SISTEMI SOCIO-TECNICI (cap. 2)

Un sistema è una collezione significativa di componenti interrelati che lavorano assieme per realizzare un determinato obiettivo. Questo può includere software, componenti meccanici, elettronici, ecc.

I sistemi che includono software sono organizzati in due categorie:

1. Sistemi tecnico-informatici: comprendono componenti hardware e software, ma non procedure o processi (TV, cellulari, ma anche elaboratori di testi, ecc.);
2. Sistemi socio-tecnici: includono uno o più sistemi tecnici, e le informazioni sull’utilizzo del sistema per raggiungere un obiettivo più vasto. Tali sistemi quindi hanno alcuni processi operativi ben definiti, comprendono persone (gli operatori) come parte integrante del sistema, sono governati da politiche e regole.

Le caratteristiche essenziali dei sistemi socio-tecnici sono tre:

1. Proprietà complessive: queste sono del sistema interno, non associate alle singole parti, ma dipendono sia dai componenti del sistema, sia dalla loro relazione;
2. Non-deterministici: quando viene data loro un determinato input, l’output potrebbe non essere sempre lo stesso; questo poiché dipende da operatori, cioè esseri umani.
3. Obiettivi: l’aiuto del sistema per il raggiungimento di tali obiettivi dipende non solo dal sistema stesso, ma anche dalla stabilità di tali obiettivi, dai loro conflitti e relazioni e da come le persone li interpretano. Quindi una nuova gestione potrebbe reinterpretare tali obiettivi, causando un fallimento del sistema.

(VEDI)

2.1 Proprietà complessive di un sistema.

Le proprietà complessive non possono essere attribuite a nessuna parte specifica del sistema, ma emergono solo quando i componenti del sistema sono integrati. Alcune di queste possono essere derivate direttamente da simili proprietà dei sottosistema, ma più spesso risultano da complesse interrelazione tra i sotto-sistemi. (VEDI e SCRIVI fig. 2.1)

Ci sono 2 tipi di proprietà complessive.

1. Proprietà complessive funzionali: appaiono quando tutte le parti di un sistema lavorano assieme per raggiungere un obiettivo.
2. Proprietà complessive non funzionali: si riferiscono al comportamento del sistema nel suo ambiente operativo. Esempi sono l’affidabilità, le prestazioni, la sicurezza e la protezione, che sono spesso critiche per i sistemi informatici, poiché bisogna raggiungere un numero minimo di tali proprietà per rende un sistema perlomeno utilizzabile.

L’affidabilità di un sistema deve essere sempre considerato rispetto all’intero sistema e non ai singoli componenti, poiché sono interdipendenti e il fallimento di uno può propagarsi attraverso il sistema e modificare il comportamento degli altri. È difficile prevedere come questo accade, quindi non è possibile fare stime accurate dell’affidabilità generale del sistema in base ai dati di affidabilità dei suoi componenti.

Sull’affidabilità generale del sistema ci sono tre influenze correlate:

1. Affidabilità hardware: probabilità di rottura di un componente hardware e tempo di riparazione;
2. Affidabilità software: probabilità output errato. Questi problemi sono temporanei, quindi il sistema continua a lavorare anche dopo un risultato sbagliato;
3. Affidabilità dell’operatore: probabilità che l’operatore del sistema commetta uno sbaglio.

Tutte queste proprietà sono strettamente collegate; un piccolo errore iniziale, quindi, può sviluppare velocemente problemi più gravi, che richiedono un fermo totale del sistema.

Come l’affidabilità, anche le altre proprietà complessive (prestazione, usabilità, ecc.) sono difficili da prevedere, e possono essere valutate solo quando il sistema è operativo. Proprietà come la sicurezza e la protezione, però, pongono problemi diversi: non si parla più di un comportamento generale del sistema, ma di uno che non dovrebbe tenere; un sistema sicuro è quello che impedisce l’accesso non autorizzato ai suoi dati: ciò è impossibile, quindi queste proprietà sono effettive solo su carta. Si può sapere che un sistema non è sicuro solamente quando qualcuno riesce a entrarvi.

2.2 Ingegneria dei sistemi.

L’ingegneria dei sistemi è l’attività di specifica, progettazione, implementazione, convalida, impostazione e mantenimento dei sistemi socio-tecnici. Gli ingegneri di sistemi si occupano non solo di software, ma anche di hardware e dell’interazione del sistema con gli utenti e con il loro ambiente; devono pensare ai servizi che il sistema offre, ai vincoli a cui deve sottostare durante la sua costruzione e il suo uso, e ai modi in cui il sistema viene usato per raggiungere il suo scopo.

Le fasi del processo di ingegneria dei sistemi sono mostrare in fig. 2.2: questo processo ha un’influenza importante sul modello “a cascata” della lavorazione software.

Ci sono delle distinzioni fondamentali tra il processo di ingegneria dei sistemi e il processo di sviluppo del software:

1. Spazio limitato per modifiche durante lo sviluppo del sistema: alcune decisioni potrebbero non essere modificabili; uno dei motivi che ha reso il software così importante è appunto la possibilità di apportare cambiamenti durante la fase a sviluppo, in risposta a nuovi requisiti;
2. Coinvolgimento interdisciplinare: nell’ingegneria dei sistemi possono essere coinvolte molte discipline ingegneristiche, e quindi possono crearsi incomprensioni a causa delle diverse terminologie e convenzioni usate dai vari ingegneri.

L’ingegneria dei sistemi è un’attività interdisciplinare che coinvolge squadre provenienti da mondi diversi. Quest’ultime sono necessarie a causa della vasta conoscenza richiesta per valutare tutte le conseguenze delle decisioni di progettazione del sistema. (fig. 2.3)

Per molti sistemi, ci sono possibilità infinite di scambi tra diversi tipi di sottosistemi. Non c’è una direzione giusta su come un sistema deve essere scomposto, ma esistono solo alternative, e non è sempre possibile scegliere la soluzione tecnica migliore.

2.2.1 Definizione dei requisiti del sistema.

La definizione dei requisiti di sistema specifica cosa il sistema dovrebbe fare (le sue funzioni) e le proprietà essenziali desiderate. Ci sono tre tipi di requisiti:

1. Requisiti funzionali astratti: sono le funzionalità basilari; le specifiche più dettagliate avvengono a livello dei sottosistemi;
2. Proprietà del sistema: proprietà non funzionali complessive del sistema, come disponibilità, prestazioni, sicurezza. Interessano i requisiti di tutti i sottosistemi;
3. Caratteristiche che il sistema non deve fare: bisogna indicare cosa il sistema non deve fare.

Una parte importante della fase di definizione dei requisiti è stabilire l’insieme degli obiettivi generali che il sistema deve raggiungere. Questi potrebbero non essere espressi in termini di funzionalità del sistema, ma potrebbero piuttosto definire perché il sistema viene preparato per un particolare ambiente.

(VEDI SLIDE)

2.2.2 Progettazione del sistema.

Il processo di progettazione del sistema definisce in quale modo i componenti devono fornire le funzionalità del sistema (fig. 2.4):

1. Suddivisione dei requisiti: si analizzano i requisiti e si organizzano in gruppi collegati. Ci sono diverse possibili opzioni di suddivisione e si possono quindi suggerire alcune alternative;
2. Identificazione dei sottosistemi: si identificano i sottosistemi che possono soddisfare, individualmente o collettivamente, i requisiti;
3. Assegnazione dei requisiti: in teoria dovrebbe essere molto semplice, per via delle fasi precedenti, ma in pratica non c’è mai riscontro tra riscontri e sottosistemi. Le limitazioni di sottosistemi acquistati da esterni possono richiedere un cambiamento dei requisiti per soddisfare questi vincoli;
4. Specifica delle funzionalità del sottosistema: si dovrebbe anche provare a identificare le relazioni tra i vari sottosistemi;
5. Definizione delle interfacce dei sottosistemi.

I processi di ingegneria dei requisiti e di progettazione sono assolutamente collegati tra loro: i vincoli dati da sistemi già esistenti possono limitare le scelte di progettazione, e queste scelte possono essere specificate nei requisiti. Potrebbe essere necessario fare una progettazione iniziale per strutturare e organizzare il processo di ingegneria dei requisiti. Mentre la progettazione procede, si potrebbero incontrare problemi con requisiti esistenti, e potrebbero comparire nuovi requisiti. (fig. 2.5)

Il processo a spirale riflette la realtà: i requisiti condizionano le decisioni di progettazione e viceversa.

Per quasi tutti i sistemi ci sono molti modi di progettarli in modo da soddisfare i requisiti richiesti, coprendo una gamma di soluzioni che combinano hardware, software e operazioni umane. (VEDI)

2.2.3 Modellazione del sistema.

Durante l’attività di definizione dei requisiti e di progettazione, i sistemi possono essere modellati come un insieme di componenti e di relazioni tra questi, illustrati graficamente in un modello di architettura del sistema.

Quest’ultima può essere presentata come un diagramma a blocchi che mostra i sottosistemi principali e le interconnessioni tra questi: i sottosistemi sono rappresentati da un triangolo, e le relazioni tra questi con delle frecce (fig. 2.6). il diagramma a blocchi dovrebbe essere accompagnato da un breve descrizione di ogni sottosistema (fig. 2.7).

A questo livello, il sistema è suddiviso in sottosistemi che interagiscono (VEDI)

In passato, il modello architetturale del sistema era usato per identificare i componenti hardware e software che potevano essere sviluppati in parallelo, ma oggi questa distinzione sta scomparendo, e ormai tutti i componenti ora comprendono capacità di calcolo integrate.

A livello architetturale, è più appropriato classificare i sottosistemi secondo le loro funzioni prima di prendere decisioni sugli scambi hardware/software: la decisione di fornire una funziona via hardware o software dipende da alcuni fattori (disponibilità componenti, tempo di sviluppo, ecc.).

I diagrammi a blocchi possono essere usati per i sistemi di tutte le dimensioni: in quelli grandi (fig. 2.8) i sottosistemi sono dei sistemi, e le frecce che li collegano mostrano il flusso di informazioni.

2.2.4 Sviluppo dei sottosistemi.

Durante lo sviluppo, i sottosistemi identificati vengono implementati: questo richiede l’avvio di un nuovo processo di ingegneria del sistema per i singoli sottosistemi o, se si tratta di software, di una lavorazione software che coinvolge la creazione di requisiti, la progettazione, l’implementazione e la prova.

Alcuni sottosistemi sono creati da zero nella fase di sviluppo, mentre altri sono COTS (commercial off-the-shelf, prodotto commerciale standard), comprati per essere integrati nel sistema finale, poiché più economico. In questo caso, bisognerà tornare all’attività di progettazione per adattare un componente acquistato.

I sottosistemi sono generalmente sviluppati in parallelo, e quando si incontrano problemi che superano i confini di un sottosistema, bisogna modificare l’intero sistema; se però richiede una complessa ingegneria hardware, fare modiche è molto costoso. Per risolvere questi problemi, si usano dei work-around, che implicano cambiamenti al software, il quale deve essere progettato in modo tale da poterlo modificare per implementare nuovi requisiti, senza costi addizionali eccessivi.

2.2.5 Integrazione dei sistemi.

L’integrazione è la fase in cui i sottosistemi vengono uniti per formare un sistema completo. Questa integrazione può essere fatta usando un approccio “big bang”, dove tutti i sottosistemi sono integrati allo stesso tempo, ma la soluzione migliore è l’integrazione incrementale, per 2 motivi.

1. È impossibile che gli sviluppi di tutti i sottosistemi vengano completati nello stesso momento;
2. Questo tipo di integrazione riduce i costi dovuti alla ricerca di errore, poiché, in quello simultaneo, l’errore potrebbe essere in uno qualsiasi di questi sottosistemi, mentre in quello incrementale l’errore risulterà nel nuovo sistema, o nell’interazione tra questo e il precedente.

Poi si passa al test totale del sistema.

Durante la fase di integrazione vengono a galla gli errori dei sottosistemi causati da asserzioni sbagliate su altri sottosistemi: i vari appaltatori discutono su quale sottosistema è errato, e la risoluzione di tale problema può richiedere molto tempo.

Con l’aumentare dei sistemi, costituiti integrando componenti COTS hardware e software, il processo di integrazione sta diventando sempre più importante, tanto che i sottosistemi, in alcuni casi, non vengono sviluppati separatamente e l’integrazione coincide con l’implementazione del sistema.

(INSTALLAZIONE DEL SISTEMA)

2.2.6 Evoluzione del sistema.

L’evoluzione del sistema, come quello del software, è costosa per diverse ragioni:

1. I cambiamenti proposti devono essere analizzati con molta cura da una prospettiva tecnica e aziendale, devono contribuire allo scopo del sistema, e non devono avere solo motivi tecnici;
2. Poiché i sottosistemi non sono mai completamente indipendenti, il cambiamento di uno influenza negativamente il comportamento/prestazioni di altri sottosistemi, con conseguenti ulteriori cambiamenti;
3. I motivi delle decisioni prese durante la progettazione iniziale sono spesso sconosciuti, e i responsabili dell’evoluzione del sistema devono tentare di capirli;
4. Con l’invecchiamento del programma, la sua struttura solitamente si corrompe a causa dei cambiamenti, tanto da fare aumentare il costo di ulteriori modifiche.

I sistemi che si sono evoluti nel tempo spesso fanno affidamento su tecnologie hardware/software vecchie, ma hanno un ruolo chiave in un’azienda (sistemi ereditati, legacy systems): sistemi che l’azienda vorrebbe sostituire, se i rischi di una loro sostituzione non fossero troppo alti.

2.2.7 Rimozione del sistema.

Per i sistemi hardware può richiedere lo smontaggio e il riciclaggio dei materiali / trattamento di sostanze tossiche. Il software non ha problemi di rimozione fisica, ma alcuni possono essere utilizzati per agevolare il processo di rimozione (es. tenere sotto controllo i componenti hardware). Quando un sistemi è rimosso, i componenti che non sono rovinati possono essere riutilizzati in altri sistemi.

Se i dati nel sistema da rimuovere sono ancora importanti per l’azienda, e quindi è necessario convertirli per poterli riutilizzare: per esempio, le strutture dati potrebbero essere definite all’interno del software stesso che, quindi, deve essere analizzato per scoprire come sono strutturati i dati, per riorganizzarli in una struttura compatibile con il nuovo sistema.

2.3 Organizzazioni, persone e sistemi informatici.