadMap).

Gli obiettivi possono derivare sia da requisiti di carattere funzionale (ovvero quei requisiti che esprimono un'azione che il sistema dovrebbe eseguire) sia da requisiti di carattere non funzionale (ovvero tutti quelli che non possono essere descritti tramite casi d'uso, per esempio l'usabilità e la sicurezza) molto spesso di "alto livello" e poco dettagliati.

Nelle metodologie "Goal Based" il valore dell'obiettivo è parte fondamentale in un PM, quindi si deve essere in grado di stilare quali siano tutti i goal che si intendono realizzare cercando di non tralasciare nulla.

Esistono diversi strumenti di specifica degli obiettivi che guidano il project manager ad una maggior comprensibilità degli scopi e della valutazione di progetto, tra cui si trovano i diagrammi I*.

2.3.1.1 Diagrammi I*

"I* modeling" è un framework per il linguaggio di modellizzazione che ha lo scopo di descrivere fondalmentalmente tutte le dipendenze tra vari attori utilizzatori all'interno di una suite di prodotti, che siano essi software o di altro genere.

In questa modellistica sono presenti 4 elementi fondamentali: goal, soft goal (requisiti non funzionali e le loro relazioni), task e resources.

Il modello poi è suddiviso poi in due componenti:

1. Strategic Dependency model (SD): modello che descrive una rete di relazioni di dipendenza tra attori vari in un contesto specifico. E' caratterizzato da un insieme di nodi e link che connettono gli attori. I link di dipendenza indicano che un «attore» (depender) dipendono da un altro «attore» (the dependee) attraverso «qualcosa» (the dependum). Il «dependum» può essere un task, un goal o una risorsa. [37]

2. Strategic Rationale model (SR): modello che permette di descrivere come gli attori realizzano i loro goal. Questo modello mostra le dipendenze degli attori inclusi in SD Model. Si attribuiscono goals, task, resouces e soft goal ad ogni attore segnando dei link (mean end link); i «mean and link» collegano i task ad un goal, ed ogni task può essere suddiviso in subtask (o subgoal o resouces o softgoal). Tra i vari componenti sopra elencati possono essere presenti anche relazioni AND e OR, che indicano alternatività o obbligatorietà. [37]

Il diagramma fornisce la possibilità di realizzare un insieme di analisi e di modellistica simile ad UML partendo dai requisiti funzionali e non funzionali.

I* (insieme a Tropos e KAOS, che sono altre metodologie di analisi dei requisiti e di modellistica) rappresenta ciò che viene chiamata "intentional ontology" utilizzata

spesso nella fase di Requirements Engineering per cercare spiegare il "perchè" di un sistema software [28].

Infatti in i* modeling vi è più l'interesse a capire il "perchè" delle cose (lo scopo) piuttosto che il "cosa" o il "come" (come invece propone UML).

Di seguito viene presentato un esempio di diagramma SD tratto dalla guida on line di i* modeling (http://istar.rwth-aachen.de/tiki-index.php?page=iStarQuickGuide) che rappresenta un modello di «Buyer Drive E-Commerce».

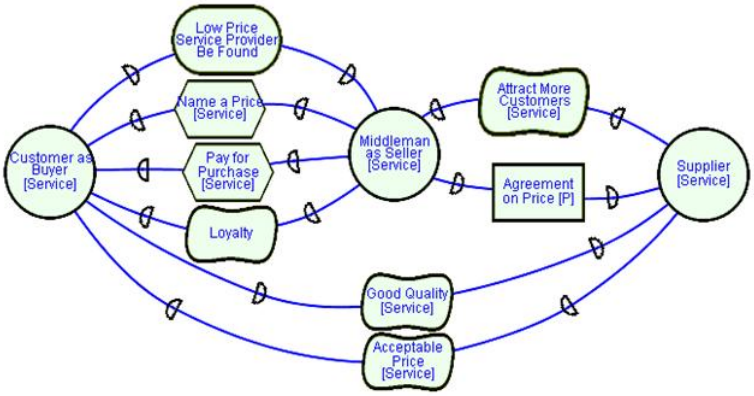

Figura 2.7: Esempio diagramma SD

1. Gli attori sono rappresentati da un cerchio (in questo caso sono i customer, i venditori e i fornitori).

2. I task sono rappresentati da un ottagono e rappresentano tutti i compiti che gli attori devono o possono compiere all'interno del processo (in questo caso il customer ha il compito di trovare il servizio a minor prezzo).

3. Ci sono i soft goal ed i goal, rappresentati da rettangoli dalle forme arrotondate, che in questo caso sono quelli di avere una buona qualità di servizio, oppure di avere un prezzo accettabile ed una buona fedeltà del venditore nei confronti del customer. I goal rappresentano un desiderio intenzionale dell'attore, degli obiettivi forti da raggiungere; invece i softgoal sono degli obiettivi che un attore si ritiene soddisfatto qualora li raggiungesse.

4. Ed infine ci sono le risorse a disposizione che sono rappresentate da un rettangolo (in questo caso sono rappresentate dagli accordi sul pagamento).

Al diagramma SD è affiancato il diagramma SR che descrive piu dettagliatamente il tutto, introducendo links che vanno a dare un profilo più dettagliato dell'interazione tra i vari attori. Di seguito è presente il corrispettivo SR diagram dell'SD diagram presentato precedentemente:

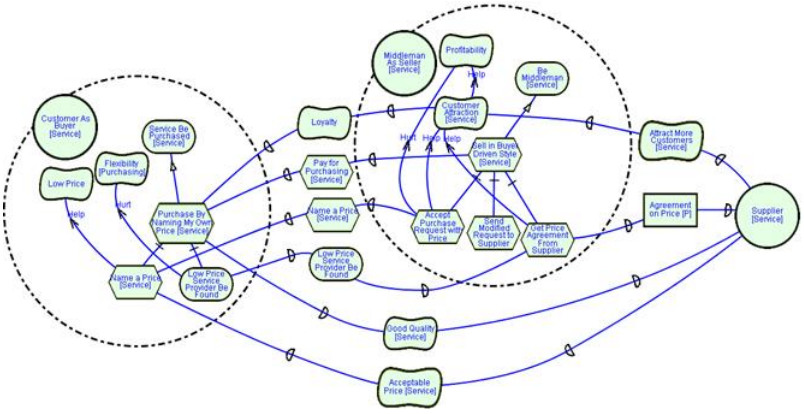

Figura 2.8: Esempio diagramma SR

2.3.2 Riorganizzazione

Una volta specificati gli obiettivi da raggiungere, i requisiti da rispettare, i task che si ha la possibilità di compiere e le risorse necessarie per il loro raggiungimento, la fase successiva è quella di riorganizzare gli obiettivi dando una rappresentazione più compatta del prodotto da sviluppare nel PM.

I diagrammi I* hanno la capacità di dare un grande contributo nella fase di planning e di analisi dei requisiti, mentre nelle successive fasi si avrà il compito di definire in maniera più chiara «chi può fare cosa», ovvero si avrà il compito di riorganizzare gli obiettivi associandoli ad un attore (un prodotto, un software) attraverso metodologie di rappresentazione più compatte per una migliore organizzazione del lavoro e anche di costi (di gestione, di tempo e di sviluppo).

Tra queste metodologie si trovano le «feature model» che, a differenza dei Goal Model i* (che catturano un largo numero di alternative AND/OR con un insieme di operazioni, task e configurazioni che possono essere trovate per cercare di raggiungere i goal degli stakeholder [35]), rappresentano modelli più compatti che siano in grado di rappresentare tutta questa decomposizione.

2.3.2.1 Feature Model

Le Feature Model sono una forma di rappresentazione di una vasta gamma di prodotti. I prodotti sono rappresentati tramite «feature» e le «feature model» sono spesso utilizzate in una Sofware Product Line (SPL) per cercare di trovare le parti comuni e le parti variabili in una vasta gamma di sofware.

Una Feature model è rappresentata solitamente a forma di albero (cross-tree) caratterizzato da una root e da una serie di “rami” che indicano le relazioni che intercorrono tra i diversi nodi. A seconda del tipo di estensione della feature model, su ogni ramo possono esserci presenti le cardinalità delle relazioni (uno-uno, uno-molti).

Per una maggior comprensione, sotto viene presentata la feature model di un automobile.

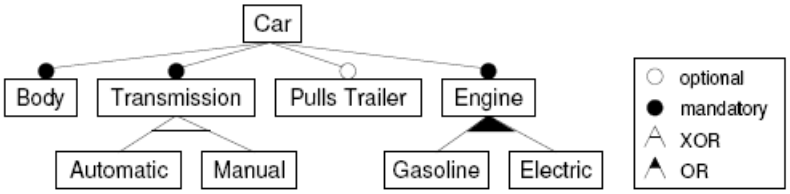

Figura 2.9: Feature model di un automobile

Nel linguaggio naturale, questa figura descrive il fatto che un'automobile possiede obbligatoriamente un corpo, un meccanismo di trasmissione ed un motore (opzionalmente può avere un gangio per un eventuale rimorchio).

La trasmissione può essere sia automatica sia manule e il motore può essere a benzina o elettrico (o entrambi).

Come si nota, le relazioni padre-figlio all'interno dell'albero sono caratterizzate da livelli di obbligatorietà, di opzionalità, alternatività e esclusività in modo tale da rappresentare tutta la gamma progettuale come J.Mylopoulos sosteneva nelle sue teorie di modellizzazione della RE [35].

La notazione utilizzata è di tipo GP (Generative Programming), ovvero:

Symbol	Feature Type	Meaning
aFeature	mandatory	All included
aFeature	optional	Option (may be included or not)
	alternative	XOR-specialization
	or -feature	OR-specialization

Figura 2.10: Notazione «Generative Programming»

In questo modello ad albero si potrebbero avere anche link (linee orienate identificate con un tratto discontinuo) che identificano vincoli di "required" e "excluded" (A <<required>> B o A<<excluded>>B dove A e B rappresentano le feature).

2.3.3 Parametrizzazione

Una volta modellizzati e riorganizzati gli obiettivi, si deve procedere con la parametrizzazione e la messa in opera del processo di valutazione.

In questa fase quindi è necessario capire ciò che si ha in input ed in output definendo delle metriche su cui poi applicare il processo di valutazione.

Il protagonista in un processo di valutazione è un obiettivo o una serie di obiettivi da raggiungere. Per considerare che si è raggiunto un obiettivo si ha la necessità di valutarlo secondo delle metriche non solo soggettive, ma soprattutto oggettive.

La fase di parametrizzazione consiste nel trovare metriche e meccanismi utili alla fase successiva, ovvero alla fase di valutazione; di seguito saranno presentati i meccanismi «Goal Question Metric» e di «Neural Fuzzy Network».

2.3.3.1 Goal Question Metric

Il «Goal Question Metric» (GQM) è un approccio di metrica del software proposto da Victor Basili, per dare un carattere oggettivo alla valutazione. Vengono proposti 3 tipi di livelli di misura:

1. «Goal»: obiettivo prefissato
2. «Question»: una serie di domande per definire i modelli dell'oggetto di studio affinchè si possa raggiungere un goal specifico
3. «Metric»: metriche dei modelli associate alle question definite.

Il GQM è basato sull'assunzione che un'organizzazione, per "misurare" in modo mirato un progetto, dovrebbe partire dalla specifica di un goal [29] e poi proseguire con il resto della specifica. Seguendo il GQM bisogna aver chiaro quindi cosa si vuole raggiungere per poi capire cosa è necessario.

Intorno all'obiettivo viene definito un insieme di domande la cui risposta ha il compito di definire l'obiettivo stesso, ad esempio:

1. «Esistono mal funzionamenti critici dal punto di vista dell'utente?» oppure «Quale è il tempo medio tra due malfunzionamenti?» sono domande che possono misurare l'affidabilità di un sistema.
2. «Quanto sforzo deve essere applicato alla fase di test o al test del singolo modulo?» e «Quale percentuale del codice sorgente è stata coperta nella fase di test?», sono domande rilevanti per valutare costo e affidabilità del sistema.

Può capitare che diverse domande possono essere utili per raggiungere uno stesso obiettivo ma anche che la stessa domanda può essere usata per diversi obiettivi. In una metodologia GQM completa si giunge ad una collezione di obiettivi e di diverse domande.

Ad ogni domanda infine viene associata una metrica precisa, ad esempio la risposta a «Quale percentuale percentuale di diramazioni è stata ricoperta?» sarà sicuramente un numero compreso tra 0 e 100 (la valutazione da parte dell'utente di una criticità di un malfunzionamento può essere definita mediante un numero che convenzionalmente è compreso tra 1 e 10, dove a 10 si trova la criticità massima).

Si noti che alcune domande comportano una risposta umana (per esempio «quanto tempo è stato utilizzato per il debugging») mentre altre possono ricevere risposte automaticamente, almeno in linea di principio («quale percentuale di condizioni è stata ricoperta»)

L'esempio presentato vuole illustrare le difficoltà associate agli approcci tradizionali alle metriche del software: si devono assegnare misure quantitative a fatti che hanno solo un valore soggettivo.

Anche quando la domanda ha una risposta ovvia e ben definita, come la percentuale di diramazioni ricoperte da un frammento di codice, l'impatto reale della misura sull'obiettivo che si vuole definire può non essere evidente.

Pertanto è necessario applicare buon senso, grande attenzione ed esperienza nella definizione di aspetti critici quali gli obiettivi, le domande e le misure [7].

Questo meccanismo è stato utilizzato anche dalla NASA per valutare una serie di difetti in un insieme di progetti (NASA Goddard Space Flight Center).

In seguito alla selezione di una feature e dei goal, è possibile procedere utilizzando questo meccanismo arrivando così a trovare le metriche necessarie per la valutazione nel PM.

2.3.3.2 Reti neurali

Le Reti neurali artificiali sono un meccanismo che prendono spunto dalle reti neurali presenti nel nostro cervello. Il cervello è una complessa organizzazione di cellule nervose, con compiti di riconoscimento delle configurazioni assunte dall'ambiente esterno, memorizzazione e reazione agli stimoli. Al fine di compiere queste operazioni, la rete biologica cerebrale si serve di miliardi di semplici elementi computazionali (neuroni) fittamente interconnessi in modo da variare la loro configurazione in risposta agli stimoli esterni: in questo senso si può parlare di processi di evoluzione per apprendimento e di sistemi d'intelligenza artificiale che cercano di replicare questo modello. Tradizionalmente, il termine "rete neurale" viene utilizzato come riferimento ad una rete o ad un circuito di neuroni biologici, ma se ne è affermato l'uso anche in matematica, con riferimento ai modelli matematici delle reti neurali artificiali, che rappresentano l'interconnessione tra elementi definiti neuroni artificiali, ossia costrutti matematici che in qualche misura "imitano" le proprietà dei neuroni viventi.

I compiti ai quali le reti neurali artificiali sono chiamate a dare risposte variano dai sistemi di controllo di veicoli e di processi industriali, alle funzioni di approssimazione per la previsione delle tempeste metereologiche, alla simulazione di videogame, all'identificazione di volti, lineamenti e caratteristiche biometriche, al riconoscimento di oggetti per la sicurezza negli aereoporti, alla verifica di processi decisionali ed a molto altro ancora.

Molto spesso i metodi utilizzati nelle reti neurali atificiali hanno comprese delle "fuzzy logic", ovvero delle logiche che a differenza delle logiche binarie non hanno solo valori vero (pari a 1) e falso (pari a 0), ma anche dei valori intermedi (compresi da 0 a 1).

Solitamente ai valori fuzzy si identificano delle "fuzzy rule" sotto forma di clausole "IF THEN ELSE" che rappresentano le relazioni inferenziali tra ciò che c'è in input e ciò che avremo in output.

Le logiche fuzzy sono tipiche del soft computing e si prefiggono di valutare, calcolare, decidere e controllare lo scenario in un ambiente impreciso, vago, fluido o soggetto a continui e repentini cambiamenti, emulando e utilizzando la capacità degli esseri umani di eseguire le suddette attività sulla base della loro esperienza.

2.3.3.3 Fuzzification e Neural Fuzzy Network

Il meccanismo chiamato "Fuzzyfication" è quel meccanismo che permette di rendere "imprecisa" una variabile in input o in output, usando la fuzzy logic nello sviluppo delle euristiche delle fuzzy rules.

Per ogni input (variabile da analizzare e/o requisito funzionale o non funzionale), si devono definire delle Membership Function (MF) che permettono di classificare il nostro input in una categoria qualitativa stabilita (es: low, normal or high).

Il grafico risultante di questa funzione può essere di diverso tipo anche se solitamente è un trapezoide o un triangolo (o pseudo-trapezioide).

Una volta definite le MF e il grafico, si deve procedere con la formazione delle fuzzy rule seguendo questo formato "IF «antecedents» THEN «conclusions» permettendo così di trasformare gli input nei rispettivi valori di output.

A seconda del numero di MF per gli input e per gli output, si dovrà essere in grado di definire più o meno regole fuzzy; più variabili si avranno e più dovremmo definire regole per dare completezza al nostro lavoro.

Le fuzzy rule sono di diverso tipo e dipendono da caso a caso e dal tipo di analisi o vincolo che si vuole porre. Ecco un esempio di fuzzy rule:

$$input : x,\ output : y, z \Rightarrow rule : \ IF\, x\, is\, low\, THEN\, y\, is\, low$$

Le variabili in input ed in output sono delle variabili linguistiche, ovvero si intendono quelle variabili i cui valori sono parole o frasi di un linguaggio naturale o artificiale.

Esempio:
«Età» è una variabile linguistica se i suoi valori sono linguistici piuttosto che numerici, ovvero: «giovane», «non giovane», «molto giovane», «vecchio», «non molto vecchio». Queste variabili sostituiscono i classici valori di numerici che esprimono il

concetto di età (esempio:25 anni)

Una volta stabilite quali siano le fuzzy rule è possibile modellizzare su di esse delle Neural Fuzzy Network (NFN), un ottimo strumento utile a fornire delle indicazioni anche in termini di valutazione in tempi ragionevoli.

Infatti le Neural Network offrono la possibilità di risolvere problemi di "tuning" e dal lato opposto il fuzzy controller, seguendo le logiche strutturate della conoscenza attraverso le fuzzy rule, permette all'utilizzatore di avere una interpretabilità semplice del risultato [30].

Le reti neurali fuzzy al giorno d'oggi sono spesso utilizzate come meccanismi di "prediction and forecasting" di un evento futuro che permettono di effettuare decisioni chiave per la gestione del servizio da analizzare: nel 2005 Nayak propose una rete adattativa neurale per la previsione di un flusso di un fiume, nel 2008 venne proposto un approccio basato sullo studio del livello di innalzamento del mare con questa metodologia. I problemi che si possono incontrare utilizzando i meccanismi "Goal model" o "Decision making" sono di diverso carattere tra cui:

1. Tempo: molto tempo può essere consumato per la presa della decisione. La decisione individuale può pretendere abbastanza tempo perchè il manager è chiamato a fare uno studio sui risultati ottenuti (con magari grandi quantità di dati) ed attuare così uno studio sui meriti e sui demeriti di tutte le alternative proposte.

2. L'incertezza e l'incontrollabilità dei fattori ambientali e tecnologici, può portare il manager verso una scelta errata. E' necessario quindi effettuare studi e introdurre politiche (regole) che ne permettono di controllarne i risultati tenendo conto sia delle regole già prefissate sia delle esperienze passate (un po come il nostro cervello) affidandoci meno ai calcoli probabilistici del momento.

3. Richieste di variazione per i prodotti, i componenti, i progetti sviluppati implicando così una riorganizzazione dei goal.

L'introduzione di una fuzzy network permetterebbe di ovviare a questi problemi.

Costruita la Neural Fuzzy Network ed applicate le fuzzy rule prestabilite, ci permetterebbero così di avere dei risultati in tempi rapidi senza alcun meccanismo di studio, ma tutto già prestabilito al momento opportuno alla stilazione degli obiettivi, dei requisiti (funzionali e non funzionali) e delle variabili di incertezza.

Le fuzzy rule servono quindi come approccio meccanico dell'Analisi Decisionale e come procedura logica e sistematica per poter decidere e valutare.

Il livello di astrazione è forte ed è possibile quindi applicare questo metodo con variabili in input eterogenee, anche molto diverse tra loro in quanto il tutto sarà presentato sotto forma di regole linguistiche.

2.3.3.4 Applicazioni fuzzy

La strategia fuzzy viene spesso utilizzata nella valutazione dei rischi e nella valutazione delle attività di un progetto (PERT- Project Evaluation Review Technique [10]).

Il rischio è una delle maggior preoccupazioni che un'azienda deve avere a che fare, dato appunto l'alto grado di imprecisione e l'elevata incertezza di previsione. Zadeh (il primo ad introdurre la logica fuzzy) si era posto proprio il compito di cercare di rappresentare l'incertezza tramite dei metodi matematici che siano in grado di descrivere l'incertezza; da questo spunto furono nati i metodi fuzzy.

I "fuzzy number" sono utilizzati nella valutazione dei rischi in diversi modi nelle letterature presentate: Kucka per esempio presentò una metodologia fuzzy per misurare le criticità di un'attività di progetto o dell'intero progetto [13], la misura di criticità serve a misurare il rischio ed aiutare nella scelta da intraprendere (accettare il rischio o meno).

In questo approccio, il decision maker esprime ciò che è "veramente critico", "abbastanza critico" e "poco critico" ponendo la criticità pari a:

$$CritPr = \sum_{i=1}^{N} w_i Crit_i$$

dove w_i sono i pesi e $\sum_{i=1}^{N} w_i = 1$, N è il numero delle attività del progetto e $Crit_i$ sono le criticità del progetto i-esimo [17].

Altre applicazioni le troviamo con: Bovicini [5] che propose una metodologia fuzzy logic per la valutazione dei rischi nel trasporto dei materiali pericolosi,Carr e Tah [8] che presentano un'analisi del fuzzy risk prendendo spunto dal classico modello qualitativo di valutazione del rischio, Cho et al [11] che proposero una metodologia per incorporare l'incertezza usando una convenzione fuzzy all'interno della convenzionale valutazione del rischio.

Nelle valutazioni di rischio e non solo, spesso vengono utilizzate anche metodologie AHP (Analytic Hierarchy Process).

L'AHP è uno dei metodi di analisi multicriterio e consente prevalentemente di assegnare una priorità ad una serie di alternative decisionali (stimoli) o di mettere in relazione criteri caratterizzati da valutazioni qualitative e quantitative e quindi non direttamente confrontabili, combinando scale multidimensionali di misure in una singola scala di priorità (Saaty, 1980; Figueira, 2005).

L'analisi multicriterio per scopo decisionale (Multi Criteria Decision Analysis, MCDA) è una disciplina orientata a supportare il decisore qualora si trovi a operare con valutazioni numerose e conflittuali, consentendo di ottenere una soluzione di compromesso in modo trasparente.

Alcuni dei metodi MCDA più utilizzati sono: Analitical Hierarchy Process (AHP), Multi-Attribute Global Inference of Quality (MAGIQ), Goal Programming, ELECTRE (Outranking), PROMETHÉE (Outranking), Data Envelopment Analysis, The Evidential Reasoning Approach, Dominance-based Rough Set Approach (DRSA), Aggregated Indices Randomization Method (AIRM).

Questi metodi presentati di fuzzificazione sono utilizzati spesso come metodologie da praticare nella fase pre sviluppo di un PM, ovvero nella fase di Analisi.

2.3.4 Valutazione

La fase conclusiva di un processo di valutazione è proprio la valutazione stessa.

In questa fase lo scopo è quello di arrivare alla costruzione di un metodo semplice ma efficace di valutazione caratterizzato dalla presenza di bassa incertezza. I progetti vengono valutati per una serie di motivi e può significare di misurare il valore di un determinato modulo o programma.

I processi di valutazione si trovano quasi in tutte le fasi di un processo ed esistono diverse fasi e modelli di valutazione (come visto precedentemente) che ognuna ha le sue componenti di analisi dei problemi; i modelli assistono il project manager nello sviluppo di un quadro di come questa valutazione sarà effettuata in tutte le fasi di progetto.

Il compito è quindi quello di trovare un criterio di valutazione che permetta al progetto di essere classificato per poter poi fare le considerazioni necessarie alla fine di un processo.

La metodologia fuzzy e i meccanismi GQM della fase di parametrizzazione rappresentano la premessa della fase di valutazione, come avviene anche nella «Fuzzy Multi Agent Measurements» (una metodologia utile alla valutazione della qualità di un software).

2.3.4.1 Fuzzy Multi Agent Measurements

Mir Ali Seyyedi, Mohammad Teshnehlab e Fereidoon Shams nel 2005 proposero un metodo di "Fuzzy Multi Agent Measurements" che studia la qualità e la maturità di un software partendo dalla «Software Capability Maturity Model» (SW-CMM) e dalla «Multi-level Fuzzy Inference Model».

La «Software Capability Maturity Model» è un modello che provvede a dare le linee guida in un "software process" ed è formato sostanzialmente da 5 livelli: «Initial», «Repeated», «Defined», «Managed» e «Optimized».

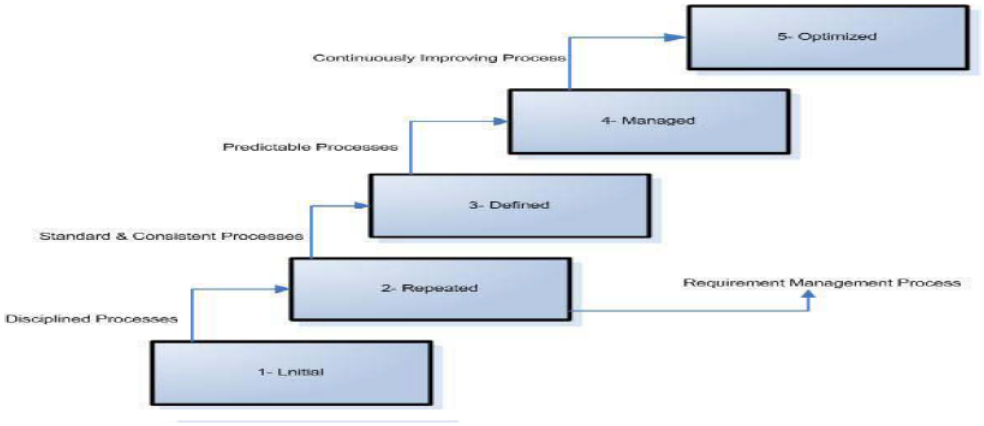

Figura 2.11: SW-CMM

Il Requirement Management Process, all'interno del «software process», ha lo scopo di creare una comprensione reciproca tra i customer e il fornitore del servizio ICT.

La proposta di Mir Ali Seyyedi, Mohammad Teshnehlab e Fereidoon Shams è quella di affiancare il SW-CMM al GQM in questo modo:

1. SW-CMM: definisce uno o più goal per ogni livello proposto
2. GQM: utilizza i goals prodotti da SW-CMM , propone question e istituisce metriche da utilizzare

Lo scopo del fuzzy system interposto tra i modelli è quello di dare una sorta di «human knowledge» alla metodologia.

I fondatori del metodo sostenevano che se il fuzzy model utilizzato è capace di apprendere ed espandere la propria conoscenza di base tramite regole ed è ampliato applicando la metodologia del sistema nervoso, si può arrivare ad un miglioramento delle regole e delle membership function [25].

Secondo il Software Engineering Institute (SEI, 2008), il CMM aiuta a integrare funzioni organizzative che tradizionalmente erano separate, a definire obiettivi e priorità per il miglioramento dei processi e a fornire una guida per la qualità dei processi come punto di riferimento per la valutazione dei processi attuali.

Il SW-CMM definisce un modello dei processi di sviluppo del software e un insieme di regole per il loro miglioramento.

Il Capability Maturity Model (CMM) utilizza una scala ordinale a 5 livelli per valutare le organizzazioni. I livelli di maturità vanno dallo stato «caotico e destrutturato» (livello 1) ad uno disciplinato ed in grado di "misurarsi per migliorare" (livello 5).

Per ogni livello di maturità sono definite le "process capabilities" ovvero un insieme di risultati attesi dalla messa in atto dei processi di sviluppo software a quel

determinato livello di maturità. Con l'eccezione del livello 1, ogni livello ha associate alcune «Key Process Areas» (KPA) che rappresentano i processi cruciali che una organizzazione deve implementare correttamente per poter raggiungere i risultati attesi per quel livello.

Rispetto ad ogni KPA sono identificati dei «goal» ovvero degli obiettivi da raggiungere per poter considerare soddisfatta l'implementazione.

Nel livello 2 (Repeated), i processi di gestione (project management) dei progetti più importanti sono definiti in modo da:

1. permettere la loro pianificazione,
2. tenere traccia dei costi (tempo e denaro),
3. analisi/convalida dei requisiti e definizione delle funzionalità richieste.

Nella realtà il SW-CMM è utilizzato da alcune organizzazioni per la scelta dei fornitori di un servizio: per esempio per partecipare a certe commesse il fornitore avrebbe dovuto dimostrare un certo livello di maturità (per esempio il livello 3), associato ad una certificazione conseguibile a seguito di una formale valutazione effettuata da personale a sua volta certificato dal SEI.

Ci sono aziende invece che usano questi modelli di maturità soltanto per capire a quale livello si trovano, per altre aziende conoscere la propria posizione rappresenta il punto di partenza per pianificare e sviluppare processi di miglioramento.

Il "Fuzzy Multi Agent Measurements" è stato adottato come caso di studio in un progetto di telecomunicazioni iraniano ed ha dimostrato la sua enfasi nella misurazione del raggiungimento dei goal che sono basati su un modello formale [25].

2.3.4.2 Quality Function Deployment

Per capire veramente gli aspetti innovativi della metodologia fuzzy proposta si prenderà come punto di riferimento un'altra metodologia di valutazione ovvero il «Quality Function Deployment » (QFD), altra tecnica trasversale tra le metodologie di valutazione presentate.

QFD si pone tra la tecnica di goal based (in quanto mette al centro gli obiettivi utilizzando «la voce del cliente» come linea guida per lo studio della valutazione) e la tecnica di decision making tipica del CIPP (considerando la valutazione come parametro di informazione per la pianificazione individuando i problemi, le esigenze e le opportunità per sviluppare gli obiettivi di un programma).

Il primo articolo che parla del QFD è apparso su una rivista americana nel 1983 ad opera di Kogure e Akao.

Il QFD costituisce uno strumento in grado di orientare il progetto di un prodotto verso le reali esigenze di chi lo utilizza e in questo senso rappresenta un potente mezzo per l'impostazione strutturata e finalizzata dei progetti. Normalmente il suo impiego

precede le attività di sviluppo (ma può essere anche utilizzato poi come mezzo di valutazione) [9].

Secondo la definizione data dal prof. Akao, padre della metodologia, il QFD è:

1. una serie di processi per trasformare le richieste del cliente/utilizzatore in caratteristiche che le rappresentino (caratteristiche della qualità), che definiscano la qualità e il valore del prodotto finito;

2. tali processi devono far sì che questa qualità sia sviluppata anche nei componenti e sottoinsiemi come pure nei singoli elementi del processo di produzione.

I risultati che si possono raggiungere con un'applicazione del QFD in tutte le fasi dello sviluppo possono essere sintetizzati in:

1. maggior soddisfazione del cliente, orientando al cliente stesso la progettazione ed enfatizzando la "voce del cliente" (the voice of the customer);

2. miglioramento del processo di sviluppo, riducendo i tempi ed i costi dovuti alle modifiche apportate al progetto;

3. miglioramento nella qualità, in quanto i prodotti sono pensati in ogni dettaglio per soddisfare le esigenze implicite ed attraenti del cliente

La qualità secondo QFD viene studiata all'interno di una "House of Quality" (HoQ), mostrata qui sotto:

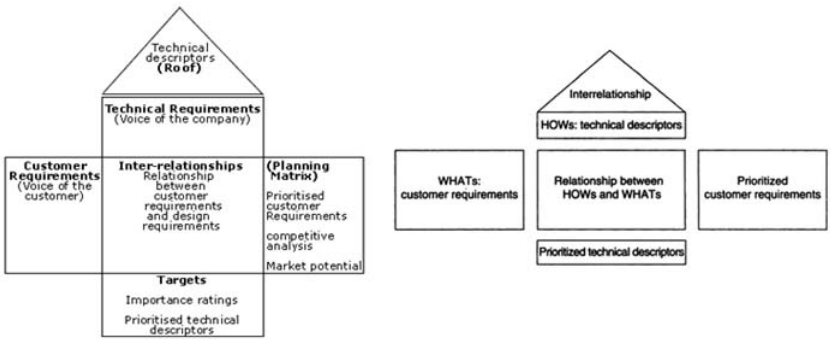

Figura 2.12: House of Quality

Come si può notare dalla Fig. 2.12 nella HoQ bisogna identificare i WHATS e gli HOWS:

1. i WHATS sono i bisogni e le esigenze dei clienti,

2. gli HOWS sono le caratteristiche tecniche che il prodotto deve possedere per realizzare i bisogni dei clienti, ovvero sono una lista di parametri/caratteristiche tecnico-ingegneristiche che descrivono i prodotti in termini misurabili (sono grandezze gestibili dall'azienda e un HOW può influenzare più WHATS).

In calce alle varie colonne vengono riportati gli indici assoluti e relativi dell'importanza della caratteristica tecnica, calcolati come somma dei prodotti tra gli indici di correlazione per i rispettivi indici di importanza del bisogno. In questo modo viene stilata una graduatoria e si evidenziano le specifiche tecniche più rilevanti.

Alcuni dei principali svantaggi connessi all'utilizzo del QFD sono:

1. tabelle eccessivamente estese che diventano difficili da maneggiare,
2. confusione nel trovare i requisiti dei clienti,
3. rischio di confodere le richieste del cliente con le caratteristiche di prodotto,
4. la difficoltà nel trovare la correlazione tra i bisogni del cliente e le caratteristiche del prodotto,
5. l'ouput è ancora una serie di dati da analizzare (non vi è un singolo output, un valore preciso).

2.4 Conclusioni

In questo capitolo si presentate le diverse metodologie di valutazione tra cui: il «Goal Based», il «Goal Free», il CIPP.

I loro approcci in termini di valutazione si sono rivelati di diverso genere.

Come detto precedentemente, Tayler propose modelli basati sull'obiettivo che è misurato attraverso prestazioni effettive credendo che gli obiettivi devono guidare tutte le procedure di selezione e di valutazione; di conseguenza gli obiettivi stanno in mezzo a tutte le attività svolte, i processi sviluppati e i prodotti generati.

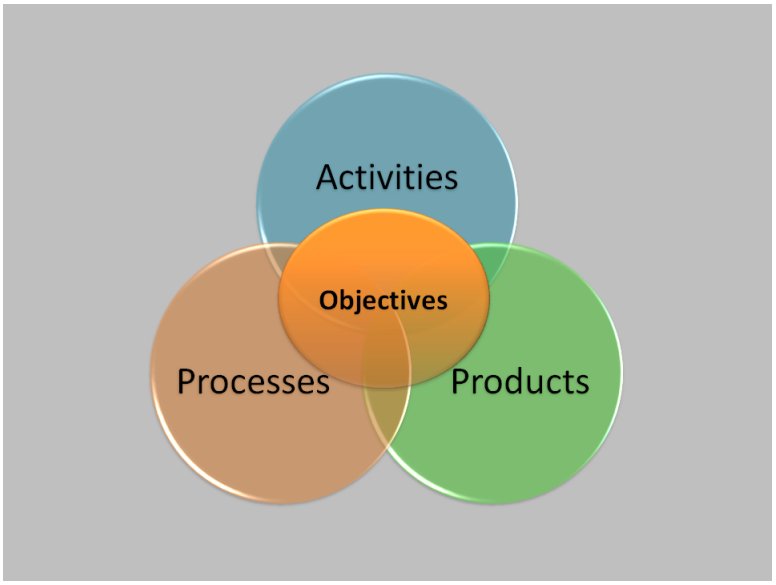

Figura 2.13: Tayler goal based

Scriven nel sua teoria di goal-free invece ritiene che gli obiettivi di un particolare programma non devono essere presi come un dato, ma esaminati e valutati. Il modello goal-free si concentra più sui risultati effettivi di un programma (outcome) o di un attività, piuttosto che solo gli obiettivi identificati.

Di conseguenza il modello di Scriven pone molto più al centro il processo di valutazione invece degli obiettivi.

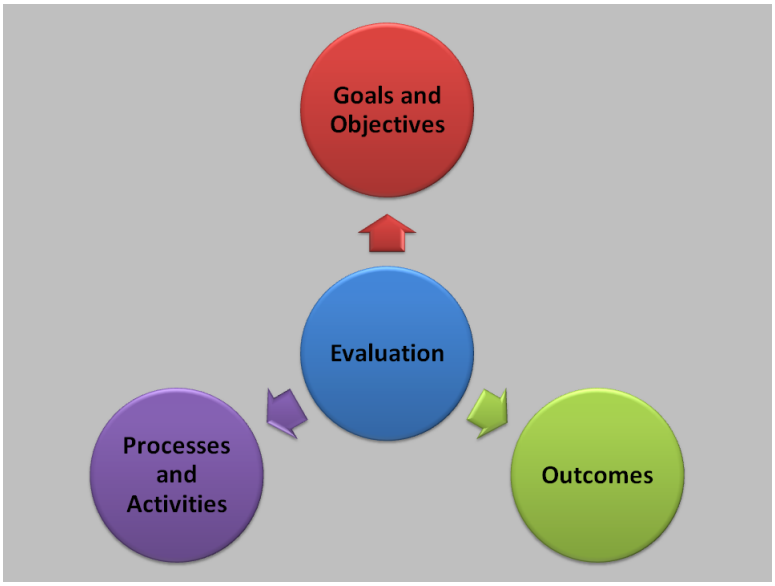

Figura 2.14: Scriven goal free evaluation

Nel CIPP invece la valutazione funge da parametro di informazione per la pianificazione individuando i problemi, le esigenze e le opportunità per sviluppare gli obiettivi di un programma. Il processo di valutazione valuta se sono necessarie delle modifiche all'interno del progetto e si concentra sui risultati di un programma o di un'attività consentendo di capire agli sviluppatori se continuare, terminare, modificare o riorganizzare. In questo caso si permette di determinare in corso d'opera se la progettazione del prodotto procede nel verso giusto.

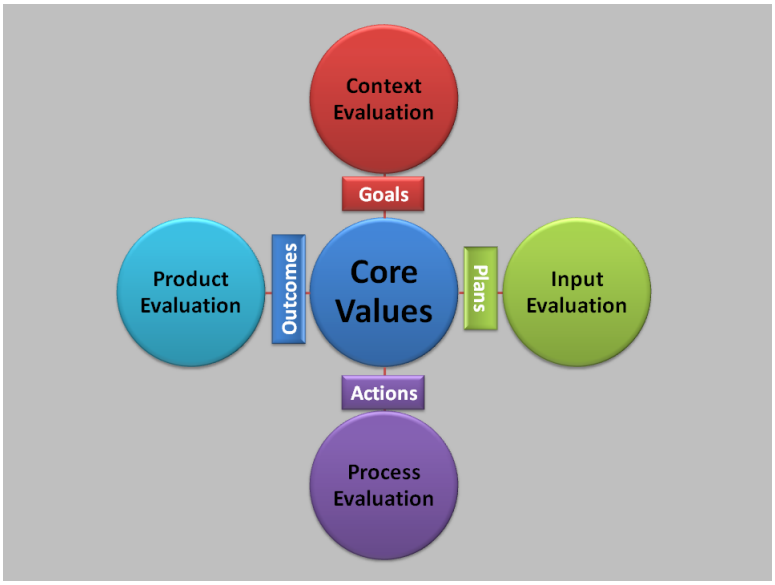

Figura 2.15: CIPP - processo di valutazione

Sono tutte metodologie diverse di valutazione, che cercano sempre di mettere al centro un qualcosa (che sia un obiettivo, un attività o un processo), che se applicate contemporaneamente si ottengono risultati differenti.

In questo caso quindi manca una sorta di omogeneità, un meccanismo che passi attraverso queste fasi: un punto di incontro tra i meccanismi goal model e decision making.

In questo capitolo si sono analizzate anche tutte le fasi di un processo di valutazione all'interno di un PM mostrando quali sono i meccanismi principali utili al processo e necessari a svolgere un analisi di progetto.

Ora che si sono presentate tutte le metodologie di valutazione presenti, nel prossimo capitolo verrà presentato il framework proposto in questo elaborato: un framework che taglia trasversalmente tutte queste tre metodologie e che punta a portare innovatività nel processo di valutazione, sopperendo alcuni limiti presenti nei meccanismi che implementano queste metodologie.

Tutte le fasi di questo nuovo framework, i vantaggi e i limiti che sopperirà saranno presentati nei capitoli successivi.

Capitolo 3

Metodologia fuzzy per il processo di valutazione

Questo capitolo contiene la presentazione della nuova metodologia fuzzy utile al processo di valutazione all'interno del PM.

Verranno spiegati in dettaglio i passi da seguire per la realizzazione del framework ed inoltre verranno presentate tutte le guideline e i meccanismi utili all'implementazione della metodologia: dalla stilazione guidata delle fuzzy rule al suo utilizzo.

3.1 Premessa

Le metodologie viste fino ad ora permettono di capire come negli anni il processo di valutazione abbia cambiato sia la forma sia lo scopo sia l'elemento fondante (cambiando le logiche di valutazione).

Come si è visto nel Capitolo 2, le logiche di valutazione possono essere orientate verso una tecnica di tipo goal based (di Tyler o Scriven) o verso una tecnica di tipo decision making (tipica del CIPP).

Ciò che si vuole raggiungere in tutte e due le logiche è lo stesso scopo ovvero quello di definire la qualità di un prodotto o di un prototipo ed essere in grado di dire oggettivamente se il software sviluppato riesca a rispettare le caratteristiche di progetto che si erano prefissate, ovvero che riesca a raggiungere la qualità desiderata dal project manager.

L'idea che ha portato alla creazione di questa nuova metodologia è stata quella di creare un framework che permetta di descrivere la qualità del software tenendo in considerazione sia la tecnica goal based orientata verso gli obiettivi, sia la tecnica decision making orientata verso l'utilizzo di un criterio di scelta, attuando così un processo di valutazione che «tagli trasversalmente» le due tecniche e nello stesso tempo ne dia una visione ancora più completa ed efficace.

Il framework presentato è basato sull'utilizzo di un approccio fuzzy della qualità che permette di tener conto delle possibili incertezze nello sviluppo e della tecnologia.

3.2 Introduzione alla metodologia

La misura della qualità di un processo software può essere determinata attraverso un approccio fuzzy che molto spesso è affiancato ad altre tecniche di modellizzazione [par. 2.3.4.1].

La metodologia che si andrà a presentare in questo capitolo propone di affiancare alle metodologie fuzzy e GQM, la modellistica delle Feature Model e diagrammi i* come formalismo per la specifica di goal all'interno di un CMM (ovvero, la fase dell'indicazione dei goal nel SW-CMM è presentata da i* modeling e le feature model).

Specificatamente, il KPA inerente la "gestione dei requisiti" («Requirement Management») in questa metodologia sarà determinato dall'utilizzo delle feature model e di i* modeling che permetteranno di analizzare e di definire le funzionalità richieste (in accordo con il livello 2 del CMM).

Il contesto sociale in cui i prodotti ICT sono applicati è estremamente complesso e ricco di incertezza, per questo motivo Eric S. Yu [37] propose la modellizzazione dei requisiti tramite i* che, anche al giorno d'oggi, offre un metodo efficace di analisi per descrivere ciò che non può essere realizzato meccanicamente. Egli propose l'SD model per le relazioni tra gli stakeholder e l'SR model come metodo per descrivere i goal e tutto ciò che serve per raggiungerli (anche in questa metodologia si seguirà questa logica).

Attraverso i diagrammi ad albero delle feature model si sarà in grado di identificare e capire quali sono le "commonality" e le "variability" di una grande categoria di software cercando di descrivere tutti i criteri di: obbligatorietà, opzionalità e alternativa all'interno di un dominio.

Il framework con la metodologia fuzzy si propone di creare componenti riusabili già nella stilazione dei goal con una vasta gamma di prodotti che abbiano una caratteristica di qualità in più già in partenza.

Per tutti questi motivi in questa metodologia la ricerca di goal sarà eseguita tramite feature model e diagrammi i*: l'idea sarà quella di associare questi modelli (feature model e modellistiche ì*) ad applicazioni con meccanismi fuzzy.

Non è la prima volta che nel settore dell'ICT le feature model sono affiancate a meccanismi tipicamente fuzzy, ad esempio nel 2003 Silva Robak e Andrzej Pieczynski dell'università di Zielona Gora (Polonia) proposero un metodo per modellizzare le variabilità in una Software Product Line (SPL) [33] assegnando dei pesi ad ogni scelta da poter effettuare (pesi stimati con valori fuzzy all'interno di un fuzzy set).

All'interno della metodologia di valutazione presentata in questo elaborato si useranno i fuzzy set sia per le variabili in input sia per le variabili in output; in questo caso si faranno riferimento a 3 fuzzy set per gli input ("low", "medium" e "high") e su 4 fuzzy set per gli output («bad», «good», «very good» e «excellent») in ordine crescente di qualità nella valutazione.

Il feature diagram verrà inteso come una rappresentazione visuale della conoscenza che può essere facilmente compresa anche dagli stakeholder e che può essere di grande aiuto nella fase di valutazione; motivo anche per cui è nata la scelta del suo utilizzo anche in questa metodologia fuzzy.

Il framework di questo elaborato si propone come una metodologia da utilizzare in fase di controllo e valutazione, ponendosi in modo trasversale all'interno dei principi goal based e decision making.

3.3 Panoramica della metodologia

In questo paragrafo verrà presentato il flusso informativo che si dovrà seguire per la costruzione completa del framework utile alla metodologia fuzzy.

Il flusso informativo descrive nell'ordine tutti i modelli da dover applicare all'interno della nuova metodologia fuzzy mostrando ciò che si avrà in input («in») ed in output («out») per ogni fase.

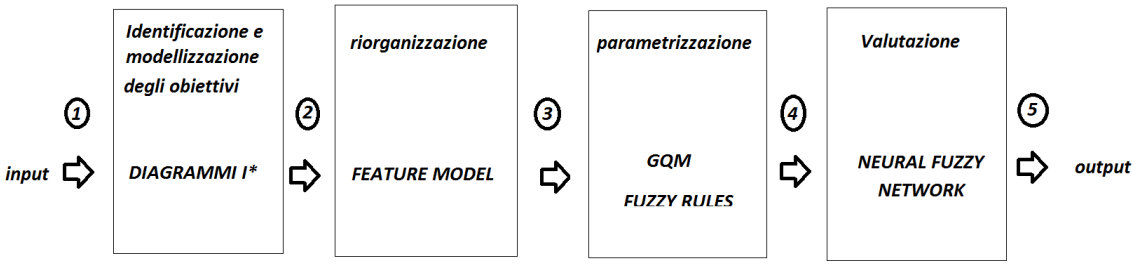

Figura 3.1: Flusso informativo

1. Input: l'input del framework è il documento di «Product RoadMap», ovvero quel documento dove vengono descritte quali sono le esigenze e quali sono le richieste provenienti dai potenziali customer del prodotto. In questa fase il documento sarà poco dettagliato e conterrà solo paragrafi di tipo descrittivo.

2. Identificazione e modellizzazione degli obiettivi: una volta ricevute in input le specifiche di progetto, bisognerà identificare e modellizzare gli obiettivi in maniera più dettagliata e strutturata. Il documento di RoadMap infatti è un documento che segue l'indirizzo implementativo del customer non tenendo in considerazione tutti gli aspetti implementativi (o di vincolo di progetto) e di tutti i dettagli che i requisiti richiedono per uno sviluppo ottimale.

   (a) In: Product Roadmap o documento di specifica.
   (b) Out: diagrammi SR e SD dell'i* modeling con tutti gli attori, i goal, softgoal, task e risorse del prodotto da sviluppare che permetteranno di esprimere, sotto forma di modello, quali sono gli obiettivi da raggiungere all'interno del progetto (o prodotto).

3. Riorganizzazione: presi in considerazione tutti gli attori, i goal, i softgoal, i task e le risorse ottenute dai diagrammi SR e SD, si deve ora cercare di definire per ogni attore quali sono le sue funzionalità e i compiti che possiede costruendo attorno ad esso le feature model.

   (a) In: gli attori, i goal, softgoal, task e risorse dei diagrammi SR e SD.
   (b) Out: feature model dei prodotti da sviluppare.

4. Parametrizzazione: consiste nella definizione delle metriche e nella costruzione delle fuzzy rule. Le fuzzy rule seguiranno dei meccanismi che verranno presentati successivamente.

   (a) In: feature model dei prodotti (ovvero si ha la definizione del prodotto con tutte le sue risorse, funzionalità e specifiche).
   (b) Out: metriche definite (tramite GQM) e fuzzy rule implementate.

5. Valutazione: costruzione della NFN seguendo le fuzzy rule implementate ed inizio del processo di valutazione.

   (a) In: fuzzy rule.
   (b) Out: NFN costruita.

6. Output: valutazione finale del prodotto.

   (a) In: NFN costruita.
   (b) Out: valutazione del prodotto completata.

3.4 Fase iniziale di sviluppo

L'ingegneria dei requisiti è un processo di definizione sia dei servizi che un cliente richiede ad un sistema sia degli scenari dell'uso del sistema, ma anche dei vincoli di sviluppo ed operativi. Definire i requisiti è la prima parte che si richiede nello sviluppo di un prodotto. I requisiti vanno scritti in modo tale da essere oggettivamente verificati nel prodotto finale cosicchè da rendere oggettiva anche la fase di valutazione.

Nella prima fase di sviluppo del framework è necessario quindi:

1. definire in modo chiaro quali sono i requisiti da rispettare (fase di «identificazione e modellizzazione degli obiettivi»),
2. costruire i diagrammi i* [par. 2.3.1],
3. delineare le feature model [par. 2.3.2];

per ogni feature model identificata dopodichè si procederà con la definizione delle fuzzy rule.

Ecco le linee guida per la loro costruzione:

1. definizione delle metriche di progetto e di prodotto,
2. fuzzificazione delle metriche di progetto e di prodotto,
3. stilazione delle fuzzy rule.

Verranno ora presentati i meccanismi e le guideline per creare le fuzzy rule necessarie al framework per la valutazione del prodotto finale o di un prototipo del prodotto stesso.

Innanzitutto si parte con una premessa che sarà utile per il seguito di questa trattazione, ovvero si definisce cos'è un «requisito funzionale» e cos'è un «requisito non funzionale»:

1. «Requisito funzionale»: tutto ciò che può essere descritto da casi d'uso. Requisito che esprime un'azione che il sistema dovrebbe eseguire, definendo sia lo stimolo sia la risposta (ossia sia l'input sia l'output) e guida la cosiddetta fase di sviluppo all'interno di un progetto descrivendo le funzionalità ed i servizi forniti dal sistema.
2. «Requisito non funzionale»: tutto ciò che non può essere descritto da casi d'uso. Non è collegato direttamente con le funzioni implementate dal sistema, ma piuttosto alle modalità operative e di gestione definendo vincoli sullo sviluppo del sistema.

3.5 Metriche

Misurare un software non è per niente facile da effettuare. Per misurazione si intende quel processo di assegnazione di simboli, solitamente numeri, per rappresentare un attributo dell'entità di interesse.

Sono necessari due elementi:

1. «entità» (l'oggetto o l'evento su cui si indaga),
2. «attributo» (caratteristica dell'oggetto).

Il termine «metrica» si utilizza per indicare la misura diretta (quando esiste uno strumento per effettuarla) o indiretta (quando non esiste uno strumento per effettuarla se non tramite una combinazione di misure dirette) di un qualche attributo di un'entità di interesse.

Si possono definire le metriche sia per i requisiti funzionali sia per i requisiti non funzionali.

3.5.1 Classificazione

Per determinare la qualità complessiva di un prodotto software concorrono due punti di vista (come definito nello standard ISO/IEC 9126 [1]):

1. «Misura esterna»: esprime il comportamento dinamico del software nell'ambiente d'uso.

2. «Misura interna»: esprime la misura in cui il codice software possiede una serie di attributi statici indipendentemente dall'ambiente di utilizzo e dall'utente.

Si definiscono le qualità interne ed esterne come l'insieme di caratteristiche sotto elencate (tra parentesi vengono definite le metriche utilizzate):

1. Affidabilità:

   (a) Maturità: capacità di evitare fermi della applicazione a seguito di errori nel software (MTBF),

   (b) Tolleranza errori: capacità di mantenere determinati livelli di prestazione in caso di errori (media e varianza di errori in relazione alle funzioni totali del sistema),

   (c) Recuperabilità: capacità di ripristinare livelli di prestazione predeterminati e di recuperare i dati a seguito di errori (tempo medio e varianza di ripristino del sistema dopo una interruzione).

2. Funzionalità:

   (a) Adeguatezza: presenza di funzioni appropriate per compiti specifici (numero di funzionalità implementate/numero di funzionalità richieste),

   (b) Accuratezza: capacità di fornire risultati o effetti in accordo con i requisiti (numero di risultati corretti/numero di risultati),

   (c) Interoperabilità: capacità di interagire con altri sistemi (numero di funzioni di import/export con altri applicativi),

   (d) Sicurezza: capacità di evitare accessi non autorizzati a programmi e dati (probabilità di accesso non autorizzato).

3. Efficienza:

   (a) Comportamento rispetto al tempo: tempi di risposta e di elaborazione richiesti per eseguire le funzioni richieste in determinate condizioni (Tempo medio di risposta),

   (b) Uso di risorse: quantità e tipo di risorse usate per eseguire le funzioni richieste in determinate condizioni (occupazione media e varianza dell'utilizzo di risorse, per esempio CPU),

4. Manutenibilità:

   (a) Analizzabilità: impegno richiesto per diagnosticare carenze o cause di fallimento, o per identificare parti da modificare (Effort),

   (b) Modificabilità: impegno richiesto per modificare, rimuovere errori o sostituire componenti (Effort),

   (c) Stabilità: capacità di ridurre il rischio di comportamenti inaspettati a seguito di modifiche (numero di errori scoperti da un test/FP),

   (d) Provabilità: impegno richiesto per validare le modifiche apportate al software (effort, dimensione del software).

5. Portabilità:

   (a) Adattabilità: capacità da parte del software di adattarsi a nuovi ambienti operativi applicando le sole azioni previste per questo motivo da parte del software stesso (effort: numero di mesi uomo necessari),

   (b) Installabilità: impegno richiesto per installare il software in un particolare ambiente (tempo medio e varianza di installazione),

   (c) Coesistenza: capacità del software di coesistere con altri software nel medesimo ambiente, condividendo risorse (numero medio di occupazione risorse di altri applicativi),

   (d) Sostituibilità: capacità di essere utilizzato al posto di un altro software per svolgere gli stessi compiti nello stesso ambiente (effort).

Analizzando le qualità interne ed esterne di un prodotto, si può notare che la maggior parte si occupa di definire dei requisiti ingegneristici o di progetto rispetto ai requisiti provenienti dai customer (ovvero dagli utilizzatori software).

Si può ora definire ciò che sarà «for customer» (FC) e ciò che sarà «for engineer» (FE):

1. FC: ciò che soddisfa il cliente.

2. FE: ciò che soddisfa gli «aspetti ingegneristici» del prodotto e che influenzerà indirettamente i customer nelle loro scelte di obiettivo, visibili a chi sviluppa.

Più propriamente si possono dividere le metriche in:

1. «metriche legate al prodotto ed al sistema»: che sono tipiche del prodotto stesso che si sta sviluppando,

2. «metriche legate i progetti»: che sono tipiche di ogni progetto e utili in ogni prodotto e sistema sviluppato o che si svilupperà.

Oppure un'altra suddivisione proposta proprio a livello di valutazione può essere in:

1. «metriche ex-ante»: metriche previsionali che si inseriscono nella prima fase del ciclo del progetto, utili prima del processo di valutazione ex-ante ovvero l'identificazione. La valutazione fatta in questa fase del ciclo di vita ha come finalità quella di valutare preliminarmente e con forte sforzo di astrazione il progetto così come identificato.

2. «metriche ex-post»: metriche posteriori, utili successivamente aver praticato il processo di valutazione ex -post ovvero la valutazione che permette di esprimere un giudizio circa il grado di raggiungimento dell'obiettivo generale prestabilito.

Di seguito verranno mostrate le metriche legate ai progetti (ovvero: affidabilità, SLOC, effort, produttività, costo, robustezza, scostamento di tempo, scostamento di effort) e metriche legate sia al prodotto sia legate ai progetti (metriche proventienti da GQM) entrambe che possono essere utilizzate come metriche ex-ante ed ex-post.

3.5.2 Introduzione alle metriche di progetto

Lo scopo delle metriche di progetto è quello di descrivere la qualità di un progetto sottoponendolo ad una stima dei costi, della forza lavoro da allocare, del tempo di sviluppo, della produttività e dell'affidabilità.

Quando si parla di metriche di progetto si parla di: linee di codice, costo, numero di sviluppatori, effort, affidabilità e produttività. A livello software è ragionevole pensare il costo all'interno dello sviluppo definito come funzione della dimensione del prodotto stesso [7]:

$$C = f(v)$$

Dove :
C è il costo di sviluppo
v è il volume (dimensione del progetto)

Per trovare quindi il costo di sviluppo, il problema sta nel trovare v e la forma della funzione f.

In questo ambito le metriche più utilizzate sono SLOC (numero di linee di codice) e i Function Points (FPs).

SLOC è la metrica più facile da tenere in considerazione in ambito software anche se si possono riscontrare delle ambiguità di utilizzo ovvero ci si pongono le seguenti domande:

1. nel conteggio delle SLOC si devono prendere in considerazione i commenti?
2. Come si contano le linee di codice con più statement?
3. Come si contano i contributi delle librerie? Senza considerare poi che ogni linguaggio di programmazione ha un'espressività diversa (esempio: uno statement Java equivale a più linee di Assembler).

Tendenzialmente si applicano delle scelte, tra cui quella di escludere: le righe di commenti, le linee bianche, le linee di libreria e quelle automaticamente generate da un eventuale tool; invece per quanto riguarda gli statement multi-riga si tende a contare uno statement in base al numero di righe occupate (implicando quindi il conteggio dei «return»).

Il primo passo da compiere per un'analisi dei costi di sviluppo è la previsione delle righe di codice in base all'analogia con altri progetti effettuati di cui si hanno già i dati.

Di particolare importanza sono infatti i «dati storici» che ogni azienda dovrebbe avere a disposizione, ovvero si dovrebbe costruire una sorta di database da cui derivare la produttività per ogni tipologia di progetto (produttività misurata in SLOC/MM, ovvero linee di codice/mesi uomo) che di conseguenza porteranno anche a stimare l'effort della fase di sviluppo [12] misurato in MM.

Si può notare però il fatto che più un programmatore utilizza un linguaggio di basso livello e più il programmatore può risultare produttivo; ad esempio, confrontando 800 SLOC/MM utilizzate per Assembler e 200 SLOC/MM per una programmazione Java sembrerebbe che il programmatore assembler sia molto più performante di un programmatore Java, ma come si sa il tempo per lo sviluppo in Java è complessivamente più breve avendo l'assurdo che il programmatore Assembler abbia migliore produttività anche se consegnerebbe dopo il proprio lavoro.

La metrica SLOC quindi viene utilizzata soprattutto per tracciare la «stima dei costi» ma non la «stima della produttività» (che invece utilizza i FPs).

I FPs furono introdotti da Allan Albrech [2] ponendo in primo piano «cosa fa» un programma rispetto a «come è fatto», indipendentemente quindi dal linguaggio di programmazione e dalla tecnologia utilizzata, quindi la produttività può essere confrontata tra diversi linguaggi di programmazione.

Si tratta di un metodo empirico usato ampiamente nella programmazione OO e si basa soprattutto sulla misurazione delle funzioni consegnate all'utente definendo 5 differenti «Function Type» [15]:

1. «External Input» (EI): dati in ingresso dall'utente o da altre applicazioni
2. «External Output» (EO): dati restituiti in uscita
3. «External Inquiry» (EQ): una coppia input/output unica, dove un input online produce la generazione di una risposta immediata del software, sotto forma di output online
4. «Internal Logic File» (ILF): un raggruppamento unico di dati logicamente correlati, o di informazioni di controllo, identificabile dall'utente, usato ed aggiornato dall'applicazione
5. «External Interface File» (EIF): un unico gruppo di dati logicamente correlati, o di informazioni di controllo, identificabile dall'utente, che risiede esternamente all'applicazione ma fornisce dati usati dall'applicazione

I FPs sono un meccanismo molto soggettivo e si basano su informazioni che possono essere difficili da reperire che non hanno un vero e proprio significato fisico, per questo motivo spesse volte si propongono di utilizzare metodi alternativi per lo studio dell'effort tra cui troviamo il metodo COCOMO.

COCOMO è composto da una serie di equazioni e parametri statistici e permettono di calcolare l'effort in funzione della dimensione dei sorgenti. COCOMO è l'abbreviazione di «COnstructive COst MOdel» ed è un modello matematico creato da Barry Boehm utilizzato da chi progetta software per stimare alcuni parametri fondamentali come il tempo di consegna e i mesi-uomo necessari per lo sviluppo di un prodotto software [4] .

Comprende tre modelli diversi (con livello di dettaglio crescente):

1. Basic: applicato a priori
2. Intermediate: si applica dopo che i requisiti sono stati specificati
3. Advanced: si applica dopo che il design è stato completato

Ed ha tre modalità diverse [4]:

1. Organic: team relativamente piccoli che sviluppano software in un ambiente familiare e per uso interno (progetti piccoli)
2. Semi-detached: fase intermedia tra organic ed embedded. Di solito fino a 300 KLOC

3. Embedded: operando all'interno di vincoli stretti, il prodotto è fortemente legato alla «complessità di hardware, software, regole e procedure di funzionamento»

COCOMO utilizza due equazioni per calcolare lo sforzo in mesi uomo (MM) e il numero di mesi stimati per il completo progetto (TDEV)

$$MM = a(Size)^b$$

$$TDEV = c(MM)^d$$

I coefficienti a, b, c e d differiscono a seconda di quale modalità si sta usando [26]:

Tabella 3.1: COCOMO: parametrizzazione dei coefficienti

Modalità	**a**	**b**	**c**	**d**
Organic	2.4	1.05	2.5	0.38
Semi-detached	3.0	1.12	2.5	0.35
Embedded	3.6	1.20	2.5	0.32

COCOMO è il metodo più diffuso di stima dei costi del software anche se è consigliabile usare anche un altro metodo che differisca molto da COCOMO, in modo tale che il progetto venga esaminato da più angolazioni [14].

Oltre ai parametri presentati precedentemente, in una valutazione della qualità di progetto si fa spesso riferimento anche alla caratteristica di «affidabilità», ovvero a quell'insieme di attributi che sono in relazione con la capacità del software di mantenere un livello di prestazioni in determinate condizioni e per un determinato periodo di tempo.

Tra i parametri di affidabilità si trova MTBF («Mean Time Beetween Failure») ovvero il «tempo medio tra i guasti», che è applicabile non solo ad applicazioni software ma anche a dispositivi meccanici, elettrici ed elettronici.

Per MTBF si intende la somma di due tempi: MTTF (Mean Time To Failure) e MTTR (Mean Time To Repair):

$$MTBF = MTTF + MTTR$$

L'MTTF non è una funzione del tempo, ma è una costante (dimensionale, con dimensione di un tempo) che descrive sinteticamente l'affidabilità del componente mediante un numero deterministico ovvero dispositivi "molto affidabili" avranno valori alti di MTTF.

MTTR invece rappresenta il tempo impiegato per la riparazione di un eventuale guasto che messo in relazione con MTTF può essere interessante studiare l'affidabilità anche tramite un parametro A definito come:

$$A = \frac{MTTF}{MTTF + MTTR} = \frac{MTTF}{MTBF}$$

Se A tende a 1 (MTTF>MTTR) allora il sistema può essere considerato affidabile.

I metodi di previsione dell'MTBF calcolano un valore unicamente in base a un progetto di sistema, solitamente attuato nella fase iniziale del ciclo di vita del prodotto. In presenza di informazioni sufficienti sull'utilizzo reale del prodotto, i metodi di previsione non dovrebbero essere impiegati. All'opposto, dovrebbero essere utilizzati i metodi di stima dell'MTBF perché tali metodi rappresentano le misurazioni effettive dei guasti. I metodi di stima dell'MTBF calcolano un valore in base all'osservazione di un campione di sistemi simili e in genere vengono utilizzati in seguito all'utilizzo effettivo di una vasta popolazione di prodotto. La stima dell'MTBF è il metodo più utilizzato, principalmente perché si basa su prodotti reali in uso effettivo.

Tutti questi metodi sono di natura statistica, ovvero forniscono solo un valore approssimativo dell'MTBF reale: non esiste un metodo standardizzato per l'intero settore. Pertanto è di importanza fondamentale che il produttore comprenda e scelga il metodo migliore per una determinata applicazione.

Queste tipologie di misurazioni risultano molto interessanti anche se sono difficili da stimare/misurare in quanto il sistema deve essere attivo, ovvero occorre misurare il tempo o avere un log di sistema per cercare di documentare il tutto. Sono metriche ingegneristiche e come si può notare sono collegate alla densità degli errori in fase di testing ma in modo non immediato.

In tutte queste metriche, raccogliere i dati utili alla qualità richiede di interagire con il team di sviluppo e un tipico errore che si riscontra è quello di raccogliere una quantità di dati superiore alle necessità per mancanza di obiettivi precisi (una statistica del 1983 mostra che presso la NASA il costo della raccolta di dati è arrivato al 15% del costo del progetto [31]).

Il GQM raccoglie le informazioni in maniera razionale permettendo di progettare le procedure per la raccolta (automatica o manuale) e analisi dei dati.

Per quanto riguarda il GQM è stata fatta una trattazione più ampia precedentemente, capendo già la sua importanza per la definizione di metriche oggettive.

In seguito si suddivideranno le metriche in categorie utili a studiare la qualità del prodotto o del progetto anche in relazione a ciò che si è descritto precedentemente.

Si identificano le seguenti categorie:

1. la dimensione,

2. le performance,

3. le risorse,

4. l'impegno.

3.5.3 Dimensione

Per studiare la dimensione di un progetto o di un prodotto in questo framework viene proposto di utilizzare come metrica SLOC.

3.5.3.1 SLOC

La metrica SLOC è finalizzata all'analisi del codice prodotto ed appartiene alla categoria delle metriche dimensionali che permette di misurare la lunghezza di un prodotto software in termini di numero di linee di codice.

Il numero di linee di codice dipende fortemente dal tipo di prodotto e dalla quantità di funzionalità da implementare. Per comprendere ciò tutto ciò, di seguito viene mostrata una tabella dove vengono presentati diversi sistemi operativi in relazione ai loro corrispettivi SLOC:

Tabella 3.2: SLOC: esempio

Prodotto	**SLOC**
Windows XP	**45 M** [24]
Windows Server 2003	**50 M** [24]
Linux kernel 3.6	**15.9 M** [23]
Mac OS X 10.4	**86 M** [19]

Per creare livelli fuzzy di questa metrica ci si dovrà quindi affidare alle esperienze precedenti ed istituire un livello massimo di numero di SLOC che rappresenteranno il 100%; dopodichè si stabilirà una scala di valori fuzzy affidandosi ai livelli «LOW», «MEDIUM» e «HIGH».

Consapevoli delle limitazioni tipiche di questa particolare metrica, la metrica SLOC è stata maggiormente impiegata nella misurazione indiretta di altri attributi come ad esempio: la probabilità di presenza di errori nel programma, il tempo necessario per completare lo sviluppo del prodotto software e la stima dei costi di un particolare progetto.

La stima delle SLOC infatti non è mai precisa in quanto dipende molto dalla capacità dello sviluppatore e dalla propria esperienza rispetto a progetti precedenti: uno sviluppatore con più anni di esperienza presumibilmente saprà fare una stima più precisa del valore massimo di SLOC rispetto ad uno sviluppatore alle «prime armi».

Dati questi limiti della metrica SLOC è quindi necessario introdurre i Function Points (FPs).

Quantitative Software Management, Inc. (QSM) è l'organizzazione leader nei processi di valutazione di software che definisce una tabella guida per la ricerca dei FPs avendo a disposizione le SLOC sviluppate.

Di seguito viene presentato un scorcio di tabella proposta da QMS [27]

Language	QSM SLOC/FP Data			
	Avg	Median	Low	High
C *	97	99	39	333
C++ *	50	53	25	80
C# *	54	59	29	70
HTML *	34	40	14	48
J2EE *	46	49	15	67
Java *	53	53	14	134
JavaScript *	47	53	31	63
JCL *	62	48	25	221

Figura 3.2: Function Point Language table

I Function Point misurano esclusivamente i requisiti funzionali di un prodotto software di conseguenza, conoscendo approssimativamente quante righe di codice si sono sviluppate o si svilupperanno per raggiungere una funzionalità, seguendo la tabella sarà possibile trovare il numero di FPs a seconda del linguaggio di programmazione utilizzato in tutti i livelli presentati (la media, la mediana, il valore minimo e massimo).

La metrica SLOC è legata a tutti i progetti (non solo ad un singolo sviluppo o prodotto) e può essere sia di tipo ex-ante sia di tipo ex-post.

3.5.4 Performance

Per studiare le performance di un progetto o di un prodotto in questo framework viene proposto di utilizzare l'affidabilità e la robustezza.

3.5.4.1 Affidabilità

Precedentemente si è visto che l'MTBF dipende dal valore dei parametri MTTR e MTTF.

Si era definito un valore A che più questo valore si avvicina ad 1 e più si sarà soddisfatti del prodotto, quindi più il tempo medio di funzionamento fra un guasto e il successivo si avvicinerà al tempo medio al verificarsi di un guasto (MTTF=MTBF) e più si sarà soddisfatti e di conseguenza il software sarà più affidabile. Partendo

da questo valore si può quindi definire la «scala di affidabilità» per questa metrica, proponendo dei livelli linguistici che identifichino i livelli fuzzy.

In seguito viene mostrata la tabella per i fuzzy set dell'affidabilità.

Tabella 3.3: Affidabilità: livelli fuzzy

A	Fuzzy level
$(0\% - 30\%)$	**LOW**
$> 30\% \,\&\, \leq 80\%$	**MEDIUM**
$> 80\% \,\&\, \leq 100\%$	**HIGH**

Questi valori saranno molto variabili a seconda se si vogliono fuzzy set più restrittivi o meno. Nel caso di studio presentato successivamente si avranno valori più rigidi, con intervalli non omogenei.

La metrica è legata a tutti i progetti (non solo ad un singolo sviluppo o prodotto) e può essere sia di tipo ex-ante sia di tipo ex-post.

3.5.4.2 Robustezza

La robustezza rappresenta la capacità del prodotto di continuare a funzionare senza degrado nelle sue prestazioni anche in condizioni particolari (esempio: uso continuativo 24 ore su 24, 7 giorni su 7), oppure in condizioni limite (esempio: 500 utenti collegati contemporaneamente). Si misurano le prestazioni del prodotto (tempi di risposta ed utilizzo delle risorse) e si verifica che non ci sia degrado nelle prestazione [12].

Per prestazione si può intendere:

1. il tempo medio di risposta di una transazione;
2. percentuale di utilizzo delle linee di trasmissione, delle risorse;
3. probabilità di accessi non autorizzati (sicurezza);
4. probabilità di errore;
5. tempo di ripristino errore.

Ognuna di queste metriche deve essere fuzzificata secondo la propria esperienza (affidandosi ai livelli «LOW», «MEDIUM» e «HIGH»), quindi attraverso valutazioni empiriche.

La metrica è legata a tutti i progetti (non solo ad un singolo sviluppo o prodotto) e può essere sia di tipo ex-ante sia di tipo ex-post.

3.5.5 Risorse

Per studiare le risorse che si hanno a disposizione in un progetto o in un prodotto, in questo framework si fa riferimento alla produttività e al costo.

3.5.5.1 Produttività

A livello ICT, la produttività rappresenta le risorse impiegate nelle diverse fasi del progetto.

Si parla quindi di produttività di progetto oppure di produttività di analisi e disegno oppure ancora di produttività di programmazione. E' misurata in termini di numero di linee di codice (SLOC) o di punti funzione (FPs) sviluppati da una persona nell'unità di tempo stabilita (mese, settimana, giorno, ora), quindi la produttività di una fase specifica è misurata in numero di linee di codice sviluppate per mese-uomo (o numero di casi di test eseguiti per giorno-uomo o nel numero di pagine scritte per giorno-uomo).

Essa è utilizzata per valutare lo sforzo richiesto per lo sviluppo del progetto a fronte delle sue dimensioni [3]:

$$Produttività = SLOC/MM$$

anche se più precisamente bisognerebbe utilizzare la seguente formula:

$$Produttività = FPs/MM$$

La metriche deve essere fuzzificata secondo la propria esperienza (affidandosi ai livelli «LOW», «MEDIUM» e «HIGH»), quindi attraverso valutazioni empiriche. La produttività è legata a tutti i progetti (non solo ad un singolo sviluppo o prodotto) e può essere sia di tipo ex-ante sia di tipo ex-post.

3.5.5.2 Costo

Quando si parla di «costo software» ci si riferisce al costo di sviluppo ovvero si analizzano le risorse a disposizione e si delineano dei costi massimi da poter praticare.

Il costo di un software è direttamente collegato alla capacità economica/finanziaria dell'azienda, dalla dimensione del progetto e delle risorse e tecnologie necessarie allo sviluppo.

Per lo studio di questa metrica quindi sarà necessario fare delle analisi più approfondite 4 livelli:

1. «budget»: disponibilità finanziaria dell'azienda fornitrice del servizio,
2. «progetto»: dimensione del progetto in termini fuzionali e di requisiti,
3. «risorse umane»: sviluppatori a disposizione o che si intendono utlizzare,
4. «risorse tecnologiche»: risorse tecnologiche-informatiche a disposizione (calcolatori a disposizione, database, ambienti SAAS ed altre risorse del genere)

Per ogni livello è possibile stabilire quali sono i limiti minimi e massimi entro a quale operare a seconda della propria capacità aziendale.

A livello «budget» ci saranno per esempio organizzazioni che indicheranno a 1.000.000 € il budget massimo di spesa per lo sviluppo di un prodotto, ci saranno altre organizzazioni che invece stanzieranno 100.000 € per lo sviluppo dello stesso prodotto. Il livello «budget» è quindi qualcosa di più soggettivo e dipende dalle capacità economiche aziendali.

Per quanto riguarda gli altri parametri si può darne invece una misura più oggettiva, ovvero prenderanno spunto da parametri stimati come per esempio la dimensione del progetto.

In questa metodologia si utilizzarà come metrica di costo il livello delle «risorse umane», ovvero il numero di sviluppatori necessari per la costruzione del prodotto. Per calcolare per esempio il numero di sviluppatori che servono per completare il lavoro in un tempo T (mesi), sapendo che per portalo a termine si impiegherà X (Mesi/Uomo), si avrà bisogno quindi di N sviluppatori

$$N = \frac{X}{T}$$

N sarà il valore minimo di sviluppatori per arrivare ad avere il prodotto pronto nei termini di tempi stabiliti; va da sè che se N aumenta, T diminuirà a parità di X. Di conseguenza avremmo che:

$$N_{min} = \frac{X}{T_{max}}$$

$$N_{max} = \frac{X}{T_{min}}$$

Si potrà quindi fuzzificare il valore:

Tabella 3.4: Costi e Tempi: livelli fuzzy

N e T	**Fuzzy**
min	**LOW**
min<N o T<max	**MEDIUM**
max	**HIGH**

Naturalmente più risorse si hanno già a disposizione e più il costo sarà inferiore, in quanto non si devono sostenere costi aggiuntivi. La metrica è legata a tutti i progetti (non solo ad un singolo sviluppo o prodotto) e può essere sia di tipo ex-ante sia di tipo ex-post.

3.5.6 Impegno

Per studiare l'impegno impiegato in un progetto o in un prodotto, in questo framework si fa riferimento all'effort, allo scostamento di effort e allo scostamento di tempo.

3.5.6.1 Effort

L' effort è quella metrica che permette di misurare lo «sforzo», ovvero la quantità di lavoro necessaria a completare un progetto ed è normalmente espresso in termini di giorni/uomo o mesi/uomo. Come descritto precedentemente, l'effort in un progetto software è legato inesorabilmente alla quantità di SLOC da sviluppare da parte del programmatore. Per stabilire quindi quali livelli proporre per l'effort, è necessario aver fatto una stima di quale sia la dimensione del nostro progetto ed aver già fuzzificato i valori di dimensione (sia in termini di SLOC o di FP).

Decidendo poi a quale livello di COCOMO lavorare (Organic,Semi-detached,Embedded) si stimeranno i parametri di effort in mesi uomo.

Si potranno quindi identificare i valori LOW, MEDIUM e HIGH di effort.

La metrica è legata a tutti i progetti (non solo ad un singolo sviluppo o prodotto) e può essere sia di tipo ex-ante sia di tipo ex-post.

3.5.6.2 Scostamento di effort

Al pari dello scostamento di tempo, un'altra metrica molto importante è lo scostamento di effort (ovvero dei mesi-uomo consuntivati rispetto a quelli pianificati).

Anche in questo caso l'obiettivo è quello di dare un indice oggettivo che ne permetta di capire se vi è stato un sovraccarico in termini di mesi uomo.

Analogamente alla metrica di scostamento di tempo, si indicherà con S_E lo scostamento di effort calcolandolo nel modo seguente:

$$S_T = \frac{Effort\,consuntivato}{Effort\,pianificato}$$

Un valore maggiore di 1 sarà un indicatore negativo di prodotto o del prototipo sviluppato in temini di quantità di lavoro necessaria a completare un progetto. In linea di massima più il valore di S_T è basso e più la soddifazione dell'azienda fornitrice e dei customer saranno elevati.

La metrica deve essere fuzzificata in base all'esperienza (affidandosi ai livelli «LOW», «MEDIUM» e «HIGH») ed è legata a tutti i progetti (non solo ad un singolo sviluppo o prodotto) ed è di tipo ex-post.

3.5.6.3 Scostamento di tempo

Un'altra metrica molto utile a livello di valutazione è lo scostamento del tempo di sviluppo, ovvero un'indicazione oggettiva che metta a confronto ciò che è stato consuntivato rispetto a ciò che si è pianificato.

Questa metrica si pone come obiettivo quello di mostrare al project manager se è proceduto tutto come si era pianificato o se in fase di sviluppo si è andati oltre del limite di tempo richiesto causando un ritardo.

E' importante analizzare le cause dei ritardi per verificare se il problema è stato generato da stime iniziali non corrette, da una pianificazione non realistica o da una conduzione non adeguata.

Si indicherà la metrica con S_T calcolandola nel modo seguente:

$$S_T = \frac{Tempo\,consuntivato}{Tempo\,pianificato}$$

Un valore maggiore di 1 sarà un indicatore negativo di prodotto o del prototipo sviluppato. In linea di massima più il valore di S_T è basso e più la soddifazione dell'azienda fornitrice e dei customer saranno elevati.

La metrica deve essere fuzzificata in base all'esperienza (affidandosi ai livelli «LOW», «MEDIUM» e «HIGH») ed èè legata a tutti i progetti (non solo ad un singolo sviluppo o prodotto) ed è di tipo ex-post.

3.5.7 Altre metriche

Le metriche non ancora analizzate sono quelle provenienti dal GQM e che anch'esse verranno fuzzificate come le metriche presentate precedentemente.

Le metriche di progetto provenienti da GQM che non sono riconducibili alle metriche precedenti ricadono sotto la categoria di «altre metriche».

3.5.7.1 Metriche GQM

Purtroppo risulta difficoltoso generalizzare tutte le metodologie di fuzzificazione per i metodi GQM in quanto dipendono fortemente dal goal trovato e dal prodotto sviluppato.

Nel caso di studio presentato nel capitolo successivo si analizzeranno queste metriche proponendo livelli fuzzy anche per metriche tipiche del goal e non generali (ovvero metriche più legate al singolo prodotto) e possono essere sia di tipo ex-ante sia di tipo ex-post.

Anche queste metriche devono essere fuzzificate in base all'esperienza (affidandosi ai livelli «LOW», «MEDIUM» e «HIGH»).

3.5.8 Tabella riassuntiva delle metriche

Ecco una tabella riassuntiva dove vengono riportate tutte le metriche presentate fino ad ora con le loro caratteristiche, ovvero se sono metriche legate ai progetti, al singolo prodotto (o sviluppo), se sono ex-ante e/o ex-post.

Ogni metrica appartiene ad una categoria ovvero: dimensione, performance, risorse, impegno.

Tabella 3.5: Metriche: tabella riassuntiva

Categoria	Metrica	progetti	prodotto	ex-ante	ex-post
Dimensione	**SLOC**	•		•	•
Performance	**A**	•		•	•
	Robustezza	•		•	•
Risorse	**Produttività**	•		•	•
	Costo	•		•	•
Impegno	**Effort**	•		•	•
	Scost. tempo	•			•
	Scost. effort	•			•
Altre	**Altre metr. GQM**	•	•	•	•

3.6 Costruzione delle fuzzy rule

Ora che si sono definiti i livelli fuzzy delle metriche di progetto e di prodotto, si procederà a descrivere come costruire le fuzzy rule.

I confronti che si devono effettuare per aver un buon risultato in termini di qualità sono da fare tra ciò che richiede il cliente e ciò che richiede l'ingegnere.

Un prodotto software, per essere in grado di offrire un servizio di qualità, deve provenire da una necessità o da una mancanza.

Per l'industria ICT i servizi vengono offerti ad ogni tipologia di settore (meccanico, industriale, dei servizi, sociale, medico ed in altri svariati ambiti) senza distinzione. Ogni settore per semplicità si potrebbe definire un «customer» dei servizi offerti.

I customer forniscono l'input e le persone competenti del settore restituiscono l'output considerando una serie di vincoli presenti.

In seguito alla costruzione delle feature model del prodotto, si è in grado di avere una rappresentazione compatta e di capire le funzionalità e gli obiettivi che ci permetteranno di stilare le fuzzy rule.

Gli obiettivi sono così utilizzati per trovare le metriche GQM e per delineare le fuzzy rule: stabilito un goal per la feature, si stabiliscono le sue metriche e si descrivono i suoi vincoli progettuali attraverso delle regole.

Esempio: Ad un'azienda di pulizie è stato fornito un software che permette, tramite un portale adibito, di avere informazioni real time di ciò che sta accadendo nei vari edifici dove gli addetti alle pulizie sono coinvolti. Gli addetti hanno la possibilità di comunicare il loro inizio/fine lavoro in una particolare sede tramite un dispositivo RFID, il cui tag verrà memorizzato all'interno del programma al momento della timbratura, permettendo quindi ai reponsabili di identificare dove e quando l'addetto alle pulizie ha effettuato il suo lavoro. Uno dei goal che questo prodotto software deve soddisfare e che l'azienda di pulizie richiede, è la «Comunicazione efficace tra dispositivi differenti» (dispositivo RFID e server). Si vuole quindi fare un focus di qualità sulla comunicazione e il coordinamento tra le risorse, ponendosi la domanda (Question) «Come deve essere la comunicazione all'interno dell'applicativo per essere efficace?». Ecco le metriche a disposizione:

1. M1: numero medio di informazioni scambiate (robustezza),
2. M2: risorse a disposizione (costo),
3. M3: velocità di scambio informazioni (robustezza),
4. M4: tempi di risposta (robustezza),
5. M5: qualità e quantità dei dati (affidabilità),
6. M6: altre metriche a nostra disposizione,

Il «customer» proporrà il goal, l'«engineer» avrà il compito di analizzarlo tramite fuzzy rule utilizzando metriche del software e metriche di soddifazione del cliente. L'output delle fuzzy rule (più propriamente del valore del «conseguente» della regola) indicherà proprio questo livello, ovvero il raggiungimento dell'obiettivo.

Ogni feature possederà più obiettivi, alcuni anche collegati tra loro unendo le loro metriche e dando origine ad un output globale del sistema (mettendo in «union» le nostre fuzzy rule).

Ad esempio il goal «Capacità di avere le timbrature real time» sarà l'unione del goal precedente («Comunicazione efficace tra dispositivi differenti») e del goal «La stabilità dei dati».

Questa analisi multi-obiettivo è cio che cambia rispetto al meccanismo QFD, che fornisce invece una valutazione mono-obiettivo (obiettivi valutati uno per volta).

Nella HoQ mostrata qui sotto, si è evidenziata la sezione dove viene analizzata la qualità di un prodotto: come si può notare, la qualità è definita come un «WHAT»

intersecato ad un «HOW» e non come un «WHAT» intersecato ad un «WHAT», ciò che invece permetterebbe il framework proposto.

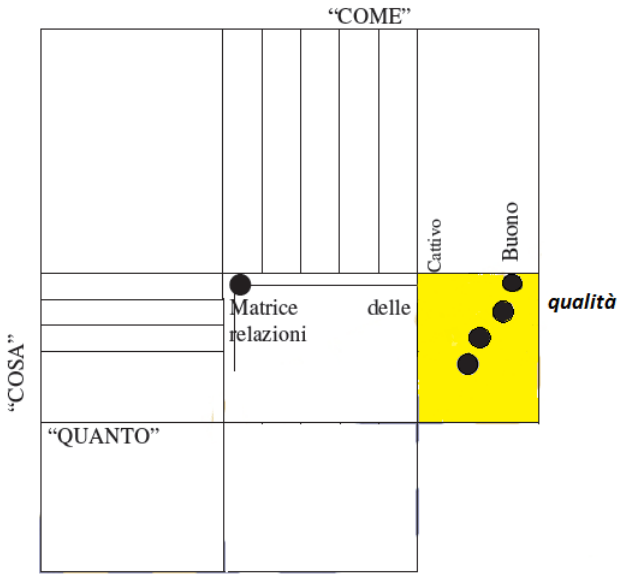

Figura 3.3: QFD Quality

In una HoQ, non si avrà modo di fare una valutazione complessiva del prodotto in maniera oggettiva, come si analizzerà poi più dettagliatamente nel Capitolo 5.

3.7 Costruzione della NFN

In conclusione si deve procedere con la costruzione della rete NFN.

Il vantaggio principale che offre una rete neurale è legato soprattutto all'esperienza che acquisisce durante la sua esecuzione. Infatti, definire tutte le possibili combinazioni di metriche di progetto o di prodotto nelle fuzzy rule è quasi impossibile in quanto richiederebbe un lavoro molto dispendioso (si avrebbero tutte le possibili combinazioni di un numero n di metriche da utilizzare).

Per ovviare al problema si è deciso di utilizzare la NFN in quanto farà una stima approssimativa dei valori trovati assegnandoli sempre a dei livelli fuzzy, completando in un certo senso tutte le regole mancanti.

Per esempio se si avesse un livello «MEDIUM» di effort e questo livello non facesse matching con il livello delle altre metriche nelle fuzzy rule definite, la NFN definirà in base all'esperienza passata se far ricadere l'effort in un livello «LOW» o in un livello «HIGH» in modo tale da ricondurci ancora in una delle fuzzy rule definite.

Per costruire la NFN si può fare riferimento ad un tool, tra cui si ricorda Neuroph ovvero un framework Java per lo sviluppo di reti neurali che consiste in li-

brerie Java e di una GUI che fa da editor chiamata Neuroph studio (download: http://neuroph.sourceforge.net/download.html).

Usare Neuroph è molto semplice, ed è necessario solo avere installato sul proprio computer la versone 1.7 della Java VM, il resto verrà fornito dal download.

Una volta installato è possibile:

1. creare la propria neural network,
2. impostare gli input,
3. definire i layer e le fuzzy rule per determinare tutti gli output.

E' un tool semplice da utilizzare che può risultare molto utile al fine dell'utilizzo di questa metodologia.

Capitolo 4

Caso di studio

In questo capitolo si analizzerà il caso di studio su cui si applicherà la metodologia fuzzy presentata nel Capitolo 3.

Si seguiranno tutte le fasi descritte precedentemente inerenti il flusso informativo, arrivando fino alla fase di valutazione (che sarà presentata poi dettagliatamente nel Capitolo 5).

Per motivi di riservatezza aziendale tutti i dati presentati da ora in poi sono a livello prototipale ed esemplificativo che possono essere quindi anche non conformi al prodotto finale in produzione; tutto ciò è per mantenere una riservatezza aziendale dei propri sviluppi.

4.1 Panoramica: ZTimesheet

Il software di cui si farà riferimento e su cui si costrurà il framework con la metodologia fuzzy è «ZTimesheet».

ZTimesheet è il software Zucchetti per l'efficiente organizzazione delle risorse nelle aziende che lavorano per progetti, commesse e clienti che offre tutta la semplicità e l'innovazione necessarie per il controllo dei tempi di lavoro e il miglioramento delle performance aziendali.

La semplicità di rilevare le ore di lavoro del personale, l'innovazione di una soluzione che permette di controllare puntualmente la redditività delle commesse, dei progetti, delle attività svolte quotidianamente all'interno dell'azienda sono i suoi punti di forza.

ZTimesheet è la soluzione ideale per le aziende che vogliono uno strumento per:

1. valutare la redditività dei progetti/commesse/clienti;
2. pianificare e monitorare le attività delle risorse impegnate;
3. ridurre i tempi delle gestioni amministrative;

4. fornire al management le informazioni necessarie per prendere le migliori decisioni nel minor tempo possibile.

ln input il software avrà: l'acquisizione anagrafici da ERP (commesse, attività, centro di costo, preventivi ore, allocazione risorse, working breakdown entity), l'acquisizione ore dal programma Zucchetti «Presenze» (timbrature, giustificativi di assenza, straordinari), l'acquisizione del costo orario dal programma «Paghe» (valorizzato mensilmente in base alle variabili utilizzate in ciascuna elaborazione).

ZTimesheet è stato studiato ed è realizzato secondo un approccio modulare per consentire a ciascuna azienda di configurare la soluzione sulle proprie specifiche esigenze e disporre così di un supporto perfettamente rispondente alle proprie necessità di verifica dell'andamento delle attività, del carico di lavoro dei collaboratori, della redditività dei progetti/commesse/clienti.

La soluzione è completamente realizzata in tecnologia web ed è accessibile tramite un portale che mette a disposizione all'azienda una serie di servizi utili per agevolare e semplificare le attività quotidiane e massimizzare l'efficienza dell'intero processo di gestione.

Ecco come è formato il prodotto:

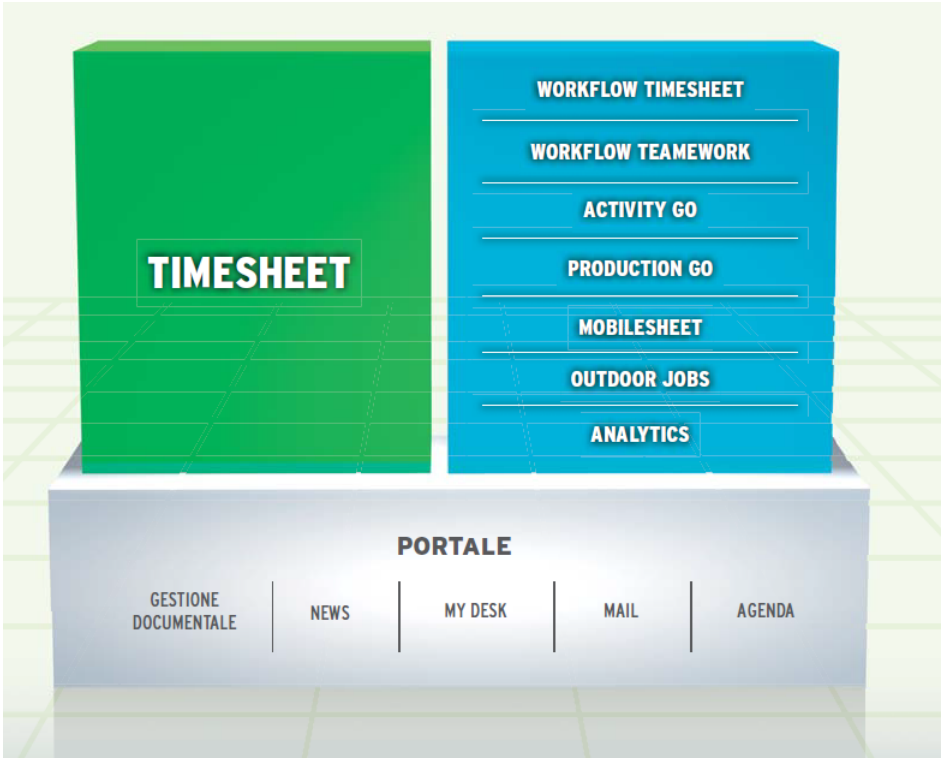

Figura 4.1: ZTimesheet

Timesheet è il modulo principale della suite e consente di verificare tempestivamente il tempo/costo che ciascuna risorsa dell'azienda dedica alle varie attività lavorative.

Con Timesheet è possibile: effettuare una quadratura sistematica tra le ore ripartite per commessa/cliente/progetto e le ore che saranno utilizzate per la retribuzione del personale; inoltre è possibile valorizzare puntualmente le ore con il costo reale del personale e alimentare automaticamente i sistemi aziendali di controllo di gestione, contabilità e fatturazione.

I moduli aggiuntivi sono:

1. WORKFLOW TIMESHEET: è il modulo che permette alle risorse associate a ciascun cliente, progetto, commessa o attività di provvedere autonomamente all'inserimento delle ore lavorate secondo un processo semplificato e immediato. Il Project Manager con estrema semplicità dispone, così, di tutti i timesheet compilati, verifica la correttezza e autorizza o respinge le ore inserite. Uno strumento utile e strategico per tutte le aziende, in particolare per quelle di servizi la cui redditività è strettamente connessa alla produttività delle risorse.

2. WORKFLOW TEAMWORK: permette di monitorare ogni singola prestazione lavorativa confrontando il budget previsionale con le ore effettivamente lavorate dai dipendenti. Si tratta, quindi, di un modulo strategico che permetterà di verificare con immediatezza l'andamento delle attività, il carico di lavoro delle risorse impiegate e, nel caso di scostamenti rispetto a quanto preventivato, di intervenire con tempestive azioni correttive. L'inserimento delle ore per i dipendenti e per le commesse/servizi è in capo ai responsabili di commessa/servizi e alle segreterie di area secondo una logica di workflow che prevede un iter di valutazione e di approvazione dei dati inseriti. Il modulo permetterà anche di gestire il personale non direttamente assegnato attraverso la selezione dei "prestiti".

3. MOBILESHEET: una soluzione per la compilazione dei timesheet attraverso strumenti mobile (smartphone e tablet). Anche in assenza di connessione dati, il collaboratore e/o il responsabile può classificare giornalmente le ore per clienti, commesse, attività e comunicarle con semplicità al responsabile o al project manager per la relativa approvazione. Sarà possibile allegare la fotografia dei documenti (rapporti, copie cartacee) per velocizzare le fasi di controllo e di chiusura senza dover attendere la consegna della copia cartacea.

4. PRODUCTION GO: modulo per le aziende con personale che hanno una sede fissa di lavoro (stabilimenti, capannoni, aree di produzione, reparti, uffici ecc.) e che devono rilevare le tipologie di prestazioni svolte quotidianamente. E' installato sul terminale che permette di gestire in maniera controllata, interattiva e guidata la consuntivazione delle ore di produzione, la sincronizzazione remota delle anagrafiche (commesse, attività, centro di costo, reparto) e dei filtri di attribuzione direttamente sul terminale.

5. ACTIVITY GO: modulo specificatamente pensato per le aziende che offrono servizi di manutenzione (estintori, filtri dell'aria, illuminazione, porte, caldaie, ascensori), di pulizia (servizi igienici, servizi in aree e strutture pubbliche e private), di vigilanza (piantonamento, servizio ispettivo, trasporto valori) e del settore edilizia (costruzione edifici, manutenzione impianti idraulici ed elettrici). Il modulo infatti supporta l'azienda dapprima nelle fasi di pianificazione degli interventi e successivamente di controllo delle attività.

6. OUTDOOR JOBS: è il modulo pensato per la gestione del personale che lavora su commesse, progetti ed appalti fuori dai locali (ad esempio presso i clienti) che permetterà all'azienda di disporre di un valido strumento per la pianificazione degli interventi e di un'agenda organizzata per la corretta allocazione dei tecnici. La soluzione permette di gestire tutte le fasi di un intervento: pre-job, on the job, post-job.

7. ANALYTICS: la soluzione di Business Intelligence che permette agli uffici coinvolti nell'intero processo produttivo e al management di effettuare analisi e valutazioni sui costi per prendere decisioni sulla base di informazioni contestualizzate e strutturate.

Si vuole ora analizzare il prodotto presentato, contestualizzandolo ed applicandolo a tutti gli strumenti che il framework con metodologia fuzzy propone.

Seguendo tutte le fasi presentate precedentemente, verrà analizzato il flusso informativo per arrivare a formulare una soluzione di qualità del prodotto in caso di studio.

4.2 Identificazione degli obiettivi

Ciò che si è descritto fino ad ora nella «panoramica di prodotto» è sostanzialmente ciò che viene presentato nel documento di product RoadMap che fa da «pre-fase» alla fase di identificazione degli obiettivi.

Identificare gli obiettivi che si vogliono raggiungere non è semplice in quanto significa analizzare i goal da raggiungere partendo da quelli più di alto livello per poi arrivare a definire obiettivi molto più raffinati.

In questa fase, come detto precedentemente, gli obiettivi possono derivare sia da requisiti di carattere funzionale sia di carattere non funzionale ed essere anche molto di «alto livello» e poco dettagliati.

Si partirà dagli obiettivi di «alto livello», inquadrando il software e definendo quali sono i «customer» che usufruiranno di questo servizio e mostrando il punto esatto in cui questo software opererà all'interno del sistema.

ZTimesheet è una soluzione software che, per le sue caratteristiche, si riferisce soprattutto a tutte quelle aziende che svolgono azioni di Polizia o di forze dell'or-

dine, sanità pubblica, porti o areoporti, call center, aziende di produzione, grande distribuzione, aziende di servizi e aziende di videosorveglianza.

Il prodotto possiede i seguenti requisiti funzionali:

1. riduzione dei tempi di gestione;
2. controllo delle ore lavorate e della produttività delle risorse;
3. ripartizione delle ore per la contabilità analitica e le analisi a consuntivo;
4. condivisione di informazioni quali timbrature, giustificativi, anomalie, totalizzatori;
5. quadratura dei cartellini e trasmissione dei dati di presenza/assenza alla procedura «Paghe» per l'elaborazione dei cedolini;
6. comunicazione di voci giornaliere e mensili, ad esempio indennità, maggiorazioni, importi (ad esempio multe e rimborsi);
7. valorizzazione economica dei dati movimentati e trasmissione alle procedure contabili e/o di fatturazione;
8. pianificazione delle manutenzioni in modalità ciclica (con frequenza oraria, giornaliera, mensile, annuale) e programmata (con indicazione specifica della data prevista);
9. verifica dell'effettivo svolgimento delle attività manutentive;
10. predisposizione del piano di manutenzione per singola area di competenza;
11. redazione delle statistiche per la valutazione quantitativa e qualitativa delle prestazioni di ogni singolo addetto: durata di ogni manutenzione, copertura dei turni di lavoro, copertura effettiva del servizio nel rispetto dei tempi e delle tolleranze applicate, percentuale di inattività di ogni addetto;

E i seguenti requisiti non funzionali:

1. sicurezza dei dati,
2. comunicazione efficace ed efficiente,
3. dati affidabili,
4. tempi di risposta adeguati,
5. tempi di sviluppo ragionevoli in base alle esigenze,
6. gestione di informazioni multiple ed eterogenee,

7. numero medio di funzioni disponibili elevato,
8. tecnologie di archiviazione e di gestione documenti scalabili,
9. strumentazione e tecnologie opportune per lo scambio di dati,
10. integrazione completa con altri applicativi,

ZTimesheet si pone a centro di diversi applicativi Zucchetti tra cui:

1. «Paghe»: evoluto software in tecnologia web che supporta le moderne direzioni del personale nello svolgimento delle attività quotidiane. I numerosi automatismi di calcolo, l'ampiezza funzionale, i servizi di aggiornamento automatico dei CCNL (commercio, agricoltura, artigianato, dirigenti, spettacolo, edile) e delle tabelle contributive sia ordinarie che fondi negoziali, la gestione telematica integrata degli adempimenti (cassa edile, SCAU, F24, UNIEMENS) sono solo alcuni dei vantaggi del software «Paghe» e «Contributi» Zucchetti.

2. «Presenze»: soluzione web per la rilevazione delle presenze e delle assenze, che permette una gestione ottimale anche in aziende complesse con più sedi, filiali, negozi, cantieri distribuiti sul territorio. Specifici processi di workflow permettono, infatti, di automatizzare e gestire i processi tra il dipendente e l'ufficio del personale relativamente a inserimento e richieste giustificativi (permessi, ritardi, ferie, mancate timbrature, approvazione degli straordinari).

3. «WorkForce Management»: permette di pianificare e gestire i turni aziendali in base ai fabbisogni aziendali e all'esigenza di copertura di divisioni/reparti definendo orari di lavoro, standard di servizio e gli skill necessari. Consente una razionale ed omogenea organizzazione dei turni di lavoro, attribuendo in maniera funzionale i carichi di lavoro e riducendo notevolmente il ricorso a straordinari e reperibilità.

Tutto ciò rende ancora più difficile ma allettante trovare un prodotto di qualità che riesca ad integrarsi pienamente con altri prodotti che attraverso la sua modularità permetta di rendere ancora più manutenibile e di facile gestione l'introduzione di eventuali nuove soluzioni di ZTimesheet.

4.3 Diagrammi i*

Identificati quali sono gli obiettivi si procederà ora con la costruzione dei modelli i*.

Innanzitutto si costruiranno i diagrammi SD e SR del prodotto da sviluppare che è presentato come caso di studio e in seguito si mostrerà il diagramma SD di ZTimesheet, seguendo le regole di i* modeling.

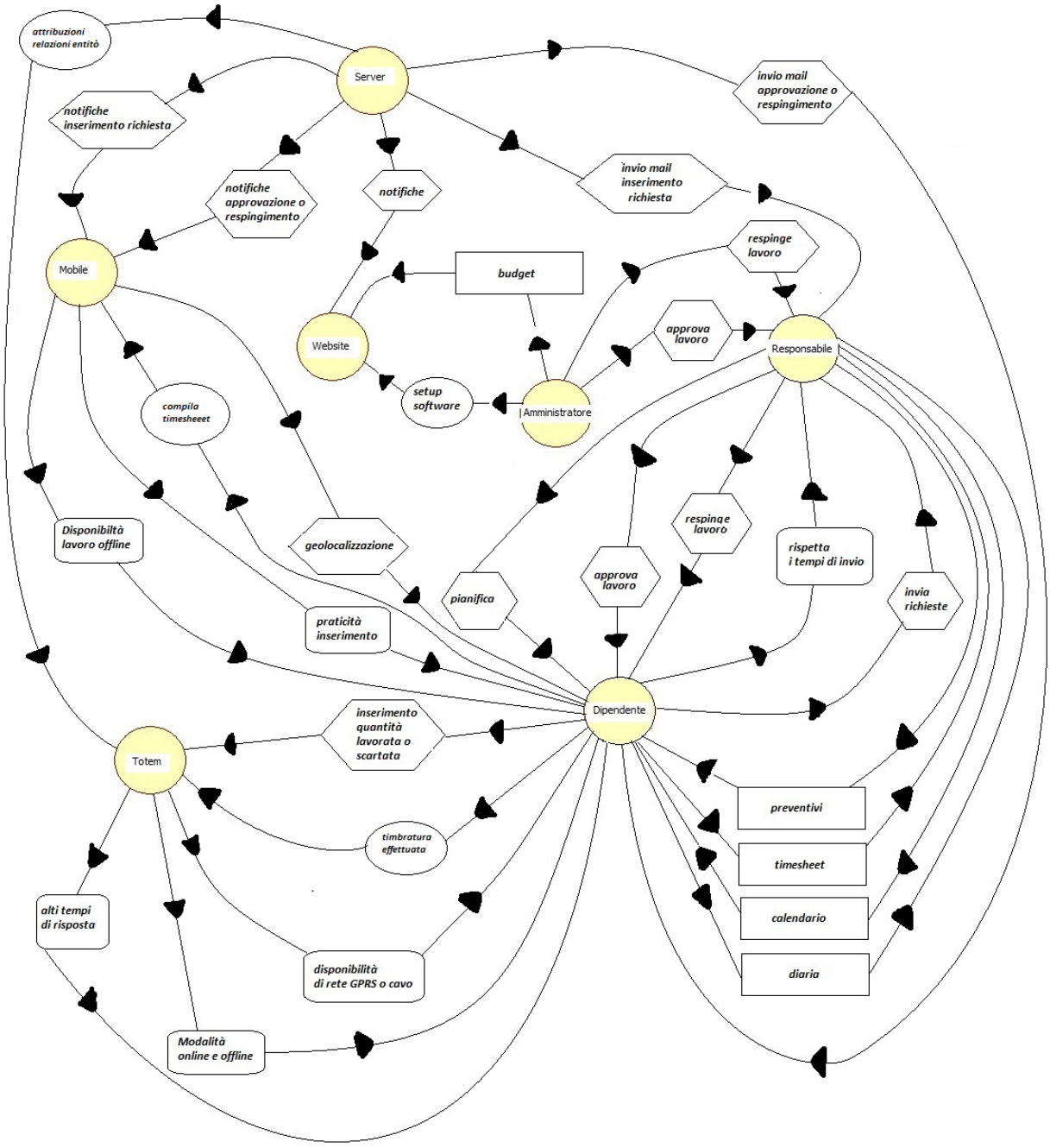

Figura 4.2: SD model

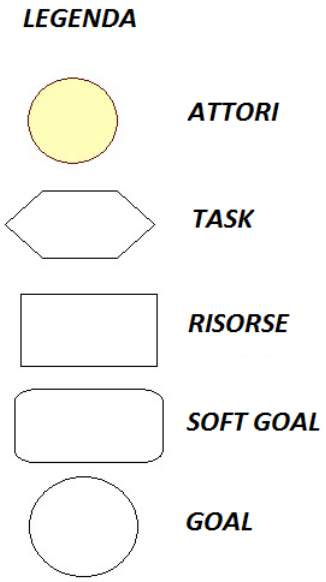

Figura 4.3: SD model: legenda

Una volta costruito l'SD model, si procederà con la costruzione dell'SR model aggiungendo all'SD tre nuove semantiche: «Task decomposition link» (che decompone il task in sub-task), «Means-end link» (la relazione tra un goal o un resouces ed un means, ovvero un elemento che li descriva), «Contribute-To Soft Goal link» (a means-end link con soft goal). Si dividerà l'SR in parti, analizzando ogni attore presentato nell'SD con le sue funzionalità.

Si partirà con l'analisi della figura del «dipendente». Dal diagramma SD, si nota che il «dipendente»:

1. inserisce le ore lavorate dalla propria postazione da applicazioni per dispositivi mobile e totem,

2. dispone on line delle informazioni necessarie allo svolgimento della propria attività lavorativa, per la compilazione di timesheet e la consultazione di ciò che il suo responsabile ha pianificato per lui,

3. consulta il calendario che mostra la sua pianificazione di lavoro e nel contempo inserisce dati (timesheet, diaria) da inviare sotto forma di richiesta al proprio responsabile,

4. effettua timbrature presso terminali,

5. alla fine del processo riceverà mail o notifiche inerenti il suo lavoro svolto, avendo un feedback della sua attività.

Ciò che il software si pone come obiettivo per la figura del dipendente è quello di rispettare i tempi di invio di richieste al proprio responsabile (obiettivo presentato anche nel diagramma SD) seguendo tutte le attività mostrate nella Fig.4.2.

Di seguito viene presentato il modello SR del dipendente con i suoi obiettivi, le sue attività e le risorse a disposizione utili al raggiungimento del goal «rispetto dei tempi di invio».

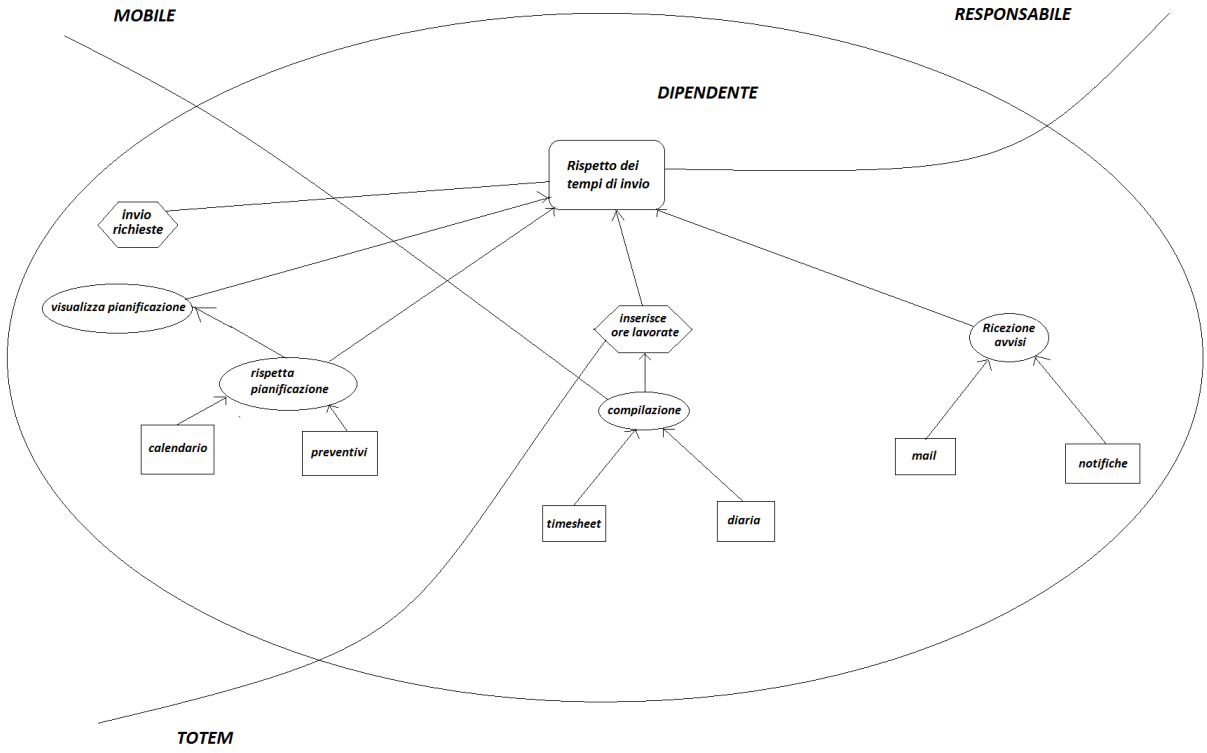

Figura 4.4: SR model «dipendente»

Si nota che il «dipendente» in questo diagramma dialoga con tre altri «attori» (responsabile, mobile e Totem) attraverso le attività che è in grado di svolgere con lo scopo di raggiungere obiettivi. In questo caso di studio si prenderà come modello esemplificativo il «Totem» che permetterà al dipendente di raggiungere l'obiettivo di «effettuare timbrature» di produzione (come mostra il diagramma SR seguente).

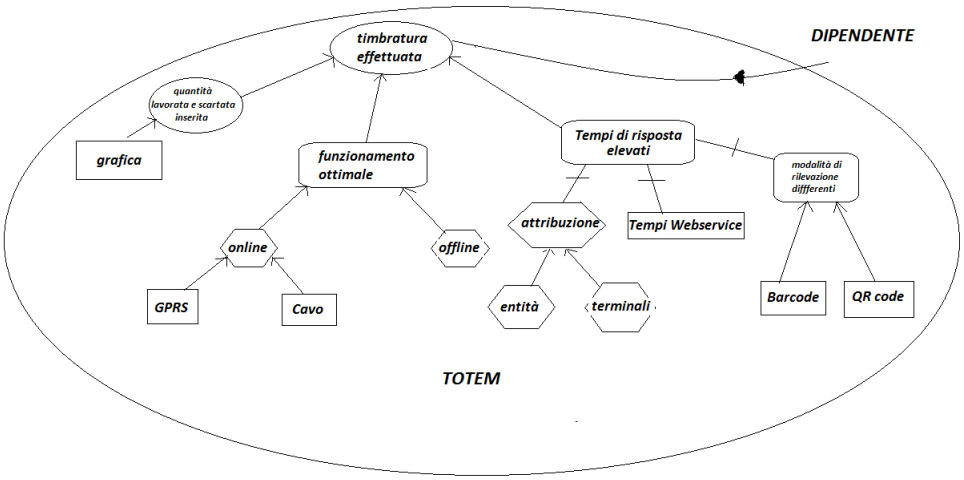

Figura 4.5: SR model «Totem»

Si nota qui la stretta collaborazione tra gli attori in gioco che saranno gli elementi centrali del software e «attori protagonisti» dello sviluppo.

Nei diagrammi SR qui mostrati si può vedere che esistono una serie di relazioni AND (identificate da una linea al centro delle frecce) e da relazioni OR (indentificate da solo una semplice freccia) che nella fase iniziale di questo framework, hanno la prerogativa di permettere di racchiudere tutte le funzionalità e gli obiettivi da raggiugere.

Per ogni attore mostrato nel diagramma SD, si dovrà procedere con la costruzione di diagrammi SR prendendo in considerazione gli obiettivi da raggiungere e proseguendo poi con tutti i passi del flusso informativo (Fig.3.1), per arrivare ad una valutazione più efficace di progetto.

4.4 Feature Model

Dopo aver costruito i diagrammi i* il prossimo passo del flusso informativo della metodologia è rappresentato dalla «ricerca delle feature model» per dare una definizione più precisa del prodotto che si intende sviluppare.

Come notato, i diagrammi i* (SD e SR) catturano un largo numero di alternative AND/OR con un insieme di operazioni, task e configurazioni per cercare di raggiungere i goal; le feature model invece sono un modello più compatto per una rappresentazione di una vasta gamma di prodotti.

Analizzati i diagrammi i*, ora si ha una panoramica più completa dei prodotti che gli «attori del software» possono utilizzare; questo permetterà di costruire attorno ad essi le feature model necessarie.

Da ora in poi si farà riferimento ad un singolo attore per trattare al meglio tutte le fasi della metodologia; va da sè che per una trattazione completa di un prodotto,

si renderà necessario analizzare ogni attore presente nel diagramma SD e seguendo lo stesso flusso informativo presentato nel capitolo precedente.

In questa trattazione, si è deciso di prendere come attore di caso di studio il «Totem».

Nella realtà il totem Zucchetti è un terminale interattivo touch screen che permetterà, tra le altre cose, la rilevazione dei dati della produzione.

Dal diagramma SR si notano notevoli caratteristiche e obiettivi da raggiungere, che permettono di dedicare così a questo attore un prodotto standard per il raggiungimento di alcuni obiettivi di progetto stilati nella fase di identificazione.

L'idea di partenza per la costruzione della feature model è quella di definire la tipologia di prodotto che si vuole descrivere analizzando gli obiettivi dalla fase di identificazione e di modeling (SD e SR). In questo caso si otterrà la seguente feature model:

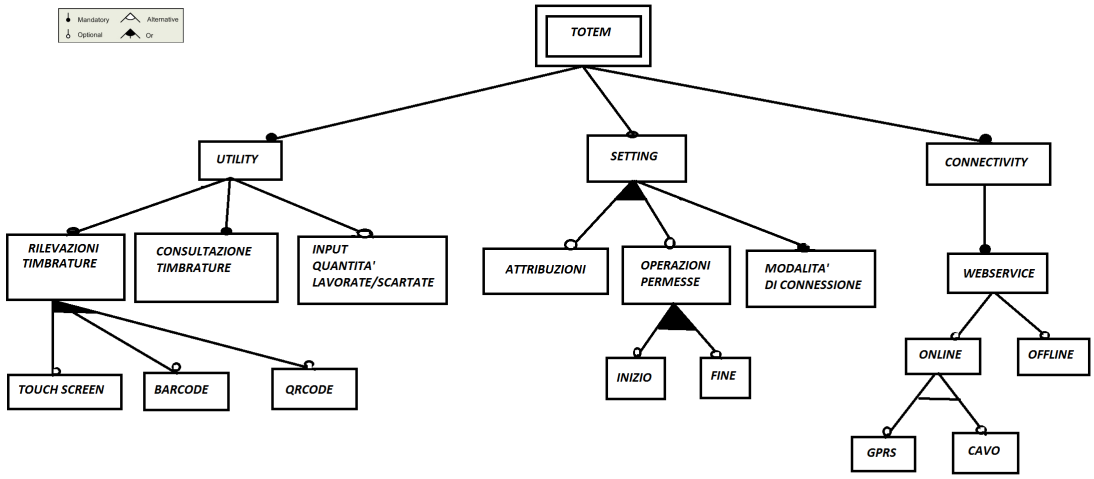

Figura 4.6: Feature model totem

Questa feature model descrive in linguaggio di modellizzazione ciò che nel linguaggio naturale si può presentare cosi:

il prodotto che si svilupperà su Totem è caratterizzato da utility, settaggi (setting) e connettività (connectivity). Tra le Utility abbiamo la possibilità di rilevare/consultare timbrature ed inoltre la possibilità di inserire le quantità lavorate o scartate a livello di produzione. Le modalità di rilevazione delle timbrature possono essere tramite touch screen (il dipendente inputerà le entità in cui ha lavorato cliccando sullo schermo), Barcode o QRcode.

Il setting del totem è caratterizzato dall'inserimento di eventuali attribuzioni (ovvero ciò che a parole si descriverebbe in questo modo: «sul terminale A possono

timbrare solo i dipendenti B,C e D e nessun altro»), dalle operazioni permesse (inizio o fine o entrambe) e dalla modalità di connessione che è legata alla tipologia di webservice da utilizzare (online o offline).

La connettività invece identifica come il totem dialoga con il server, ovvero dalla tipologia di webservice da utilizzare, che sia online (via GPRS o via cavo) o offline.

Viste le caratteristiche presentate, si può già notare che questa feature model rappresenterà il modulo PRODUCTION GO, già presentato nella panoramica del prodotto [par.4.1].

4.5 Fuzzificazione

Definito il diagramma i* e definita la feature model ora si hanno più chiari quali sono gli obiettivi da raggiungere e quali sono le risorse a disposizione, per questo motivo si procederà ora con la fuzzificazione e lo studio delle metriche utili a questo prodotto per la valutazione.

I fuzzy set che si andranno a realizzare si baseranno su diversi aspetti, tra cui:

1. le esperienze passate,
2. le valutazioni di progetti simili a ciò che si è creato,
3. la capacità aziendale in termini di risorse hardware/software e risorse umane,
4. dall'attitudine e dall'investimento effettuato per il progetto.

Ad esempio, Zucchetti è uno dei più importanti protagonisti italiani del settore dell'IT con oltre 2.800 addetti, una rete distributiva che supera i 1.100 partner sull'intero territorio nazionale e oltre 100.000 clienti. La completezza multidisciplinare permette da sempre al gruppo Zucchetti di sviluppare prodotti e servizi di altissima qualità che garantiscono all'utente di disporre della miglior soluzione del mercato.

Soluzioni software, hardware e servizi innovativi realizzati e studiati per soddisfare le specifiche esigenze di:

1. aziende di qualsiasi settore e dimensione, banche e assicurazioni;
2. professionisti (commercialisti, consulenti del lavoro, avvocati, curatori fallimentari, notai),associazioni di categoria e CAF;
3. pubblica amministrazione locale e centrale (Comuni, Province, Regioni, Ministeri, società pubbliche).

Inoltre Zucchetti possiede un datacenter di proprietà garantendo la continuità dei servizi di outsourcing in tutte le condizioni.

La realtà quindi è quella di una grande azienda, di conseguenza le fuzzy rule sviluppate saranno molto «stringenti» in termini di qualità, in quanto rappresenta un punto di forza per l'azienda.

Per la definizione dei fuzzy set si prendono in considerazione tutte le metriche presentate nel Capitolo 3 e su cui si costruiranno poi le fuzzy rule.

4.5.1 Affidabilità

E' la prima volta che Zucchetti propone il totem come strumento per la rilevazione delle timbrature di produzione.

In passato si erano proposti utilizzi di altri terminali «poco intelligenti» per l'acquisizione delle timbrature di produzione (non venivano effettuati alcun tipo di controlli di attribuzione, erano solo mezzi che veicolavano al server l'informazione).

Altri meccanismi utilizzati erano gli MDC, ovvero terminali tascabili palmari che utilizzavano la tecnologia RFID in cui bastava avvicinarlo al TAG che il rilevatore leggeva tutte le informazioni necessarie rilevando dati quali: la presenza, la data e il luogo di lavoro.

L'esperienza di terminali ha insegnato che per un'azienda che utilizzava in media per 20 ore al giorno un rilevatore poco intelligente, vi era un guasto ogni qualche mese (all'incirca 6 mesi) che rappresentava l'MTTF del sistema. La riparazione del guasto avveniva in circa qualche settimana (MTTR), di conseguenza l'MTBF era di circa 7 mesi.

Si era notato che, come scontato, più l'uso era costante e continuativo e più era elevata la probabilità di guasto di dispositivi. I guasti erano dovuti soprattutto a difetti software, oppure causati da eventi accidentali (come per esempio una caduta improvvisa).

In questo caso non si può fare molto affidamento a questi dati, poichè il totem è un dispositivo molto più sicuro e robusto rispetto a «vecchi terminali» in quanto più «intelligente» e dotato di un sistema operativo a bordo.

Si dovrà quindi tenere in considerazione la quantità di lavoro presente per un terminale e stimare mediamente il suo carico di lavoro.

In generale possiamo dire che il carico di lavoro del totem sarà più elevato rispetto ai vecchi terminali in quanto i dati di elaborazione nel totem saranno triplicati, ma c'è da considerare anche il fatto che il totem ha una maggior capacità e affidabilità avendo quindi sostanzialmente un MTTF maggiore e un MTTR presumibilmente minore rispetto ai vecchi terminali.

Si potrebbe stimare un miglioramento sia di MTTF sia di MTBF pari al 50% (quindi MTTF circa 9 mesi, MTBF circa 10,5) di conseguenza si avrà che l'affidabilità A sarà data da:

$$A = \frac{MTTF}{MTTF + MTTR} \cong \frac{MTTF}{MTBF} \cong \frac{9}{10.5} \cong 0.86$$

Tutto ciò viene descritto dall'esperienza, la stima può essere quindi molto variabile.

I fuzzy set in questo caso non sono omogeneamente distribuiti: il fuzzy set LOW è «più piccolo» rispetto a tutti gli altri fuzzy, mentre il fuzzy level MEDIUM sarà «più grande».

Questo significa che per avere una affidabilità eccellente, il prodotto deve avere delle ottime caratteristiche che lo fanno ricadere in un livello fuzzy molto stringente:

Tabella 4.1: Affidabilità: livelli fuzzy

A (%)	**Fuzzy level**
$(0 - 20)$	**LOW**
$(> 20 \,\&\, \leq 90)$	**MEDIUM**
$(> 90 \,\&\, \leq 100)$	**HIGH**

4.5.2 SLOC

Il numero di linee di codice da sviluppare farà riferimento soprattutto alla costruzione di Webservice e dei meccanismi di setting del terminale lato Web da parte dell'amministratore. Il compito sarà quindi quello di implementare delle maschere di interfaccia che andranno a configurare dei controlli da fare al momento della timbratura del dipendente. Per lo studio del numero di linee di codice da sviluppare bisognerà fare riferimento all' esperienza precedente.

Il linguaggio di programmazione utilizzato per la costruzione dei webservice è Java e l'interscabio di dati tra server e totem seguiranno un formato di tipo JSON.

A livello client la maschera è stata già costruita con del linguaggio Javascript e Html; per quanto riguarda il documento di CSS sia per il totem sia per il Web, non ci sarà bisogno di alcuno sviluppo in quanto è già presente un CSS standard utile a tutti gli applicativi appartenenti alla suite HR (In caso contrario bisognava inserire nella stima anche questo conto).

Il numero di linee di codice verrà stimato e farà parte della fase valutazione, anche se il suo peso sarà minimo in quanto SLOC sarà solo maggiormente non in modo indiretto per lo studio dell'effort.

Come detto già nel Capitolo 3, con la nascita delle tecniche «object oriented» (OO) la metrica SLOC non ha più un peso significativo nello studio della qualità di prodotto, di conseguenza anche un valore HIGH all'interno di una fuzzy rule non influenzerà significativamente la qualità del prodotto.

Per il prodotto sviluppato sul Totem Zucchetti si stima una quantità di circa 3 KLOC linee di codice che saranno all'interno del livello MEDIUM del fuzzy set.

Di conseguenza si stimeranno i seguenti fuzzy set:

Tabella 4.2: SLOC: livelli fuzzy

SLOC (KLOC)	**Fuzzy level**
$(0-2)$	**LOW**
$(>2 \,\&\, \leq 20)$	**MEDIUM**
>20	**HIGH**

4.5.3 Effort

Zucchetti è una realtà molto grande, una software house che sviluppa prodotti che devono rispettare sempre una serie di vincoli stretti, con un team di sviluppatori ampio che hanno modo di verificare ogni fase (dall'analisi ai test), per questo motivo per la parametrizzazione del COCOMO si farà riferimento alla modalità «Embedded» ottenendo così le seguenti formule:

$$size = (Val)\, KLOC$$

$$MM = 3.6 * (Val)^{1.20}$$

$$TDEV = 2.5 * MM^{0.32}$$

Dove Val si baserà approssimativamente sui valori precedenti di SLOC, di conseguenza i fuzzy set dell'effort saranno derivati dai fuzzy set del SLOC e dall'esperienza dell'azienda nel settore.

In questo caso i fuzzy set saranno:

Tabella 4.3: Effort: livelli fuzzy

Effort (MM)	**Fuzzy level**
$(0-15)$	**LOW**
$(>15 \,\&\, \leq 20)$	**MEDIUM**
>20	**HIGH**

4.5.4 Produttività

Per calcolare la produttività si farà riferimento ai FPs del prodotto, in quanto il numero di linee di codice non ci da un'indicazione significativa [par. 3.5.3.1].

Il linguaggio utilizzato maggiormente per lo sviluppo è stato Java di conseguenza, utilizzando il valore medio presentato nella Fig. 3.2, si ottiene che in media:

$$1\, SLOC = 53 FPs$$

e siccome per il prodotto sviluppato sul totem Zucchetti si stima una quantità media di circa 3 KLOC linee di codice, si avrà che

$$3000\,SLOC = 159000 FPs$$

di conseguenza sarà immediato calcolare la produttività in MM attraverso la formula presentata nel [par. 3.5.5.1] utilizzando i FPs.

Questa indica la produttività generale di progetto, si potrebbe anche dividere per il numero di sviluppatori a disposizione per avere il valore di produttività per singolo sviluppatore.

Per costruire i livelli fuzzy si farà riferimento ai valori limite dei livelli fuzzy di SLOC su cui si baseranno approssimativamente, ovvero si farà questa scelta:

Tabella 4.4: Produttività: livelli fuzzy

Produttività (FPs/mesi uomo)	**Fuzzy level**
$(0 - 15000)$	**LOW**
$(> 15000\,\& \leq 20000)$	**MEDIUM**
> 20000	**HIGH**

4.5.5 Robustezza

La robustezza rappresenta la capacità del prodotto di continuare a funzionare senza degrado nelle sue prestazioni anche in casi limite.

Come mostrato nel [par. 3.5.4.2] si avranno:

1. il tempo medio di risposta di una transazione (invio timbrature al server, tempi di risposta di webservice);
2. percentuale di utilizzo delle linee di trasmissione e delle risorse;
3. probabilità di accessi non autorizzati (sicurezza);
4. probabilità di errore.
5. tempo di ripristino errore.

Si riassume tutto in questa tabellina:

Tabella 4.5: Robustezza

Metrica	**Descrizione**	**Unità di misura**
R1	tempo medio di risposta di una transazione e delle risorse	*sec*
R2	percentuale di utilizzo delle linee di trasmissione	%
R3	probabilità di accessi non autorizzati	%
R4	probabilità di errore	%
R5	tempo di ripristino dell' errore	*min*

Di seguito vengono mostrate le tabelle dei livelli fuzzy per la metriche di «robustezza» che saranno utili per le metriche provenienti da GQM [par. 4.5.9]:

1. R1: tempo medio di risposta di una transazione.

Tabella 4.6: R1: livelli fuzzy

R1 (sec)	**Fuzzy level**
$(0-15)$	**LOW**
$(>15 \,\&\, \leq 20)$	**MEDIUM**
>20	**HIGH**

2. R2: percentuale di utilizzo delle linee di trasmissione, delle risorse.

Tabella 4.7: R2: livelli fuzzy

R2 (%)	**Fuzzy level**
$(0-9)$	**LOW**
$(>9 \,\&\, \leq 50)$	**MEDIUM**
$(>50 \,\&\, \leq 100)$	**HIGH**

3. R3: probabilità di accessi non autorizzati (sicurezza).

Tabella 4.8: R3: livelli fuzzy

R3 (%)	**Fuzzy level**
$(0-9)$	**LOW**
$(>9 \,\&\, \leq 50)$	**MEDIUM**
$(>50 \,\&\, \leq 100)$	**HIGH**

4. R4: probabilità di errore.

Tabella 4.9: R4: livelli fuzzy

R4 (%)	**Fuzzy level**
$(0-20)$	**LOW**
$(>20 \,\&\, \leq 50)$	**MEDIUM**
$(>50 \,\&\, \leq 100)$	**HIGH**

5. R5: tempo di ripristino errore.

Tabella 4.10: R5: livelli fuzzy

R5 (min)	**Fuzzy level**
$(0-5)$	**LOW**
$(>5 \,\&\, \leq 60)$	**MEDIUM**
>60	**HIGH**

4.5.6 Costo di sviluppo

Zucchetti per lo sviluppo di questo software, metterà a disposizione N sviluppatori, di conseguenza sarà messo a budget il loro costo di sviluppo per la prestazione offerta all'azienda. Calcolati i mesi uomo con lo studio dell'effort (X) e sapendo che il sofware deve essere sviluppato in un massimo T_{max} mesi e in un minimo di T_{min} mesi, seguendo le regole presentate precedentemente, si possono calcolare ora quali siano i numeri minimi e massimi di sviluppatori necessari:

$$N_{min} = \frac{X}{T_{max}}$$

$$N_{max} = \frac{X}{T_{min}}$$

Questo è ciò che servirà a livello di risorse umane, ovvero ciò che serve sapere per arrivare a raggiungere l'obiettivo, avendo quindi dei livelli fuzzy del lavoro da fare.

I valori di Zucchetti saranno:

1. $N = 7$;
2. $T_{max} = 2$ mesi , $T_{min} = 1$ mese;
3. X valore effort calcolato precedentemente (mesi/uomo).

Di seguito viene mostrata la tabella dei livelli fuzzy per la metrica «costo» intesa come numero di sviluppatori:

Tabella 4.11: Numero sviluppatori: livelli fuzzy

N	Fuzzy
0–6	**LOW**
$(> 6 \,\&\, \leq 10)$	**MEDIUM**
> 10	**HIGH**

4.5.7 Scostamento di tempo

Come detto precedentemente, Zucchetti per «Production Go» prevede uno sviluppo di circa 2 mesi (60 giorni) per la costruzione anche a livello prototipale del prodotto (valore che indicherà il «tempo pianificato»).

Alla fine del processo di sviluppo, si terrà traccia di quanto tempo è stato impiegato realmente, calcolando poi così lo scostamento di tempo [par. 3.5.6.3].

Sarà facile poi definire i livelli fuzzy per questa metrica, considerando il fatto che più il fattore è alto e più si è stati in ritardo rispetto le consegne; di conseguenza se il valore sarà maggiore di 1 sicuramente avremo un livello LOW, poi sarà l'azienda (in questo caso Zucchetti) a definire gli altri livelli di interesse (MEDIUM e HIGH).

In questo caso di studio si definiranno i seguenti fuzzy set:

Tabella 4.12: Scostamento di tempo: livelli fuzzy

S_T	**Fuzzy**
> 1	**LOW**
$(> 0.5 \,\&\, \leq 1)$	**MEDIUM**
0−0.5	**HIGH**

4.5.8 Scostamento di effort

Alla fine del processo di sviluppo si terrà traccia anche di quanto «sforzo» si è compiuto in termini di mesi uomo, calcolando così lo scostamento di effort [par. 3.5.6.2].

Nel caso di Zucchetti, per «Production Go» è stato pianificato un effort pari a 15 mesi-uomo quindi, considerando il fatto che più il valore sarà maggiore di 1 e più lo «sforzo» degli sviluppatori sarà grande, sarà quindi semplice stabilire dei livelli fuzzy che ne permettano di capire lo scostamento: se il valore sarà maggiore di 1 sicuramente avremo un livello LOW, poi sarà l'azienda (in questo caso Zucchetti) a definire gli altri livelli di interesse (MEDIUM e HIGH).

In questo caso di studio si definiranno i seguenti fuzzy set:

Tabella 4.13: Scostamento di effort: livelli fuzzy

S_E	**Fuzzy**
> 1	**LOW**
$(> 0.5 \,\&\, \leq 1)$	**MEDIUM**
0−0.5	**HIGH**

4.5.9 Goal Question Metric

Analizzando i diagrammi presentati precedentemente (SD,SR e feature model) ed analizzato il product roadmap nella fase di identificazione degli obiettivi, in questa fase ci si pone di capire per ogni prodotto (per ogni feature model) quali siano gli obiettivi, cercando di costruire delle metriche oggettive per la valutazione ed il monitoring.

Proseguendo con la trattazione precedente, si prenderà come punto di riferimento il prodotto «Production Go» (che si è identificato come il prodotto che fa riferimento alla feature model del Totem).

«Production Go» (chiamato anche modulo per «ZTotem») è il modulo per le aziende con personale che hanno una sede fissa di lavoro (stabilimenti, capannoni, aree di produzione, reparti, uffici) e che devono rilevare le tipologie di prestazioni svolte quotidianamente. Installato sul terminale Zucchetti, il software permetterà di gestire in maniera controllata, interattiva e guidata la consuntivazione delle ore di

produzione, la sincronizzazione remota delle anagrafiche (commesse, attività, centro di costo, filiale, clienti) e dei filtri di attribuzione direttamente sul terminale.

Si possono quindi identificare questi goal:

G_1: disponibilità dati,

G_2: comprensibilità del software,

G_3: disponibilità dell'applicativo,

G_4: coerenza dei dati,

G_5: sicurezza,

A cui fanno riferimento queste domande (question):

Q_1: Quanta capacità di possedere dati real time si ha a livello client?

Q_2: Quanta formazione sarà necessaria ai dipendenti utilizzatori del software?

Q_3: L'applicativo è disponibile ad operare offline in caso di mancanza di comunicazione online con il server?

Q_4: Quanto i dati sono coerenti con le configurazioni aziendali fatte lato web?

Q_5: Il software è sicuro rispetto all'accesso incontrollato dei dati?

e a cui si potranno definire queste metriche:

M_{11}: probabilità di «server down» (R4),

M_{12}: tempo medio di risposta (R1),

M_{21}: tempo necessario alla formazione dei dipendenti,

M_{22}: numero di dispositivi installati rispetto alla necessità effettiva dell'azienda,

M_{31}: probabilità errore di rete (R4),

M_{32}: numero medio di transazioni da memorizzare e di risorse da utilizzare (R2),

M_{33}: tempo medio di ripristino del sistema dopo un' interruzione (R5),

M_{41}: probabilità di allineamento tra i dati dell' anagrafica entità definita lato server e dell'anagrafica entità presentata (Affidabilità A),

M_{51}: probabilità di accesso non autorizzato (R3),

dove tra parentesi vengono indicate le metriche già consolidate (se esistono) per determinare la valutazione sia ex-ante sia ex-post.

Riassumendo:

Tabella 4.14: Goal e metriche

GOAL	METRICHE
G_1	$M_{11} = R4$ $M_{12} = R1$
G_2	M_{21} M_{22}
G_3	$M_{31} = R4$ $M_{32} = R2$ $M_{32} = R5$
G_4	$M_{41} = A$
G_5	$M_{51} = R3$

Si può notare quindi che molte delle metriche sono state ricondotte a quelle già consolidate ed altre invece appaiono molto più legate allo sviluppo del prodotto come per esempio le metriche M_{21} e M_{22}.

Per queste metriche si identificheranno i seguenti fuzzy set:

M_{21} L: <1 mese , M: 1<=M_{21}<=2 mesi, H: >2 mesi

M_{22} L: 0-40%, M: 41-90 % , H: 91-100 %

4.6 Fuzzy rule

Ora che si sono definite le metriche si procederà con la definizione delle fuzzy rule.

Prima di partire si riprenderà il documento di identificazione dei requisiti e si cercherà di realizzare in formule fuzzy ciò che viene descritto nel linguaggio naturale. Il modulo PRODUCTION GO (conosciuto anche come modulo per «ZTotem») nel product roadmap era caratterizzato dai seguenti obiettivi:

1. permettere la timbratura di produzione ad un dipendente,
2. gestire in maniera controllata interattiva e guidata la consuntivazione delle ore di produzione, la sincronizzazione remota delle anagrafiche e richiedendo on demand le tabelle di configurazione,
3. essere in grado di lavorare offline ed online, memorizzando dati ed inviando timbrature al server.

Come descritto nel Capitolo 3, si devono definire delle regole e si verificheranno gli obiettivi descritti. Sarà compito del project manager (o in generale della figura del valutatore) definire tutte le fuzzy rule che ritiene indispensabili al prodotto o al prototipo che si è sviluppato.

Ecco le fuzzy rule per «Production Go»:

Tabella 4.15: Fuzzy rule G1

M_{11}	M_{12}	**A**	**SLOC**	**E**	**COSTO**	**PROD.**	S_T	S_E	**Out**
L	**H**	**L**	**L**	**L**	**H**	**L**	**L**	**L**	**BAD**
M	**M**	**M**	**M**	**M**	**M**	**M**	**L**	**L**	**BAD**
L	**L**	**H**	**M**	**H**	**M**	**H**	**M**	**M**	**GOOD**
L	**L**	**M**	**M**	**L**	**H**	**L**	**H**	**H**	**GOOD**
M	**L**	**M**	**M**	**L**	**M**	**L**	**M**	**M**	**VERY GOOD**
M	**L**	**M**	**L**	**H**	**L**	**H**	**H**	**L**	**GOOD**
M	**M**	**H**	**H**	**M**	**M**	**M**	**M**	**M**	**GOOD**
M	**M**	**H**	**M**	**L**	**M**	**L**	**M**	**M**	**GOOD**
L	**L**	**H**	**L**	**M**	**M**	**M**	**H**	**H**	**EXCELLENT**

Tabella 4.16: Fuzzy rule G2

M_{21}	M_{22}	**A**	**SLOC**	**E**	**COSTO**	**PROD.**	S_T	S_E	**Out**
L	**L**	**L**	**L**	**H**	**M**	**H**	**L**	**L**	**BAD**
H	**M**	**L**	**M**	**M**	**H**	**M**	**L**	**L**	**BAD**
L	**H**	**H**	**M**	**H**	**M**	**H**	**M**	**M**	**VERY GOOD**
L	**H**	**M**	**M**	**L**	**H**	**L**	**H**	**H**	**GOOD**
M	**L**	**M**	**M**	**L**	**M**	**L**	**M**	**M**	**VERY GOOD**
M	**L**	**M**	**L**	**H**	**L**	**H**	**L**	**L**	**GOOD**
M	**M**	**H**	**H**	**M**	**M**	**M**	**M**	**M**	**VERY GOOD**
M	**M**	**H**	**M**	**L**	**M**	**L**	**M**	**M**	**GOOD**
L	**H**	**H**	**L**	**M**	**L**	**M**	**H**	**H**	**EXCELLENT**

Tabella 4.17: Fuzzy rule G3

M_{31}	M_{32}	M_{33}	**A**	**SLOC**	**E**	**COSTO**	**PROD.**	S_T	S_E	**Out**
H	**H**	**H**	**L**	**L**	**L**	**H**	**L**	**L**	**L**	**BAD**
M	**M**	**M**	**M**	**M**	**M**	**M**	**M**	**M**	**M**	**BAD**
H	**H**	**H**	**L**	**H**	**H**	**H**	**H**	**L**	**L**	**BAD**
M	**M**	**H**	**M**	**M**	**M**	**H**	**M**	**L**	**M**	**BAD**
M	**H**	**L**	**H**	**L**	**M**	**L**	**M**	**M**	**M**	**GOOD**
M	**M**	**M**	**H**	**M**	**M**	**L**	**M**	**M**	**H**	**GOOD**
L	**L**	**M**	**M**	**L**	**H**	**M**	**H**	**M**	**M**	**VERY GOOD**
L	**L**	**L**	**M**	**M**	**L**	**M**	**L**	**M**	**M**	**EXCELLENT**
L	**L**	**L**	**H**	**L**	**L**	**L**	**L**	**H**	**H**	**EXCELLENT**
H	**H**	**M**	**L**	**H**	**M**	**M**	**M**	**M**	**M**	**BAD**
L	**M**	**L**	**M**	**M**	**M**	**L**	**M**	**L**	**L**	**GOOD**
H	**H**	**M**	**L**	**H**	**M**	**M**	**M**	**M**	**M**	**BAD**
H	**H**	**M**	**L**	**L**	**L**	**H**	**L**	**L**	**M**	**BAD**

Tabella 4.18: Fuzzy rule G4

M_{41}	A	SLOC	E	COSTO	PROD.	S_T	S_E	Out
M	**M**	**M**	**L**	**L or M**	**L**	**M**	**M**	**EXCELLENT**
M	**M**	**M**	**M**	-	**M**	**M**	**M**	**GOOD**
H	**L**	**H**	**H**	**H**	**H**	**L**	**L**	**BAD**
L	**M**	**M**	**M**	**L**	**M**	**H**	**H**	**GOOD**

Tabella 4.19: Fuzzy rule G5

M_{51}	A	SLOC	E	COSTO	PROD.	S_T	S_E	Out
L or M	**M or H**	-	**L**	**M**	**L**	**M**	**M**	**EXCELLENT**
M or H	**L**	-	**M**	**L**	**M**	**L**	**L**	**GOOD**
H	**L**	**H**	**H**	**H**	**H**	**H**	**H**	**BAD**

Informazioni per la compilazione delle tabelle:

1. Ogni colonna delle tabella è in relazione AND con le altre colonne. Ogni cella può rappresentare essa stessa una relazione AND o OR. Per correttezza bisognerebbe completare ogni tabella con ogni possibile combinazione in modo tale da avere ogni caso coperto all'interno delle nostre regole.

2. Legenda dei simboli: L (Low), M (Medium), H (High) , $''-''$ (don't care, non interessante o indifferente alla valutazione). Ad esempio nel goal G1 se si possiede una probabilità bassa di «server down» (M_{11}) e con un tempo medio di risposta alto (M_{12}), una bassa affidabilità (A), poche righe di codice (SLOC), effort basso, costo elevato, produttività bassa, scostamento di tempo e scostamento di effort bassi $\Rightarrow$ si avrà una valutazione pessima (BAD) del prodotto.

Oltre a queste tabelle si devono poi definire le relazioni tra i vari goal (dal G1 al G5 in questo caso); queste relazioni rappresentano l'elemento di innovatività più grande rispetto alla HoQ (cfr. tabella sottostante).

Tabella 4.20: Fuzzy rule tra i goal

G1	G2	G3	G4	G5	OUT
BAD	**VERY G.**	**VERY G.**	**BAD**	**BAD**	**BAD**
GOOD	**GOOD**	-	**GOOD**	**GOOD**	**GOOD**
VERY G.	**VERY G.**	**EXCELL.**	**EXCELL.**	**EXCELL.**	**VERY G.**
GOOD	**BAD**	**BAD**	**BAD**	**GOOD**	**BAD**
VERY G.	**VERY G.**	**EXCELL.**	**VERY GOOD**	**EXCELL.**	**EXCELL.**

Per comprendere meglio questa tabella, prendiamo come esempio la prima riga: essa descrive che se avremmo una «disponibilità dei dati» (G1) bassa, una «comprensibilità del software» (G2) alta, una «disponibilità dell'applicativo» (G3) alta, bassa

«corenza dei dati» (G4) e bassa «sicurezza» (G5), il nostro prodotto sarà valutato come di basso livello (BAD).

Questa metodologia verrà confrontata nel prossimo capitolo con la HoQ, per cercare di capire i risultati ottenuti seguendo le due differenti versioni di monitoraggio e valutazione.

4.7 NFN

Per poter valutare complessivamente il prodotto ed attuare le fuzzy rule presenti nella Tabella 4.20, bisognerà creare una NFN generale che comprenda tutte le NFN di ogni goal da valutare.

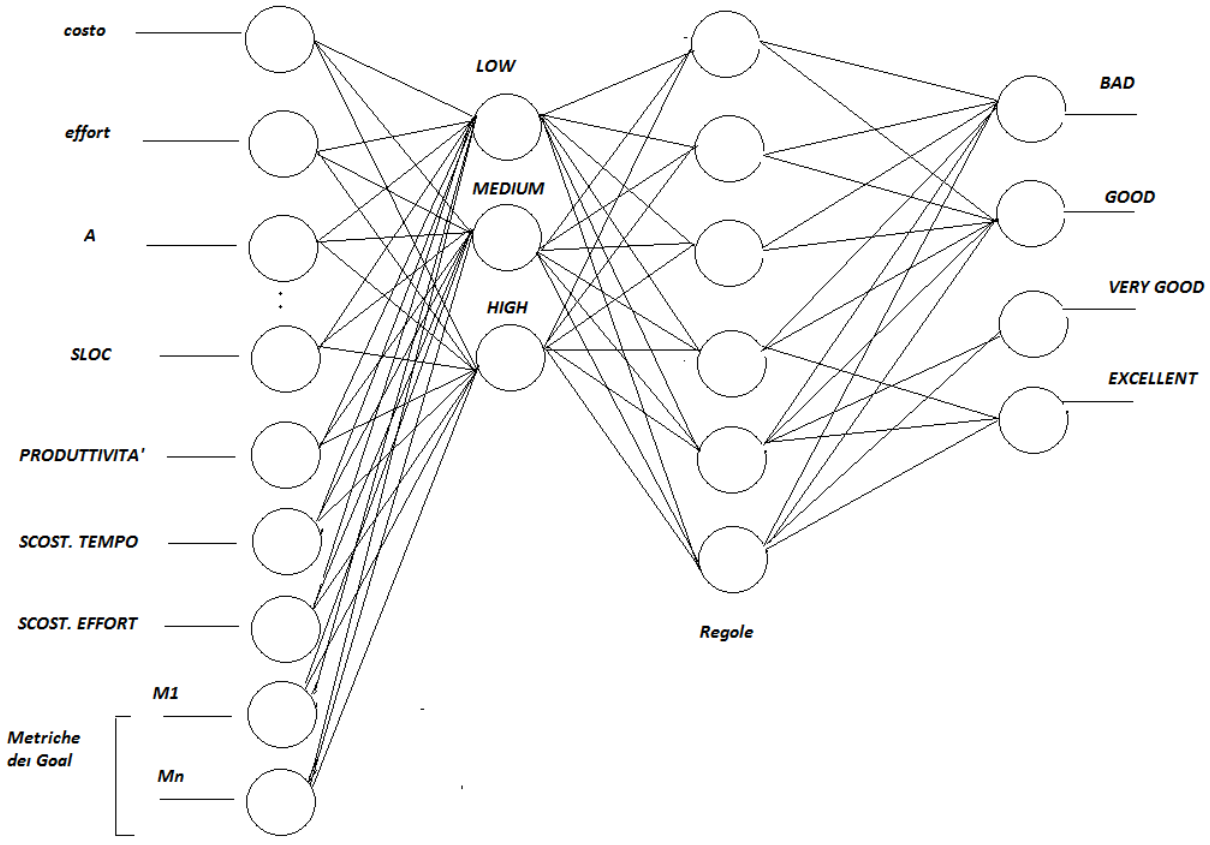

Figura 4.7: NFN schema

Ogni pallino rappresenta un «neurone» ovvero un entità che prende in ingresso degli input e fornisce degli output; il tutto darà vita ad una rete di neuroni in grado di apprendere e di valutare un risultato.

Utilizzando il framework Neuroph presentato nel Capitolo 3, si otterrà una NFN di questo tipo:

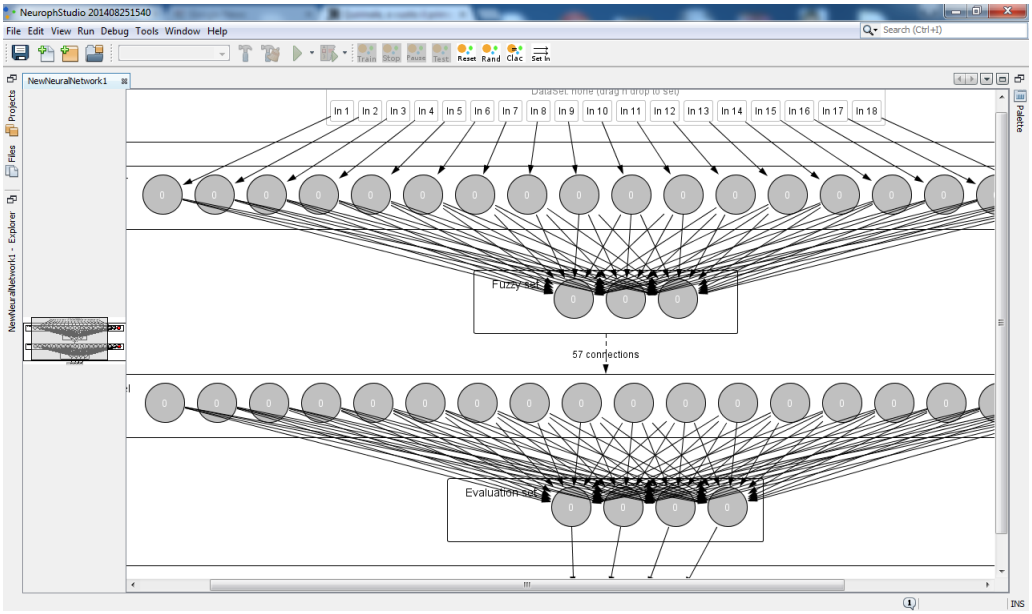

Figura 4.8: Production Go: Neuroph tool

Come detto anche nel capitolo precedente, non è necessario che si indichino tutte le possibili combinazioni di fuzzy rule per la valutazione, ma è necessario istruire efficacemente la rete NFN in modo tale che apprenda la giusta esperienza per valutare anche una combinazione di metriche non presente all'interno delle fuzzy rule.

Ora si mostra dettagliatamente questo vantaggio tramite un esempio.

Esempio:

Di seguito viene mostrata la fuzzy network che descrive la Tabella 4.20 per la valutazione generale del prodotto «Production Go» che si è costruita tramite Neuroph.

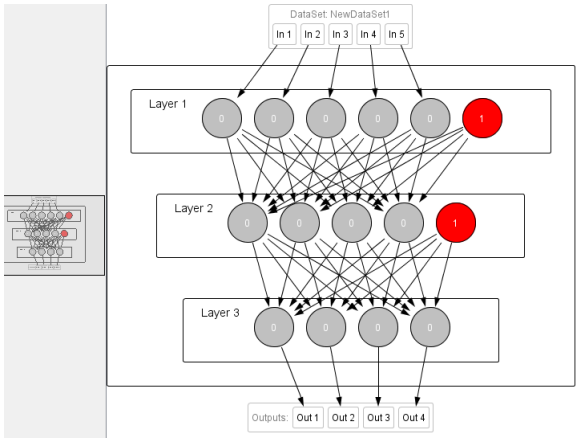

Figura 4.9: Production Go: Neuroph NFN

Gli ingressi $in1$, $in2$, $in3$, $in4$, $in5$ rappresentano gli ingressi della NFN che

sono rispettivamente G1, G2, G3, G4 e G5; in output i valori *out*1, *out*2, *out*3, *out*4 rappresentano rispettivamente i valori «BAD», «GOOD», «VERY GOOD» e «EXCELLENT».

Una volta costruita la NFN, in Neuroph per implementare le fuzzy rule è necessario creare un Dataset che la rete apprenderà in fase di training cliccando il tasto opportuno all'interno del programma.

Nel caso si studio si ha il seguente Dataset:

Input 1	Input 2	Input 3	Input 4	Input 5	Output 1	Output 2	Output 3	Output 4
1.0	3.0	3.0	1.0	1.0	1.0	0.0	0.0	0.0
2.0	2.0	0.0	2.0	2.0	0.0	1.0	0.0	0.0
3.0	3.0	4.0	4.0	4.0	0.0	0.0	1.0	0.0
2.0	1.0	1.0	1.0	2.0	1.0	0.0	0.0	0.0
3.0	3.0	4.0	3.0	4.0	0.0	0.0	0.0	1.0

Figura 4.10: Production Go: Neuroph Dataset

Il valore da 1 a 4 in input rappresenta la scala di valutazione dell'ingresso corrispondente: per esempio il valore 1 in *in*1 rappresenta il fuzzy set «BAD», il valore 3 in corrispondenza di *in*2 rappresenta il fuzzy set «VERY GOOD» e così via con gli altri valori.

Il valore 1 in output invece indica l'attivazione dell'output corrispondente ovvero: il valore 1 in corrispondenza di *out*1 indica un livello «BAD», il valore 1 su *out*2 indica un livello «GOOD», il valore 1 su *out*3 indica un livello «VERY GOOD» e infine il valore 1 su *out*4 indica un livello «EXCELLENT» del prodotto sviluppato.

La tabella del Dataset sarà quindi implementata seguendo la Tabella 4.20.

Si suppone a questo punto di voler far effettuare una valutazione del prodotto sviluppato dopo aver ottenuto ipoteticamente questi valori in input:

Tabella 4.21: Test: risultati ottenuti

G1	G2	G3	G4	G5
BAD	**VERY G.**	**BAD**	**BAD**	**BAD**

Come si può notare, questo insieme di valori fuzzy non fa matching con nessuna delle fuzzy rule presentate nella Tabella 4.20, di conseguenza la rete NFN dovrebbe autonomamente adattarsi e mostrare un risultato in base alle esperienze passate e al training delle fuzzy rule.

Infatti, dopo aver effettuato la fase di training della NFN tramite il comando opportuno («Train»), si sottopone la rete alla valutazione dei seguenti valori in input (seguendo le codifiche dei livelli descritte precedentemente):

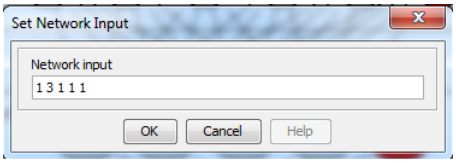

Figura 4.11: Production Go: Neuroph input di test

Cliccando su il tasto «OK» si ottiene così il seguente risultato:

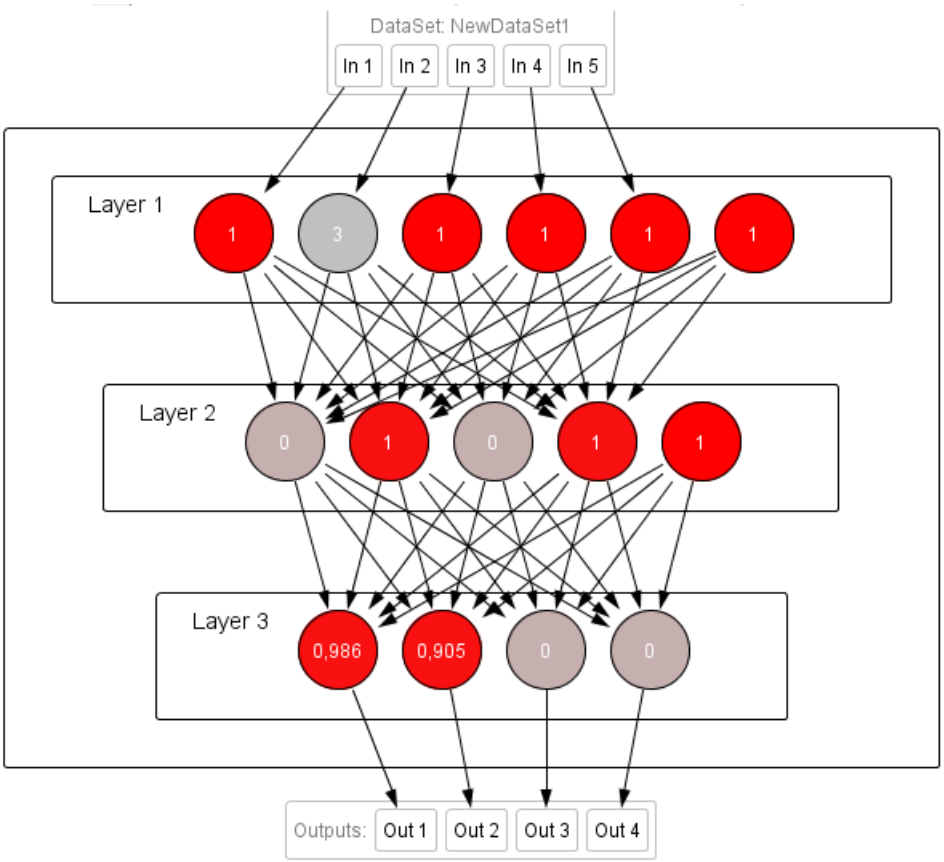

Figura 4.12: Production Go: Neuroph NFN test

Come si può notare si è ottenuto un valore $0,986$ «BAD» e un valore $0,905$ «GOOD»; in generale si può considerare il prodotto «BAD» visto i risultati ottenuti.

Questo risultato quindi dimostra la capacità delle NFN di dare una valutazione sempre e comunque anche in mancanza di fuzzy rule; il valore ottenuto darà indicazione sul livello di risultato ottenuto ovvero «BAD», «GOOD», «VERY GOOD» o «EXCELLENT».

Nel prossimo capitolo si procederà veramente attraverso la valutazione del progetto, con la definizione dei livelli fuzzy e con l'uso di dati reali di questo caso di studio, confrontando questa metodologia con la tecnica QFD (tramite l'utilizzo della HoQ).

Nell'Appendice A invece vi sarà poi presentata una possibile costruzione di un tool fuzzy che è possibile programmare ad hoc per il progetto che si vuole valutare (in questo caso il prodotto «Production Go»).

Capitolo 5

Valutazione

In questo capitolo si andrà ad analizzare il caso di studio descritto nel Capitolo 4 e si mostrerà il processo di valutazione: verranno fatti confronti con la metodologia QFD (tecnica di cui si vuole aggiungere un miglioramento nel processo di valutazione) e presentati i limiti e i vantaggi della metolodogia fuzzy.

5.1 Premessa

Si sono raccolti tutti i dati disponibili, si sono sviluppate linee di codice ed il prodotto è stato implementato (in via prototipale), arrivati a questo punto della fase di PM ora si deve procedere con il processo di valutazione.

Il prodotto sviluppato deve mostrare tutte le caratteristiche e le performance desiderate, ma soprattutto dovrebbe rispettare tutte le esigenze che si erano richieste in fase di stilazione dei requisiti.

La fase di valutazione si dimostra quindi come un «banco di prova» nei confronti del cliente (o del committente) ed è l'elemento cardine di un PM di cui non se ne può fare a meno.

Procedendo con lo sviluppo del caso di studio precedente, in questo capitolo verrà mostrato il processo di valutazione del software ZTimesheet in particolare del prodotto «Production Go».

Questa metodologia di valutazione può essere applicata anche a livello prototipale, ovvero considerando solo un ramo di sviluppo della feature model del prodotto completo, considerando quindi l'implementazione di solo alcune funzionalità richieste e non del prodotto completo.

5.2 Risultati ottenuti

Dopo aver costruito il framework, si raccolgono ora i «dati sperimentali» mostrando i calcoli di alcuni dei valori provenienti dalle metriche durante lo sviluppo

del caso di studio presentato nel capitolo precedente.

In Tabella 5.1 vengono raccolte l'elenco delle metriche con il valore trovato nel caso di studio:

Tabella 5.1: Valori trovati caso di studio

Metrica	Valore	Unità mis.
M_{11}	**10**	%
M_{12}	**12**	**sec**
M_{21}	**1**	**mese**
M_{22}	**30**	%
M_{31}	**15**	%
M_{32}	**5**	%
M_{33}	**5**	**min**
M_{41}	**5**	%
M_{51}	**1**	%
Affidabilità (A)	**86**	%
SLOC	**3**	**KLOC**
Effort	**13,45**	**mese uomo**
Produttività	**11821,56**	**FPs/mese uomo**
Costo (N° sviluppatori)	**7**	-
Scost. di tempo	**0,92**	-
Scost. di effort	**0,90**	-

Per il loro calcolo si sono utilizzate le formule presentate precedentemente nel Capitolo 3:

$$A = \frac{MTTF}{MTTF + MTTR} \cong \frac{MTTF}{MTBF} \cong \frac{9}{10.5} \cong 0.86$$

$$MM = a(SLOC)^b = 3.6 * (3)^{1.20} = 13,45\, mesi - uomo$$

$$TDEV = c(MM)^d = 2.5 * (13,45)^{0,32} = 5,74\, mesi$$

$$Produttività = FPs/MM = 159000/13,45 = 11821,56\, FPs/mesi - uomo$$

$$S_T = \frac{55\, giorni}{60\, giorni} = 0,92$$

$$S_E = \frac{13,45\, mesi - uomo}{15\, mesi - uomo} = 0,90$$

Per la metrica inerente il costo in questi caso si farà riferimento al numero di sviluppatori necessario N e prendendo come tempo T_{max} = 2 mesi e T_{min} =1 me-

se (come indicato nel caso di studio) di conseguenza, arrotondando per difetto, si otterrà:

$$N_{min} = \frac{MM}{T_{max}} = \frac{13,45}{2} = 6,72 \cong 6$$

$$N_{max} = \frac{MM}{T_{min}} = 13,45 \cong 13$$

L'utilizzo del numero di sviluppatori come unità di misura di costo è stata una scelta, nulla vietava di utilizzare il valore in euro del prodotto sviluppato e costruire su di esso dei fuzzy set.

5.2.1 Determinazione livelli fuzzy dei risultati

Per tutti i risultati ottenuti si procederà con la caratterizzazione del livello fuzzy a cui appartengono.

Si indicheranno con:

1. L: i valori LOW
2. M: i valori MEDIUM
3. H: i valori HIGH

Per determinare tutti i livelli fuzzy, si farà riferimento a questa tabella riassuntiva che racchiude tutte le metriche utilizzate con i rispettivi fuzzy set:

Tabella 5.2: Riassunto dei livelli fuzzy

Metrica	LOW	MEDIUM	HIGH
A	$(0-20)\%$	$(>20\,\&\,\leq 90)\%$	$(>90\,\&\,\leq 100)\%$
SLOC	$(0-2)$ KLOC	$(>2\,\&\,\leq 20)$ KLOC	>20 KLOC
Effort	$(0-15)$MM	$(>15\,\&\,\leq 20)$ MM	>20 MM
Prod.	$(0-15000)$FPs/MM	$(>15000\,\&\,\leq 20000)$ FPs/MM	>20000 FPs/MM
Costo	$(0–6)$	$(>6\,\&\,\leq 10)$	>10
R1	$(0-15)$ sec	$(>15\,\&\,\leq 20)$ sec	>20 sec
R2	$(0-9)$ %	$(>9\,\&\,\leq 50)$ %	$(>50\,\&\leq 100)$ %
R3	$(0-9)$ %	$(>9\,\&\,\leq 50)$ %	$(>50\,\&\leq 100)$ %
R4	$(0-9)$ %	$(>9\,\&\,\leq 50)$ %	$(>50\,\&\leq 100)$ %
R5	$(0-5)$ min	$(>5\,\&\,\leq 60)$ min	>60 min
S_T	>1	$(>0.5\,\&\,\leq 1)$	$0–0.5$
S_E	>1	$(>0.5\,\&\,\leq 1)$	$0–0.5$
M_{21}	$(0-1)$ mese	$(\geq 1\,\&\,\leq 2)$ mesi	>2 mesi
M_{22}	$(0-40)$ %	$(>40\,\&\,\leq 90)$ %	$(>90\,\&\leq 100)$ %

I risultati ottenuti della valutazione per le metriche sono:

- G1: Disponibilità dei dati

Tabella 5.3: G1: valori trovati

Metrica	Valore	unità mis.	Fuzzy set
M_{11}=$R4$	**10**	%	**M**
M_{12}=$R1$	**12**	**sec**	**L**

- G2: Comprensione del software

Tabella 5.4: G2: valori trovati

Metrica	Valore	unità mis.	Fuzzy set
M_{21}	**1**	**mese**	**M**
M_{23}	**30**	%	**L**

- G3: Disponibilità applicativo

Tabella 5.5: G3: valori trovati

Metrica	Valore	unità mis.	Fuzzy set
M_{31}=$R4$	**15**	%	**L**
M_{32}=$R2$	**5**	%	**L**
M_{33}=$R5$	**5**	**min**	**L**

- G4: Coerenza dei dati

Tabella 5.6: G4: valori trovati

Metrica	Valore	unità mis.	Fuzzy set
M_{41}=A	**80**	%	**M**

- G5: Sicurezza

Tabella 5.7: G5: valori trovati caso di studio

Metrica	Valore	unità mis.	Fuzzy set
M_{51}=$R3$	**1**	%	**L**

Altre metriche:

Tabella 5.8: Altre metriche: risultati ottenuti

Metrica	Valore	unità mis.	Fuzzy set
Affidabilità (A)	**86**	**%**	**M**
SLOC	**3**	**KLOC**	**M**
Effort	**13,45**	**mese uomo**	**L**
Produttività	**11821,56**	**FPs/mese uomo**	**L**
Costo (N° sviluppatori)	**7**	-	**M**
Scost. di tempo	**0,92**	-	**M**
Scost. di effort	**0,90**	-	**M**

Nei prossimi paragrafi si analizzeranno i risultati ottenuti attraverso l'utilizzo della metodologia con NFN e con QFD (tramite la HoQ).

5.3 Le metodologie utilizzate: NFN e QFD

Una volta stabiliti tutti i valori fuzzy, ricavati i dati e calcolati i valori utili alla valutazione si mostreranno ora gli sviluppi, i vantaggi ed i limiti ottenuti utilizzando questa metodologia rispetto ad altre.

Per cercare di fare una buona valutazione e per indicare quali siano gli aspetti innovativi e le migliorie introdotte, si presenterà la valutazione anche attraverso QFD.

Dopo aver discusso della metodologia, infine verrà presentato un intero paragrafo di discussione che metterà in relazione le due metodologie con NFN e QFD.

5.3.1 Neural Fuzzy Network

Il processo di valutazione attraverso NFN, avviene attraverso le fuzzy rule. Avendo a disposizione la tabella dei risultati ottenuti e tenendo in considerazione i livelli fuzzy a cui appartengono, rimane ora da applicare le fuzzy rule e di rendere operativa la NFN presentata nel Capitolo 4, ovvero:

1. si prendono in considerazione le metriche per ogni obiettivo,
2. all'ingresso della NFN si praticheranno così i valori fuzzy riconducibili ai valori trovati,
3. la NFN analizzerà i valori confrontandoli con le fuzzy rule specificate tramite un matching dei livelli ottenuti.

Nel caso di studio, utilizzando le fuzzy rule introdotte, si è ottenuto questo risultato:

1. G1:

$$(M \wedge L) \wedge (M \wedge M \wedge L \wedge M \wedge L \wedge M \wedge M) \Rightarrow VERY\ GOOD$$

2. G2:

$$(M \wedge L) \wedge (M \wedge M \wedge L \wedge M \wedge L \wedge M \wedge M) \Rightarrow VERY\,GOOD$$

3. G3:

$$(L \wedge L \wedge L) \wedge (M \wedge M \wedge L \wedge M \wedge L \wedge M \wedge M) \Rightarrow EXCELLENT$$

4. G4:

$$(M) \wedge (M \wedge M \wedge L \wedge M \wedge L \wedge M \wedge M) \Rightarrow EXCELLENT$$

5. G5:

$$(L) \wedge (M \wedge M \wedge L \wedge M \wedge L \wedge M \wedge M) \Rightarrow EXCELLENT$$

Nella prima parentesi vengono indicati i livelli fuzzy del valori trovati per le metriche provenienti dai goal (per esempio nel caso G1 il primo M indica il livello MEDIUM per la metrica M_{11}, il secondo valore indicato con L indica il livello LOW per la metrica M_{12}) e nella seconda parentesi il livello fuzzy del valore trovato per le metriche consolidate (il primo M indica il valore MEDIUM per l'affidabilità, il secondo M indica il valore MEDIUM per SLOC, il terzo valore indica il fuzzy set dell'effort, il quarto rappresenta il costo, il quinto rappresenta il fuzzy set della produttività e gli ultimi due rappresentano rispettivamente i fuzzy set dello scostamento di tempo e di effort).

Il valore posto dopo il simbolo $\Rightarrow$ indica invece la valutazione per quell'obiettivo considerato che deriva dall'applicazione delle fuzzy rule.

A questo punto si può ora procedere con l'analisi «globale» della qualità del prodotto, ovvero:

1. si deve prendere in considerazione ogni valutazione ottenuta per ogni goal utilizzato (in questo caso di studio si sono ottenuti 2 valutazioni $VERY\,GOOD$ e 3 valutazioni $EXCELLENT$),

2. dopodichè, utilizzando le fuzzy rule presentate in Tabella 4.20 si è ora in grado di definire globalmente la nostra valutazione ovvero si ordinano gli obiettivi ottenendo il seguente risultato (si indicano i valori $VERY\,GOOD$ con VG, mentre i valori $EXCELLENT$ con E):

$$VG \wedge VG \wedge E \wedge E \wedge E \Rightarrow VG$$

Al termine del processo di valutazione si otterà quindi che «Production Go» avrà una valutazione $VERY\,GOOD$ a livello di qualità di prodotto. Da questo risultato nascono una serie di considerazioni che si affronteranno nel paragrafo successivo.

5.3.1.1 Considerazioni

Applicato il metodo con NFN, si può notare che:

1. la valutazione complessiva del prodotto «Production Go» è $VERY\,GOOD$; questo risultato potrebbe andare contro le attese in quanto, visto il numero di $EXCELLENT$ ottenuti era in numero maggiore rispetto ai $VERY\,GOOD$, era auspicabile considerare la valutazione complessiva come $EXCELLENT$. Questo fa notare la forza delle fuzzy rule e la loro importanza all'interno di un meccanismo di valutazione.

2. La metodologia utilizzata in un primo momento è di tipo «goal based», vengono posti in primo piano gli obiettivi trovati tramite GQM; viene dato un livello fuzzy alle metriche utilizzate e, attraverso le fuzzy rule, si dà una valutazione del goal stesso e non al processo sviluppato.

3. La metodologia goal free e decision making in questa tecnica si trovano attraverso la valutazione finale del prodotto, dove viene messa in primo piano la valutazione del programma in maniera complessiva, misurando il raggiungimento degli obiettivi.

La metodologia si è dimostrata completa a livello di valutazione in quanto ha permesso di guidare lo sviluppatore e/o il project manager ad una valutazione oggettiva tramite il minimo sforzo: infatti lo sforzo più grande è stato quello di determinare i livelli fuzzy e di traformare i risultati numerici ottenuti nei rispettivi livelli fuzzy definiti, dopodichè il processo di valutazione è stata una pratica pressochè gratuita (ci ha pensato la NFN a fornire i risultati).

La NFN può essere implementata facilmente ed è rappresentata tramite un prodotto software che può essere utilizzato facilmente all'interno di qualsiasi processo di valutazione; essa prenderà come parametro il numero di goal da implementare e le regole da rispettare.

5.3.1.2 Limiti

In questo paragrafo si vuole cercare di definire quali sono i limiti della metodologia fuzzy utilizzata.

Se il processo di valutazione può sembrare molto veloce utilizzando la metodologia fuzzy, molto più difficile sarà studiare i meccanismi adatti per la costruzione delle fuzzy rule.

Infatti, una delle maggiori difficoltà di questa metodologia è quella di riuscire a costruire delle fuzzy rule che siano complete, ovvero che riescano a coprire tutte le combinazioni delle valutazioni di ogni goal (pratica non semplice).

Un altro limite che si può trovare all'interno di questa metodologia di valutazione sta nel fatto che i goal provenienti da GQM e non riconducibili a metriche standard

possono essere molto diversi da un prodotto all'altro, quindi non si riescono a dare delle metriche consolidate utili alla valutazione di qualsiasi prodotto.

Però c'è da dire che i limiti sono inferiori rispetto ai vantaggi che può dare tenendo in considerazione i meccanismi di valutazione già presenti, come per esempio il QFD.

5.3.2 QFD

Di seguito viene mostrata la HoQ al caso di studio presentato precedentemente:

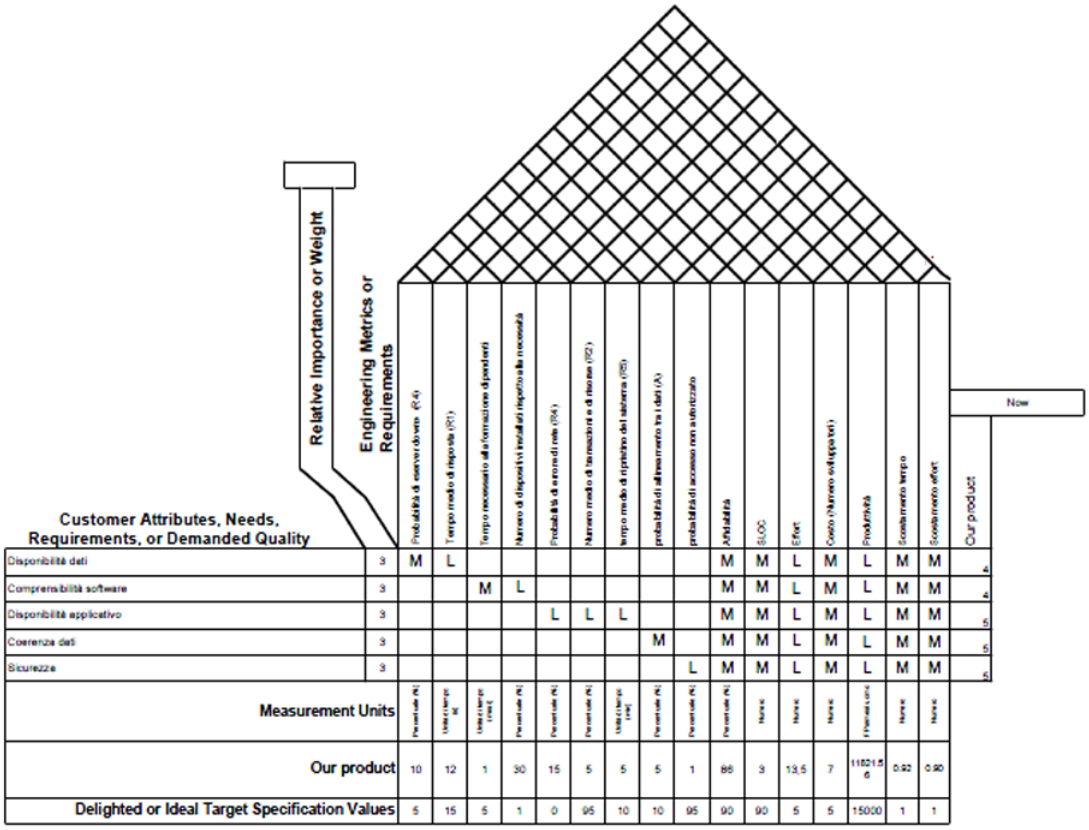

Figura 5.1: Production Go: House of Quality

Ecco come è stata costruita questa HoQ:

1. Per ogni goal ottenuto tramite GQM, si è dato loro un valore di importanza in una scala da 1 a 5 (più il valore cresce e più il goal è importante), ponendo il valore corrispondente nella colonna chiamata «Relative Importance or Weight». In questo caso sono stati messi tutti i valori pari a «3» in modo tale da non dare una maggior rilevanza di un goal rispetto ad un altro, ma ponendo tutti sullo stesso piano.

2. Per ogni goal, si sono riportati i livelli fuzzy ottenuti nel nostro sviluppo per ogni metrica utilizzata. I valori trovati nel nostro sviluppo sono indicati nella riga «Our product» sopra la colonna «Delighted or Ideal Target Specification Values».

3. Al di sotto della HoQ si riportano i valori ottenuti dallo sviluppo ed i valori di riferimento attesi (colonna «Delighted or Ideal Target Specification Values»).

4. Nella colonna «NOW» vengono posti invece i valori corrispondenti a «quanto buono» è il prodotto in seguito ai risultati ottenuti. In questa colonna viene inserito un valore compreso da 1 a 5. A questi valori, per proporre un analogia tra QFD ed il framework proposto, si devono ricavare i fuzzy set inerenti i valori ottenuti; si è quindi deciso di dividere la scala di valori da 1 a 5 omogeneamente in questo modo (nella parte di destra si trova il valore numerico, nella parte di sinistra il fuzzy set corrispondente):

   (a) 1: «BAD»

   (b) 2: «GOOD»

   (c) 3 - 4: «VERY GOOD»

   (d) 5: «EXCELLENT»

5.3.2.1 Considerazioni

Una volta costruita la HoQ, si può notare che:

1. La codifica della scala di valutazione nei corrispettivi fuzzy set è distribuita in modo non omogeneo; infatti, considerando il valore 3 nel fuzzy set «VERY GOOD» si è data una visione meno stringente dello studio della qualità del prodotto. Se si volesse attuare una suddivisione molto più restringente, si potrebbe riportare il valore 3 presso il livello fuzzy inferiore ovvero «GOOD». Attuando questa scelta si avranno quindi valutazioni che possono essere differenti ma che ai fini del nostro caso di studio rappresentano solo un dettaglio.

2. Analizzando i goal ottenuti, nella HoQ si sono ottenuti valori «VERY GOOD» per quanto la disponibilità dei dati (G1) e la comprensibilità del software (G2); si è invece ottenuta un'eccellenza («EXCELLENT») per quanto riguarda la disponibilità dell'applicativo (G3), la coerenza dei dati (G4) e la sicurezza (G5). In prima analisi questo poterebbe lo sviluppatore o il project manager a valutare il prodotto in maniera molto positiva: stando infatti alla statitica, si sono ottenuti maggior valutazioni «EXCELLENT» piuttosto che «VERY GOOD».

3. E' stata adottata una metodologia di tipo goal based per l'evaluation del prodotto, in quanto in primo piano sono stati messi gli obiettivi da raggiungere da parte dei clienti; la valutazione infatti si basa sul fatto che il prodotto è buono se vengono rispettati gli obiettivi prefissati. Utilizzando il QFD si ragiona utilizzando la formula presentata nel Capitolo 2, ovvero che la valutazione di un prodotto i (v_i) dipende dalla misura in cui i aiuta o ostacola il raggiungimento di un obiettivo j pesato dall'importanza del goal (g_j), sommato su tutti gli obiettivi j. Gli obiettivi infatti vengono pesati con la loro importanza («Relative Importance or Weight») e registrata la sua misura ed i valori attesi («Delighted or Ideal Target Specification Values»). L'obiettivo è posto al centro di tutte le attività e processi sviluppati.

4. La focalizzazione dell'informazione (ovvero il delineamento degli obiettivi), la sua organizzazione nella HoQ e la sintetizzazione dei risultati ottenuti ci conduce anche ad una visione «decision making» della valutazione. La fase di «context», di «input» e di «Process» del CIPP (cfr. Capitolo 2), sono fasi antecendenti alla HoQ che rappresentano infatti la fase di «Product» attraverso la misurazione del raggiungimento degli obiettivi dopo la realizzazione del programma.

Il meccanismo QFD quindi si è dimostrato abbastanza efficace e trasversale alla valutazione, toccando tutte le tre tecniche di valutazione (goal based, goal free e decision making).

Per ogni goal quindi si è stati in grado di descrivere a quale tipologia di valutazione siamo arrivati, ma il limite più grande che possiamo trovare nel QFD è quello dell'impossibilità di atturare un modello «prospettista» della valutazione, ovvero nel raggruppare tutti gli obiettivi e di dare una valutazione completa del prodotto:si potrebbero fare delle supposizioni, ovvero fare una media tra i risultati ottenuti o dare delle conclusioni senza alcun carattere oggettivo (per esempio: consideriamo il prodotto «EXCELLENT» in quanto abbiamo ottenuto più eccellenze nei goal).

Di questo limite ne verrà parlato molto più in dettaglio nel prossimo paragrafo.

5.3.2.2 Limiti

Si è notato che nel QFD il punto di partenza è la "voice of customer" collegata ad una serie di parametri tecnico-ingegneristici.

Nella HoQ si raggiunge la qualità di un prodotto, ma siamo sicuri di considerare proprio tutto? Senza che vi sia presente un WHATS non si potrebbe capire se si è raggiunta una qualità ottimale per il prodotto? Se esistessero dei limiti aziendali in termini di risorse, si potrà raggiungere lo stesso l'esigenza del cliente? Se un' esigenza dipendesse indirettamente o direttamente da un'altra esigenza?

Esempio:

per garantire l'esigenza di mercato X, si dovrebbero disporre di Y sviluppatori che impiegano T mesi con un costo pari a C a sviluppare il prodotto. Alla fine della produzione l'azienda sarà stata in grado di arrivare a garantire questa esigenza.; ma se per la manutenzione del prodotto ci vorrà un costo pari a P? Ovvero: si è disposti a raggiungere l'esigenza X sapendo che l'esigenza J poi non è raggiunta?

In questo caso nella HoQ non possiamo avere un legame tra le diverse esigenze, non permettendo così una valutazione congiunta delle richieste. Di conseguenza dobbiamo essere in grado di estendere il concetto di qualità incrociandolo con tutti gli elementi a disposizione, ovvero non facendo solo un prodotto cartesiano tra gli obiettivi (le esigenze) e i requisiti, ma anche tra obiettivi ed obiettivi e tra requisiti e requisiti.

In termini matematici, se si indica con $O = obiettivi$,$R = Richieste$ si deve essere in grado di far questo prodotto cartesiano:

$$O \times O$$

$$R \times R$$

Tutto ciò il framework NFN riesce a farlo tramite l'utilizzo delle fuzzy rule.

Esempio:

In un sistema input/output si ha a disposizione una variabile di ingresso u all'istante di tempo $t-1$ e $t-2$ e una variabile di uscita y all'istante $t-1$. Ingresso ed uscite sono legate dalla seguente relazione:

$$y_t = f(y_{t-1}, u_{t-1}, u_{t-2})$$

se volessimo descrivere l'uscita che otterremmo al tempo t in determinate condizioni di ingresso/uscita (valori A,B,C,D) potremmo descriverla attraverso la seguente fuzzy rule:

$$R_1 : IF\, y_{t-1}\, is\, A\, AND\, u_{t-1} is\, BAD\, u_{t-2} is\, C\, THEN\, y_t\, is\, D$$

Il successo del modello quindi consiste nell'abilità di includere nelle regole (fuzzy rule) tutti i comportamenti del sistema e scegliere in modo efficace i rappresentanti di A,B,C,D basandosi sui dati osservati.

In questo caso avremmo quindi solo il compito di inserire ogni cosa, non solo la «voice of customer» in relazione con i parametri ingegneristici, ma anche tra

«voice of customer» e «voice of customer» o tra parametri ingegneristici e parametri ingegneristici (ciò che il QFD non permetterebbe).

5.3.3 Neural Fuzzy Network vs QFD

Una volta applicata la valutazione tramite QFD e NFN, in questo paragrafo si vogliono mostrare i risultati ottenuti attuandone un confronto.

Come notato precedentemente la flessibilità e l'incontrollabilità dei fattori tecnologici può portare il manager verso una scelta errata, è necessario quindi effettuare studi e introdurre regole che permettono di controllarne i risultati tenendo conto sia delle regole già prefissate sia delle esperienze passate (un pò come il nostro cervello) affidandoci meno ai calcoli probabilistici del momento.

Da questo punto di vista la NFN è il meccanismo migliore per risolvere questa problematica infatti, se si avesse di fronte una variazione tecnologica-ambientale del prodotto, tramite le fuzzy rule si è in grado semplicemente di gestirla introducendo una regola in più nella lista delle regole fuzzy.

Nel caso di QFD invece, la flessibilità sarà complicata da gestire in quanto l'aggiunta di una regola si dovrebbe rivalutare e ricostruire tutta la HoQ, perdendo quindi del tempo necessario che potrebbe essere utile per la messa in opera del prodotto stesso.

La flessibilità infatti è uno dei punti di forza delle reti neurali in generale, perchè si basano sull'esperienza come proprio i neuroni all'interno cervello umano.

Oltre alla flessibilità, si potrebbe anche studiare l'efficacia sia per NFN sia per QFD. L'efficacia di un prodotto software per la HoQ viene studiata suddividendo il prodotto stesso in «voice of customer» e dando loro una valutazione, ottenendo quindi non un risultato finale univoco ma suddiviso.

Prendendo come spunto la HoQ del caso di studio, si avevano a disposizione cinque «voice of customer» ed ad ognuno di essi è stata data una valutazione, ottendendo due valori «VERY GOOD» e tre «EXCELLENT».

In questo caso come si fa ad avere una valutazione completa del prodotto? Nella HoQ non è definito alcun metodo di valutazione globale del prodotto, ma si possono solo fare delle considerazioni di carattere generale, ovvero: notando che a livello qualitativo si sono ottenuti più risultati «EXCELLENT» rispetto a «VERY GOOD» la supposizione può essere quella di aver ottenuto un risultato «EXCELLENT».

Non esiste alcun metodo quantitativo nella HoQ che pemetta di confermare questa affermazione, ciò che invece nelle NFN si può fare tramite l'utilizzo delle fuzzy rule tra i goal ottenuti (analogamente tra le «voice of customer»); infatti in questo caso di studio se si utilizzassero le fuzzy rule definite si potrebbe arrivare ad una valutazione più completa ed oggettiva del prodotto.

Utilizzando la terza fuzzy rule presente in Tabella 4.20, infatti si noterebbe che il prodotto ottenuto non è «EXCELLENT» ma «VERY GOOD», andando contro le supposizioni fatte.

Tutto ciò significa che ad ogni goal è stata data una sorta di «importanza pesata» in relazione a tutti gli altri goal (quindi variabile a seconda del contesto) ovvero, tramite le fuzzy rule, vi è la possibilità di specificare che se un goal A è «VERY GOOD» e se tutti gli altri goal dovessero essere indicati come «BAD», la valutazione del prodotto potrebbe essere «VERY GOOD» (anche se la maggior parte dei goal è «BAD»).

Nella HoQ questo concetto può essere parzialmente utilizzato solo attraverso un «peso assoluto del goal»: come si nota nella colonna «Relative importance or Weight» si potrebbe dare un peso relativo ad ogni «voice of customer», quindi se si avrà un peso elevato allora quasi certamente sarà il requisito più importante, ma non è possibile però definire un peso di una «voice of customer» in relazione ad ogni «voice of customer».

Prendendo come spunto il caso di studio si può spiegare che il risultato «VERY GOOD» ottenuto è dovuto soprattutto al fatto che la disponibilità dei dati (goal G1) è molto importante in relazione agli altri goal quindi si avrà un risultato relazionato al valore di G1; al contrario, nel caso però si ottenesse un'eccellente sicurezza (G5=«EXCELLENT») mantenendo una buona disponibilità dei dati (G1=«VERY GOOD»), si potrebbe avere un valore «EXCELLENT» del prodotto; quindi il peso di ogni goal all'interno di una NFN non è fisso ma varia a seconda del contesto e della valutazione degli altri goal.

5.4 Riflessione

Al termine della fase di valutazione si vuole ora proporre in questo paragrafo un breve spunto di riflessione sulle metodologie presentate ed utilizzate in questa tesi.

All'interno dello studio presentato si è avuto modo di notare che le due metodologie proposte possono arrivare a due risultati differenti a livello di valutazione: nel caso di «Production Go» con QFD si è ottenuta una valutazione EXCELLENT, mentre attraverso la NFN si è ottenuta una valutazione VERY GOOD.

A questo punto quindi rimane da capire quale sia la valutazione più corretta tra le due presentate o se si potrebbero utilizzare entrambe considerando i criteri e gli ambiti di applicabilità.

La NFN va a colmare molti limiti che il QFD possiede proponendo una valutazione complessiva del prodotto o del prototipo che sia; c'è da sottolineare però che le due metodologie presentate utilizzano punti di vista differenti, ovvero QFD utilizza quello dei customer mentre NFN utilizza un punto di vista «misto» tra customer/fornitore.

La domanda che ci si potrebbe fare è la seguente: «al termine della fase di valutazione è più importante il punto di vista dei customer o dei fornitori del servizio messi in relazione con i customer?».

Una valutazione ottima dal punto di vista dei customer implicitamente fornisce una soddisfazione e una reputazione aziendale elevata nei confronti dei fornitori; non sempre però ciò che è ottimo in assoluto per un customer è ottimo in assoluto anche per il fornitore.

Ovviamente l'obiettivo di un'azienda operante in qualsiasi settore (non solo nel mondo ICT) è quello di realizzare prodotti che soddisfino pienamente il cliente nelle loro richieste; dopodichè, conoscendo il proprio prodotto il fornitore del servizio è in grado inoltre di sapere se si possono avere dei margini di miglioramento generali.

In questo caso di studio, «Production Go» è stato un prodotto EXCELLENT per quanto riguarda il customer, il fornitore (Zucchetti) potrebbe quindi decidere di fermarsi sostenendo che il prodotto ottenuto è terminato nel migliore dei modi in quanto soddisfa le richieste del cliente; ragionando però in questi termini, non si tende a considerare ad un eventuale miglioramento futuro ovvero ci si accontenta del risultato ottenuto ma non si potrà fare niente in caso in cui il cliente chiederà qualcosa in più.

Tramite NFN quindi si ha la possibilità di guardare anche in prospettiva, analizzando tutti i punti di vista. A seconda della politica aziendale sarà possibile fare alcune scelte invece che altre, ovvero sarà possibile capire se fermarsi al punto di vista del customer o se proseguire tramite un analisi più approfondita del prodotto con NFN.

Questa scelta è ciò che si vuole proporre al lettore come spunto di riflessione, quale può essere il compromesso da raggiungere? Come compromesso si potrebbero per esempio utilizzare entrambe le metodologie in parallelo per poi effettuare le opportune valutazioni.

L'argomento è ampio e può toccare anche tematiche di tipo manageriale che dipendono fortemente da azienda ad azienda, ovvero da fornitore a fornitore.

5.5 Conclusioni

Il meccanismo con NFN si è dimostrato molto più efficiente in termini di valutazione, essendo molto più performante in termini di tempo, flessibilità e qualità. Esso pone quindi delle migliorie a livello di valutazione, improntandosi trasversalmente tra le tecniche goal based e decision making.

La metodologia presentata presenta quindi una nuova tecnica di valutazione che si è dimostrata molto utile in questo caso di studio, ma che può essere di grande aiuto per tutti quei prodotti provenienti dal mondo ICT in quanto:

1. l'incertezza della tecnologia,

2. la continua innovazione che porta a tempi sempre più ristretti tra la fase di identificazione dei requisiti e la fase di installazione del prodotto,

3. la qualità sempre più richiesta a livello software,

obbligano i project manager ad utilizzare meccanismi che siano efficaci e allo stesso tempo semplici da utilizzare.

In questo caso quindi la metodolgia NFN rappresenta un'ottima soluzione.

Capitolo 6

Conclusioni

Lo scopo di questa tesi era quello definire una metodologia con l'obiettivo di valutare i prodotti software sviluppati (anche solo a livello prototipale) considerando l'elevata incertezza e i continui cambiamenti in ambito tecnologico.

Il meccanismo con NFN si è dimostrato essere più performante in termini di flessibilità in caso di incertezze elevate e di efficacia in termini di valutazione.

Tutto ciò ha dimostrato la grande applicabilità delle logiche fuzzy che sono tipiche del soft computing, ma astraibili in svariati ambiti in modo molto semplice immediato ed efficace come in questo caso.

Il Capitolo 5 si è concluso con una riflessione che porta a considerare quanto la pratica della valutazione sia del tutto complessa che pone nella soggettività la maggior parte dei suoi fondamenti.

La riflessione fatta può essere considerata quindi come una prospettiva futura di studio nel campo della valutazione dei progetti, ovvero si deve cercare di indagare e di stilare eventualmente delle guideline oggettive che permettano di capire quale meccanismo possa essere più efficace non in assoluto, ma all'interno della propria realtà aziendale.

In questo caso di studio infatti si è preso come punto di rifermento la software house «Zucchetti», che vanta più di 100.000 clienti e che fa della qualità del prodotto la sua forza, di conseguenza se da un lato vi è la soddisfazione del cliente, dall'altro ci deve essere a tutti i costi anche la soddisfazione del fornitore.

Tenere in considerazione non solo il punto di vista del cliente specifico ma anche il punto di vista del fornitore è una scelta che offre all'azienda il pregio di non fermarsi mai al singolo sviluppo, dandole l'oppotunità di continuare ad investire ed ad innovare; infatti in questo caso accettando un valore «VERY GOOD» del prodotto con NFN, si indica che ci possono essere ancora dei margini di miglioramento che si possono o che si potevano applicare.

In generale la NFN mostra molte più aspettative, molta più oggettività e molte più garanzie rispetto al QFD, ma che possono dimostrarsi in alcuni casi anche troppe

per altre realtà aziendali che magari puntano a risultati eccellenti ma con profili inferiori.

Come ricerca e studio futuro si potrebbe validare il meccanismo NFN su più casi di studio e focalizzarsi sull'automatizzazione della metodologia.

Per quanto riguarda invece il meccanismo NFN esistono anche altri punti aperti che possono portare ad una miglioria della metodologia ed un incremento ancora più sostanziale dell'oggettività: ovvero la costruzione delle fuzzy rule.

La costruzione delle fuzzy rule in questa metodologia è un meccanismo che si basa maggiormente sullo standard aziendale e sulle esperienze passate, soprattutto nella definizione dei livelli fuzzy.

Sarebbe ancora più efficace avere a disposizione degli automatismi che identifichino i livelli fuzzy e che ne costruiscano poi le relative fuzzy rule.

Bibliografia

[1] ISO/IEC 9126. Information technology - Software Product Evaluation - Quality characteristics and guidelines for their use. 1991.

[2] A.J. Albrech. Measuring Application Development Productivity. *IBM*, 1979.

[3] Mario Cislaghi Alfredo Avellone, Ercole Colonese. Collaudo e qualitá del software. pages 250–251, 2010.

[4] Barry Boehm. Software Engineering Economics. *IEEE TRANSACTIONS ON SOFTWARE ENGINEERING*, 10, 1984.

[5] Spadoni G. Bonvicini S, Leonelli P. Risk analysis of hazardous materials transportation: Evaluating uncertainty by means of fuzzy logic. *J Hazard Mater*, 62(1):59–74, 1998.

[6] Allyssa Ingraham Brandon W. Youker. Goal-Free Evaluation: An Orientation for Foundations Evaluations. 5(7):51–61, 2014.

[7] Dino Mandrioli Carlo Ghezzi, Mehdi Jazayeri. Ingegneria del software. Fondamenti e principi. 2004.

[8] Tah JHV Carr V. A fuzzy approach to construction risk assessment and analysis: Construction project risk management system. *Adv Eng Software*, 32:847–857, 2001.

[9] Gulcin Buyukozkan Cengiz Kahraman, Tijen Ertay. A fuzzy optimization model for QFD planning process using analytic network approach. 171(2):390–411, 2006.

[10] Sue Fen Huang Chen Tung Chen. Applying fuzzy method for measuring criticality in project network. 177(12):2448–2458, 2007.

[11] Kim Y.B. Cho H.N., Choi H.H. A risk assessment methodology for incorporating uncertainties using fuzzy concepts. *Reliab Eng Syst Saf*, 78(2):173–183, 2002.

[12] Ercole Colonese. Come misurare le diverse caratteristiche del software: dimensioni, qualitá, impegno richiesto per lo sviluppo. pages 50–62, 1999.

[13] Kuchta D. Use of fuzzy numbers in project risk (criticality) assessment. *International Journal of Project Management*, 19(5):305–310, 2001.

[14] Ernesto Damiani. Il metodo COCOMO. 1991.

[15] David E. Herron David Garmus. Function Point Analysis: Measurement Practices for Successful Software Projects. 2001.

[16] Unione Europea. Gazzetta Ufficiale dell'Unione Europea L077. page 6, 23 Marzo 2005.

[17] Cengiz Kahraman Fatih Tuysuz. Project Risk Evaluation Using a Fuzzy Analytic Hierarchy Process: An Application to Information Technology Projects. 21(6):559–584, 2006.

[18] ITIL. Foundations of IT Service Management based on ITIL v3. 2008.

[19] Steve Jobs. Live from WWDC 2006: Steve Jobs Keynote. August 2006.

[20] Frank van der Linden Klaus Pohl, Gunter Bockle. Software Product Line Engineering: Foundations, Principles, and Techniques. pages 170–175, 2005.

[21] M. Knowlens. La formazione degli adulti come biografia. page 399, 1996.

[22] Dr. Koskey. Introduction to Educational Evaluation. *Institute for Systems Engineering and Automation*, 2013.

[23] Linux. Summary, Outlook, Statistics - The H Open: News and Features. December 2013.

[24] Microsoft. How Many Lines of Code in Windows XP. 2005.

[25] Fereidoon Shams Mir Ali Seyyedi, Mohammad Teshnehlab. Measuring Software Processes Performance based on the Fuzzy Multi Agent Measurements. *Proceedings of the International Conference on Information Technology: Coding and Computing*, 2005.

[26] Roger S. Pressman. Software engineering :A practitioner's approach. 2014.

[27] QMS. Quantitative Software Management, Inc. 2013.

[28] Paolo Giorgini Raian Ali, Fabiano Dalpiaz. A Goal-based Framework for Contextual Requirements Modeling and Analysis. pages 1–20.

[29] Victor R. Basili Gianluigi Caldiera1 H. Dieter Rombach. The Goal Question Metric Approch. *Institute for Systems Engineering and Automation*, pages 528–531, 1994.

[30] Gurpreet S. Sandhu and Kuldip S. Rattan. Design of a Neuro Fuzzy Controller. *Institute for Systems Engineering and Automation*, pages 3170–3175, 1997.

[31] Giuseppe Santucci. Qualitá nella Produzione del Software. 2008.

[32] Janet Shapiro. Monitoring and Evaluation. *CIVICUS -World Alliance for Citizen Partecipation*, pages 1–51, 2002.

[33] Andrzej Pieczynski Silva Robak. Employing Fuzzy Logic in Feature Diagrams to Model Variability in Software Product-Lines. 2003.

[34] IEEE Computer Society. A Guide to the Project Management Body of Knowledge. *Institute for Systems Engineering and Automation*, 2013.

[35] Shirin Sohrabi John Mylopoulos Sotirios Liaskos, Sheila A. McIlraith. Integrating Preferences into Goal Models for Requirements Engineering. *IEEE International Requirements Engineering Conference*, pages 135–143, 2010.

[36] Stijn M. J. van Osselaer and Chris Janiszewski. A Goal-Based Model of Product Evaluation and Choice. *Journal of Consumer Research, Inc*, pages 261–264, 2012.

[37] Eric S. Yu. Social Modeling and i*. 2014.

Appendice A

Costruzione di un tool NFN

Come approfondimento dei contenuti proposti, in questa sezione si vuole presentare il procedimento per realizzare un tool automatico specifico per il proprio prodotto o prototipo, che permetta di utilizzare il meccanismo NFN presentato in questa tesi.

Il tool che si propone da sviluppare è stato implementato utilizzando due linguaggi di programmazione molto diversi tra loro:

1. FCL (Fuzzy Control Language),
2. Java.

Mentre Java è un linguaggio molto conosciuto ed applicato quasi in ogni ambito, FCL invece è un linguaggio di programmazione molto diverso che si stacca dalle classiche tecniche di programmazione ad oggetti e che si classifica come un linguaggio di programmazione logico ed utile ad implementare il controllo fuzzy.

FCL è un linguaggio non interamente completo, standardizzato da IEC 61131-7 (standard dell'International Electrotechnical Commission per i controllori programmabili) ed integrato in alcune librerie java tra cui la «JFuzzyLogic.jar» (libreria proposta in questa sezione) il quale utilizza la programmazione logica supportando diverse membership function tipicamente fuzzy (ad esempio: trapezoidi e triangoli).

Lo sviluppo del tool presentato in questa sezione si servirà soprattutto dell'utilizzo dell'IDE di sviluppo «Eclipse», in quanto all'interno è fornito anche un plugin pienamente integrato per il linguaggio FCL.

Gli elementi fondamentali di un programma scritto in FCL sono:

1. FUNCTION_BLOCK: definizione del blocco di funzioni dove verranno incluse tutte le logiche fuzzy del sistema da analizzare.
2. VAR_INPUT: variabili di input del sistema.
3. VAR_OUTPUT: variabili di output.

4. FUZZIFY: fuzzificazione delle variabili di input e di output (definizione dei livelli fuzzy del controllo).

5. DEFUZZIFY: defuzzificazione delle variabili di output.

6. RULEBLOCK: blocco in cui vengono definite tutte le fuzzy rule da rispettare.

Ogni programma FCL può contenere diversi function block, di conseguenza diverse variabili in input ed in output e diverse regole fuzzy.

Per capire al meglio questo linguaggio si prenderà ora in cosiderazione il Goal 1 (G1) del caso di studio descritto nel Capitolo 4 di questo elaborato e su di esso si costruirà questo semplice programma utile al meccanismo di valutazione con NFN.

Dopo aver installato e preparato tutto l'ambiente di sviluppo all'interno dell'IDE Eclipse (http://jfuzzylogic.sourceforge.net/html/manual.html#plugin), si deve procedere innanzitutto con la definizione del file con estensione «.fcl»: in questo caso il file sarà chiamato «ZTotem.fcl».

In questo file verranno presentati tutti gli elementi fondamentali del linguaggio FCL descritti precedentemente, implementando a livello fuzzy il goal G1.

Di seguito viene presentato un estratto di questo file:

```
5
6  FUNCTION_BLOCK ZTotemG1 // Block definition (there may be more than one block per file)
7
8  VAR_INPUT               // Define input variables
9      A : REAL;
10     SLOC : REAL;
11     Effort : REAL;
12     Produttivita:REAL;
13     Costo: REAL;
14     R1:REAL;
15     R4:REAL;
16     ScostTime:REAL;
17     ScostEffort:REAL;
18 END_VAR
19
20 VAR_OUTPUT              // Define output variable
21     G1 : REAL;
22 END_VAR
23
24 FUZZIFY A           // Fuzzify input variable
25     TERM LOW := (0,0) (0, 1) (20, 1) (20,0) ;
26     TERM MEDIUM := (20, 0) (20,1) (90,1) (90,0);
27     TERM HIGH := (90, 0) (90, 1) (100,1) (100,0);
28 END_FUZZIFY
29
30 FUZZIFY SLOC            // Fuzzify input variable
31     TERM LOW := (0,0) (0, 1) (2,1) (2, 0) ;
32     TERM MEDIUM := (2,0) (2, 1) (20, 1) (20, 0) ;
33     TERM HIGH := (20, 0) (20, 1) (50, 1)(50, 0);
34 END_FUZZIFY
35
```

Figura A.1: File «fcl»: definizione variabili e fuzzificazione

Nel codice si può notare il function block inerente il goal G1 («ZTotemG1») con al suo interno tutte le variabili in input ed in output con i loro rispettivi livelli fuzzy; i valori tra parentesi dei livelli fuzzy per ogni variabile indicano invece i punti cartesiani

all'interno del grafico che verrà successivamente generato al momento dell'esecuzione del programma.

Il prossimo passo sarà quello di definire la defuzzificazione e le regole fuzzy che il sistema deve rispettare, ovvero di tutte le fuzzy rule che utilizza all'interno della valutazione. A questo scopo quindi si prenderanno come riferimento le regole fuzzy definite per G1 (Tabella 4.15) che verranno implementate nel file «fcl»:

```
DEFUZZIFY G1            // Defzzzify output variable
    TERM BAD := (0,0) (0,1)  (2, 1) (2, 0);
    TERM GOOD := (2,0) (2,1) (3, 1) (3, 0) ;
    TERM VERY_GOOD := (3,0) (3,1) (4, 1) (4, 0);
    TERM EXCELLENT := (4, 0) (4, 1) (5, 1) (5, 0);
    METHOD : COG;       // Use 'Center Of Gravity' defuzzification method
    DEFAULT := 1;       // Default value is 0 (if no rule activates defuzzifier)
END_DEFUZZIFY

RULEBLOCK No1
    AND : MIN;          // Use 'min' for 'and' (also implicit use 'max' for 'or' to fulfill DeMorgan's Law)
    ACT : MIN;          // Use 'min' activation method
    ACCU : MAX;         // Use 'max' accumulation method

    RULE 1 : IF R4 IS LOW    AND R1 IS HIGH   AND A IS LOW    AND SLOC IS LOW    AND Effort IS LOW    AND Costo IS HIGH
    RULE 2 : IF R4 IS MEDIUM AND R1 IS MEDIUM AND A IS MEDIUM AND SLOC IS MEDIUM AND Effort IS MEDIUM AND Costo IS MEDIU
    RULE 3 : IF R4 IS LOW    AND R1 IS LOW    AND A IS HIGH   AND SLOC IS MEDIUM AND Effort IS HIGH   AND Costo IS MEDIU
    RULE 4 : IF R4 IS LOW    AND R1 IS LOW    AND A IS MEDIUM AND SLOC IS MEDIUM AND Effort IS LOW    AND Costo IS HIGH
    RULE 5 : IF R4 IS MEDIUM AND R1 IS LOW    AND A IS MEDIUM AND SLOC IS MEDIUM AND Effort IS LOW    AND Costo IS MEDIU
    RULE 6 : IF R4 IS MEDIUM AND R1 IS LOW    AND A IS MEDIUM AND SLOC IS LOW    AND Effort IS HIGH   AND Costo IS LOW
    RULE 7 : IF R4 IS MEDIUM AND R1 IS MEDIUM AND A IS HIGH   AND SLOC IS HIGH   AND Effort IS MEDIUM AND Costo IS MEDIU
    RULE 8 : IF R4 IS MEDIUM AND R1 IS MEDIUM AND A IS HIGH   AND SLOC IS MEDIUM AND Effort IS LOW    AND Costo IS MEDIU
    RULE 9 : IF R4 IS LOW    AND R1 IS LOW    AND A IS HIGH   AND SLOC IS LOW    AND Effort IS MEDIUM AND Costo IS MEDIU

END_RULEBLOCK
```

Figura A.2: File «fcl»: definizione fuzzy rule

Per questioni di spazio non si riesce a mostrare completamente tutto lo svolgimento delle fuzzy rule ma solo la prima parte; come si può notare però lo svolgimento rispecchia la definizione espressa con il modello «IF-THEN» tipico delle fuzzy rule.

Una volta terminata la stilazione del file, ora non resta altro che eseguirlo ottenendo come risultato dei grafici che rispecchiano ciò che si è definito per ogni variabile; si prende come esempio l'affidabilità ottenendo il seguente risultato:

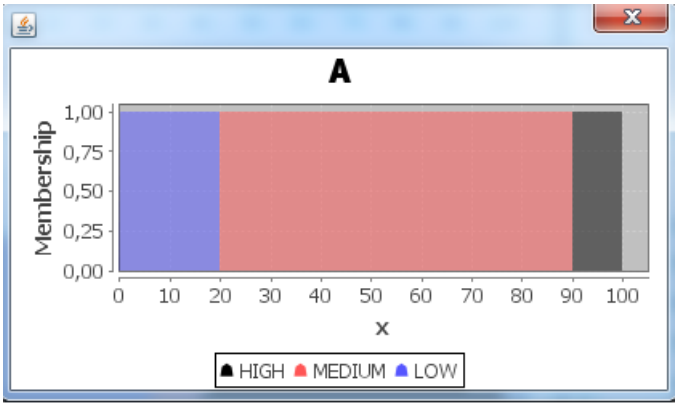

Figura A.3: Affidabilità: grafico FCL

L'esecuzione del programma produrrà un grafico completo per ogni variabile sia in input sia in otput dove ogni colore rappresenta un livello: «LOW», «MEDIUM»,

«HIGH».

Ora che si è definito il file logico, ovvero il file delle regole, non resta altro che definire il programma di lancio della valutazione che prenda in ingresso i valori ottenuti durante lo sviluppo e ne analizzi i risultati.

Il programma sarà implementato in codice Java ed utilizzerà le funzioni presenti nella libreria «JFuzzyLogic.jar», scaricabile on line gratuitamente da questo sito http://sourceforge.net/projects/jfuzzylogic/files/jfuzzylogic/jFuzzyLogic.jar/download.

Ciò che serve a questo punto è quindi un programma molto semplice che prenda in ingresso i valori ottenuti per ogni variabile in input definita e che restituisca in output il livello fuzzy ottenuto, frutto della valutazione delle regole definite nel file FCL.

E' necessario quindi implementare una classe Java che sarà chiamata «ZTotemEvaluation» dove si esamineranno i dati. Questa classe sarà costruita in questo modo:

```
import net.sourceforge.jFuzzyLogic.FIS;
import net.sourceforge.jFuzzyLogic.FunctionBlock;

public class ZTotemEvaluation {


    public static void main(String[] args) {
        // Load from 'FCL' file
        String fileName = "ZTotem.fcl";
        FIS fis = FIS.load(fileName, true);
        // Error while loading?
        if (fis == null) {
            System.err.println("Can't load file: '" + fileName + "'");
            return;
        }


        FunctionBlock functionBlock = fis.getFunctionBlock("ZTotemG1");

        // Set inputs
        functionBlock.setVariable("A", 86);
        functionBlock.setVariable("SLOC", 3);
        functionBlock.setVariable("Effort", 13.45);
        functionBlock.setVariable("Produttivita", 11821.56);
        functionBlock.setVariable("Costo", 7);
        functionBlock.setVariable("R1", 12);
        functionBlock.setVariable("R4", 10);
        functionBlock.setVariable("ScostTime", 0.92);
        functionBlock.setVariable("ScostEffort", 0.90);


        // Evaluate
        functionBlock.evaluate();

        // Show output
        functionBlock.getVariable("G1").defuzzify();

        System.out.println("G1: " +functionBlock.getVariable("G1").getValue());
```

Figura A.4: Java: codice fuzzy logic

prendendo in ingresso il file FCL definito in precedenza, che ha il compito di

inizializzare tutti i valori delle variabili in input con il valore trovato nello sviluppo del prodotto o del prototipo.

Nel momento in cui si chiama la funzione «evaluate()» del function block, si valuteranno i valori in input tramite le fuzzy rule e infine si stamperà a video (sulla console) il valore ottenuto di G1, che nel caso di studio era «VERY GOOD» (valore defuzzificato = 3.5 che equivale a «VERY GOOD» come si può notare nel grafico ottenuto in seguito all'esecuzione del file FCL).

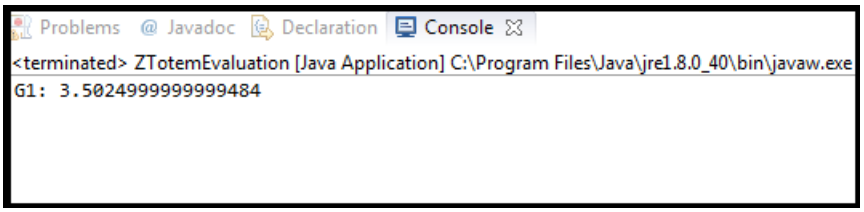

Figura A.5: Risultato valutazione

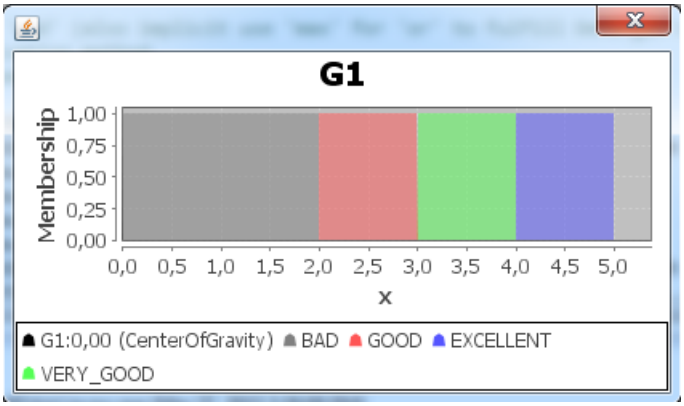

Figura A.6: Risultato valutazione: grafico

Per ogni goal del caso studio da analizzare, del prototipo da sviluppare o del prodotto da tenere in considerazione, è necessario quindi definire un function block nel file «.flc» dove si raccoglieranno tutte le variabili di ingresso, di uscita e le regole fuzzy per ogni goal; dopodichè sarà possibile praticare il meccanismo di valutazione completa attraverso la NFN.

Questo programma è il più semplice programma che possa essere implementato, va da sè che sono possibili molti miglioramenti anche a livello grafico per rendere ancora più user-friendly il contenuto.

E' possibile anche rendere parametrico il numero di input, il numero di output e le regole da implementare in quanto esistono delle funzioni specifiche all'interno della libreria «JFuzzyLogic.jar» che ne permettono l'aggiunta in maniera dinamica.