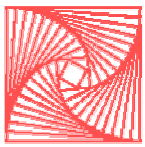


LECIT
CONSULTING

Progettazione per requisiti

White paper

Sommario

Introduzione: quali requisiti, quante architetture?

> *Tieni le tue paure per te, ma dividi la tua ispirazione con gli altri.*
> *Robert Louis Stevenson*

Nello sviluppo di un qualsiasi progetto (dall'architettura di un sistema software complesso alla costruzione di un capanno per gli attrezzi), uno dei compiti iniziali più difficili da svolgere è quello di trovare il bandolo della matassa nella giungla di requisiti che nascono dal primo impatto con il progetto, ma continuano a venire fuori fino all'ultimo giorno e, spesso, anche dopo. Se si aggiunge poi che molti di questi requisiti sono vaghi e che, meglio prima che poi, occorrerà dimostrare a qualcuno che tutti i requisiti sono stati soddisfatti, verrebbe voglia di evitare di infilarsi in questi gineprai. Poiché però i progetti vanno fatti, l'analisi dei requisiti diviene uno dei punti chiave (o forse, **il** punto chiave) di un progetto. Molte metodologie cercano di definire un processo per l'analisi dei requisiti, ma solo di rado questo processo è in grado di mantenere traccia di tutti i requisiti, di mostrare se e come un requisito è soddisfatto dal sistema sviluppato e da quale componenti del sistema, e, infine di mostrare se e come i requisiti che arrivano in corso d'opera influenzano il sistema già parzialmente sviluppato, o se sono completamente incompatibili con quanto fatto finora. Per capirne un po' di più occorre parlare dello **spazio dei requisiti**.

Lo spazio dei requisiti

Un modo integrato per trattare una quantità è quello di definirne le dimensioni. Nel caso dell'analisi dei requisiti, si parlerà di **spazio dei requisiti** come dell'insieme delle dimensioni che descrivono un requisito.

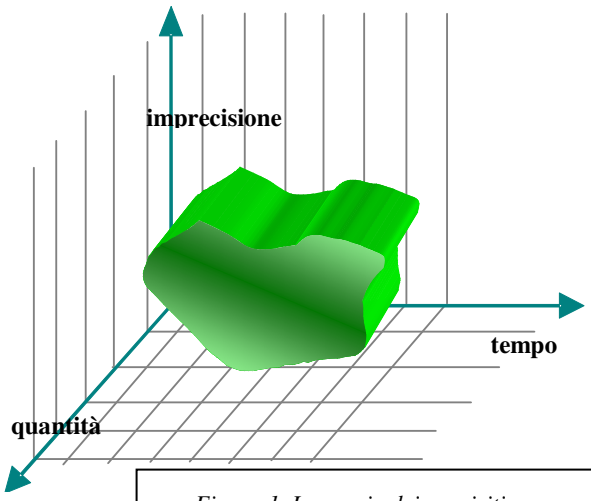


Figura 1. Lo spazio dei requisiti

A prima vista, si potrebbero definire tre dimensioni per lo spazio dei requisiti, come mostrato in Figura 1, dove gli assi rappresentano il tempo (quando i requisiti vengono presentati), il numero dei requisiti e la loro imprecisione.

Agli architetti di sistema piacciono molto spazi dei requisiti ridotti, e per di più molto vicini all'origine degli assi (**pochi** requisiti, molto **precisi** e tutti definiti **all'inizio** del progetto). Sappiamo bene che questo non succede mai. Nel mondo reale, tuttavia, si possono affrontare senza paura anche queste situazioni limite:

1. Molti requisiti vaghi, ma **definiti all'inizio** (e non più cambiati);
2. **Pochissimi** requisiti, anche se vaghi e definiti in momenti diversi;
3. Molti requisiti, definiti in momenti diversi, ma **precisi**.

In altre parole, tutte le volte che si riesce a ridurre lo spazio dei requisiti a due dimensioni, il progetto rimane trattabile, anche se in particolare l'ultimo dei casi elencati può portare a cambiamenti radicali dell'architettura in corso d'opera o anche a situazioni impossibili (requisiti in conflitto). Per esempio, in un ambiente client-server, un requisito per uno

sviluppo rapido (che potrebbe essere stato richiesto all'inizio) potrebbe portare ad una soluzione del tipo "data server", ma un altro requisito di sicurezza sulla rete (che potrebbe arrivare in un secondo momento), potrebbe portare verso un'architettura "function server".

Sfortunatamente, i requisiti non si possono semplicemente enumerare, perché appartengono in generale a categorie completamente diverse fra di loro, e quindi non sono fra loro paragonabili, e per di più debbono essere trattati in fasi diverse del progetto e quindi da persone con ruoli e competenze diversi. E poiché ogni categoria di requisiti aggiunge un'altra dimensione al nostro spazio, il problema della gestione dei requisiti sembra insolubile.

In realtà, il vero problema è quello di posizionare i requisiti in maniera coordinata ed integrata con l'architettura che si sta sviluppando. Ma quante sono le architetture che si sviluppano nel corso di un progetto?

Architettura ed architetture

È ormai un fatto accertato che non è possibile descrivere (e quindi progettare) un sistema completo usando un unico vocabolario di termini e un unico insieme di regole, in altre parole, un'unica architettura. Molti sforzi sono stati fatti per la definizione di un quadro ("framework") architetturale completo: fra questi, uno ha portato alla definizione di uno standard internazionale che codifica, appunto, il numero di architetture sufficienti ed i termini e le regole che sono validi per ciascuna architettura. Questo standard è universalmente noto come ODP (**O**pen **D**istributed **P**rocessing) ed è descritto nel documento ISO 10746-1, Information Technology - Basic Reference Model of Open Distributed Processing - Part 1: Overview. Lo standard definisce cinque modelli architetturali, che, per semplicità[1] possiamo ridurre a quattro:

1. Enterprise/Information, che descrive i domini (i campi di applicazione, le zone di responsabilità, etc.), gli attori (quali entità sono coinvolte nei processi), i ruoli (chi fa cosa quando), i tipi ed i flussi di informazioni;

2. Computational, che descrive gli oggetti computazionali (le entità applicative viste come oggetti), le loro interfacce e le loro interazioni;

3. Engineering, che descrive i meccanismi infrastrutturali a supporto degli oggetti computazionali (le modalità di comunicazione, i protocolli, i servizi transazionali, etc.);

4. Technology, che descrive le caratteristiche dei prodotti usati per costruire il sistema.

Questi modelli corrispondono a linguaggi descrittivi che complessivamente servono a descrivere completamente un sistema, e possono essere considerati come punti di vista[2] diversi della stessa cosa.

Possiamo usare lo stesso approccio di ODP per classificare i requisiti, notando che un singolo requisito, se espresso con linguaggi diversi, può avere effetto (essere "visto") in più di una architettura. Useremo quindi, ai fini della discussione che segue, alcuni termini e concetti del modello ODP: questo **non** significa che l'uso di ODP sia obbligatorio per le tecniche che individueremo. Qualsiasi modello architetturale coerente e completo (che copra cioé **tutti** gli aspetti di un'architettura) può essere usato senza per questo cambiare i risultati di questa analisi.

[1] La semplificazione deriva dal fatto che per due dei cinque modelli architetturali definiti, il linguaggio (termini e regole) è abbastanza semplice ed omogeneo.
[2] Lo standard usa appunto il termine "viewpoint".

I requisiti come vincoli

Se supponiamo di dover progettare (o descrivere) un'architettura in maniera **completamente** libera, possiamo finire in due modi:

1. non progettiamo niente, perché non sappiamo cosa progettare o descrivere

2. progettiamo o descriviamo la "cosa che fa tutto".

Nell'infinita gamma di possibilità fra il non fare niente e fare tutto, i requisiti possono essere visti come la via che ci porta a fare qualcosa di utile.

In altre parole, supponendo di avere a disposizione gli strumenti per descrivere il tutto, rappresentati nel nostro caso dai linguaggi e dai termini ODP, **i requisiti servono ad eliminare ciò che non serve**[3].

Se infine riusciamo a posizionare i requisiti ciascuno nel punto di vista, o nei punti di vista, che gli compete, otteniamo la quadratura del cerchio: ogni requisito ci suggerisce cosa usare o cosa non usare per descrivere la mia architettura da ognuno dei punti di vista necessari per tenere tutto sotto controllo.

Un esempio

Supponiamo di dover progettare un sistema applicativo, e che uno dei tanti requisiti suoni più o meno così:

> *l'applicazione sarà accessibile tramite terminali: non è prevista una interazione con altri mezzi (es. voce)*

e che un altro requisito suoni così:

> *l'applicazione tratterà dati e testi in forma elaborabile: non sono previsti oggetti multimediali (suono, immagini, etc.)*

Usando come detto in precedenza il modello ODP per disegnare l'architettura del sistema, nel corso del progetto dovremo descrivere l'architettura **computazionale**, dove identificheremo le componenti dell'applicazione in termini di oggetti che interagiscono fra loro tramite interfacce. Le interfacce, secondo lo standard ODP possono essere sostanzialmente di due tipi: operazionali e stream.

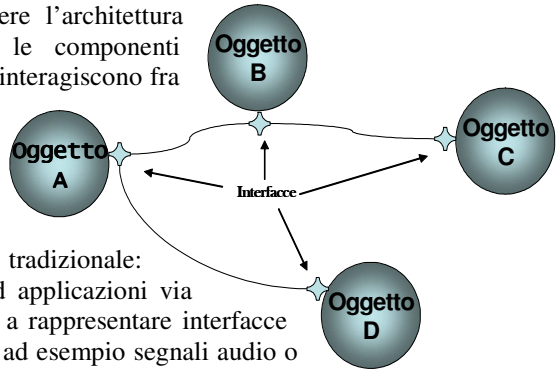


Figura 2. Oggetti, interfacce ed interazioni

Il primo tipo di interfacce è quello tradizionale: interazioni fra processi o fra operatori ed applicazioni via stringa di comandi. Il secondo tipo serve a rappresentare interfacce multimediali (flussi continui di dati, come ad esempio segnali audio o video).

La vista computazionale del sistema potrà

[3] Si pensi all'approccio di Michelangiolo con la scultura. Dato un informe blocco di marmo, Michelangelo sosteneva che la scultura era già all'interno, ed il suo compito era semplicemente quello di eliminare le parti che non servivano.

essere grossolanamente rappresentabile come un insieme di cerchi (oggetti), archi (interazioni) e connettori (interfacce fra oggetti), come in Figura 2.

Nella figura, ogni oggetto rappresenta un componente dell'applicazione.

I due requisiti che abbiamo visto **impongono l'uso di interfacce operazionali**. Questo significa che, di tutti i tipi di interazioni possibili (quelle di tipo stream sono, ovviamente, di tanti tipi diversi), **solo uno** sarà presente nel sistema applicativo in esame.

Abbiamo così usato dei requisiti per restringere l'insieme delle possibili scelte architetturali.

La progettazione per requisiti

Possiamo pensare ad uno strumento che permetta di classificare e tenere sotto controllo contemporaneamente i requisiti nel corso della progettazione?

Se consideriamo la Tabella 1 che segue, vediamo che probabilmente abbiamo uno strumento utile.

Requisiti	Enterprise/Information	Computational	Engineering	Technology
Requisito 1	Occorre saper usare terminali	Interfacce operazionali		
Requisito 2		Interfacce operazionali		
Requisito 3			Xxxxx	Yyyyyy
------	-----	-----	----	----
Requisito n	-----	-----	-----	------

Tabella 1 La matrice dei requisiti

Nella tabella abbiamo definito una prima colonna con tutti i requisiti (sono stati usati i due requisiti già espressi nell'esempio), ed altre quattro colonne che rappresentano le quattro viste del sistema[4]. Per ogni requisito, abbiamo poi valutato l'eventuale impatto per ogni diverso punto di vista.

Notiamo per inciso che il primo requisito, di cui nell'esempio avevamo considerato la sola vista computazionale, ha un impatto anche dal punto di vista Enterprise: di fatto, il requisito implica che solo persone in grado di usare terminali potranno accedere ai servizi dell'applicazione. Questo ha, ovviamente, un certo impatto sul modello organizzativo e sui flussi di informazioni (chi è abilitato a fare cosa e quando). Ritorneremo su questo punto più avanti.

La rappresentazione compatta che si è ottenuto permette, già dopo la prima analisi dei requisiti, di produrre tutte le rappresentazioni architetturali (in termini di disegni, diagrammi e quant'altro) che servono, anche se, ovviamente, in forma ancora incompleta. Ad esempio, possiamo immaginare una prima architettura Enterprise in cui sono già definiti ruoli e domini, come nel diagramma in Figura 3. Nella figura, il solo requisito già più volte considerato ha portato ad identificare due classi di utenti: quelli che sanno come usare l'applicazione e quelli che non lo sanno. Se vogliamo permettere a tutti l'uso dell'applicazione (questo non è espresso dal requisito esaminato, ma possiamo supporre che sia un altro requisito, casomai pervenuto in un secondo momento), allora dovremo inventare la figura ed il ruolo di "utente intermedio", cioé il classico operatore di

[4] Ancora una volta, l'esempio è sviluppato seguendo concetti ODP. Un diverso modello architetturale prevederà altre viste: il risultato non cambia.

sportello. Abbiamo potuto anche identificare alcuni flussi di informazioni, identificati nella figura da frecce. Infine, abbiamo identificato due diversi ***domini***, perché possiamo ad esempio immaginare che un utente finale abbia certi diritti di accesso alle informazioni, mentre l'utente intermedio ne abbia diversi. Questo fatto implica una scelta di politiche di sicurezza, di controllo degli accessi, etc. In sostanza, abbiamo già una rappresentazione a livello macroscopico del sistema, che può essere usata per, ad esempio, verificare le scelte fatte con il committente.

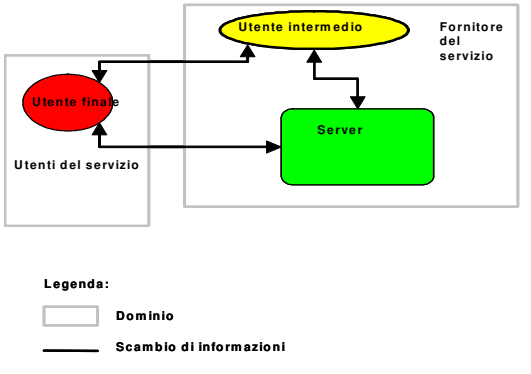


Figura 3. Un'architettura Enterprise/Information

Le rappresentazioni definitive per tutti i modelli architetturali si avranno a fine progetto.

Da quanto detto finora, discende direttamente la tecnica per il controllo dell'avanzamento del progetto. Possiamo verificare, per ogni modello architetturale e per ogni requisito, o scelta tecnica fatta, che i risultati soddisfano effettivamente le richieste. Notiamo per inciso che ogni scelta tecnica che viene fatta nel corso del progetto potrà essere trattata alla stessa stregua di un requisito. E allora, abbiamo anche la strada aperta per il problema dei "requisiti tardivi", cioé quelli che arrivano ad un certo momento, a progetto in corso. Sarà infatti sempre possibile, mantenendo aggiornati i diagrammi architetturali e la matrice dei requisiti e delle scelte, verificare gli impatti che un nuovo requisito può avere sul sistema, **perché il sistema, seppure incompleto, sarà sempre disponibile**.

Lecit Consulting

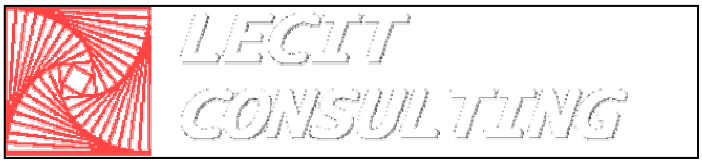
LECIT Consulting è un'azienda indipendente di consulenza di Information Technology, specializzata nella valutazione e nella progettazione delle architetture dei sistemi IT d'impresa.

Fornire consulenza indipendente non è facile. Il risultato di una consulenza spesso – soprattutto quando è fornita da una grande azienda – è quello di affidare la realizzazione del sistema software a fornitori che sono partner del consulente. In questi casi, quale obbiettività ha avuto il consulente tecnico?

Una consulenza professionale ed oggettiva è estremamente importante per il Cliente: IT manager ed imprenditori sono talvolta influenzati dal marketing del fornitore senza sapere esattamente se ciò che stanno comprando sia la migliore soluzione possibile.

Invece, oggettività ed indipendenza sono i nostri valori chiave. Questo è dimostrato dalle nostre storie professionali: anni di esperienza nella consulenza e nella progettazione, realizzazione e conduzione di progetti per clienti sia di grandi che di medie imprese che per la Pubblica Amministrazione, in settori chiave come (la lista non è esaustiva): e-commerce, CRM (Customer Relationship Management), sicurezza informatica, gestione di contenuti per siti web complessi, architetture distribuite, sistemi di pagamento.

Tutti i nostri interventi sono sempre stati attuati nell'ottica di creare valore per i nostri Clienti, senza essere condizionati dal marketing dei fornitori di tecnologie e di prodotti.



Via S. Maria 25
56100 Pisa
Italia

Tel: 3357122730
335383559
www.lecitconsulting.it
Email: info@lecitconsulting.it