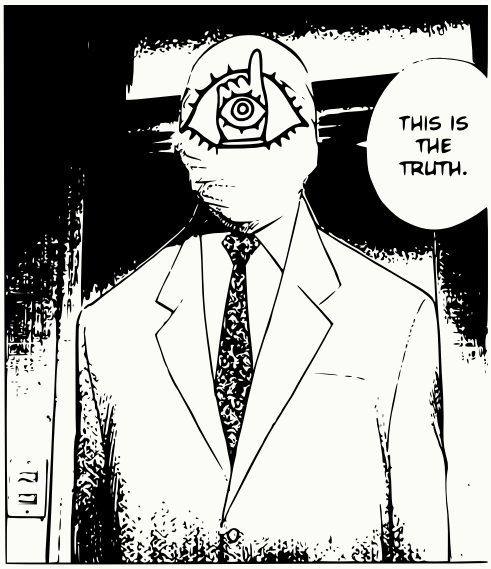
Marco Casu

Ingegneria del Software

SAPIENZA
UNIVERSITÀ DI ROMA

Facoltà di Ingegneria dell'Informazione, Informatica e Statistica
Dipartimento di Informatica

Questo documento è distribuito sotto la licenza GNU, è un resoconto degli appunti (eventualmente integrati con libri di testo) tratti dalle lezioni del corso di Ingegneria del Software per la laurea triennale in Informatica. Se dovessi notare errori, ti prego di segnalarmeli.



INDICE

CAPITOLO

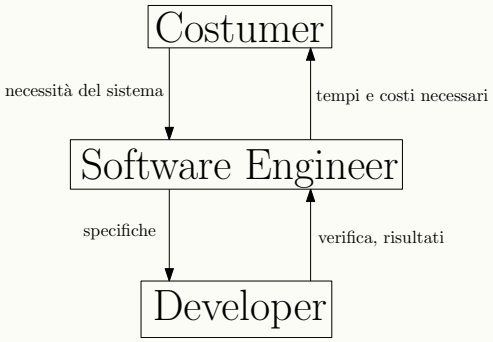
1 INTRODUZIONE

Quando si vuole produrre software complesso, non terminante e di grandi dimensioni, è necessario ingegnerizzare il metodo di sviluppo ed astrarre il modello del software. L'obiettivo dell'ingegneria del software non riguarda il codice stesso, ma la definizione dell'architettura e delle specifiche del sistema. Vanno definiti i processi di sviluppo.

È cruciale l'analisi dei requisiti, e la modellizzazione delle specifiche di sistema, UML è un linguaggio che permette di descrivere la *dinamica* ed il comportamento del modello. Una volta definiti i requisiti, si affronta la fase di *planning*.

Un modello computazionale che risulterà utile prende il nome di *digital twin*, ossia una copia del software sulla quale è possibile testare i vari *scenari operativi*, tramite la generazione di test automatici. Essendo il sistema discreto, a stati non necessariamente finiti, è possibile descriverlo tramite una catena di Markov.

Nella produzione software prendono parte 3 principali attori



1.1 Modellazione

Il linguaggio UML è utilizzato per modellare sia i requisiti che il sistema stesso, esso è composto da differenti diagrammi

- activity
- use case
- sequence

- class
- state
- context

Si consideri il seguente esempio di un sistema di gestione di una clinica psichiatrica, lo schema in figura 1.1, è il context diagram, e rappresenta l'insieme dei serivizi che il sistema offre. Un activity diagram descrive

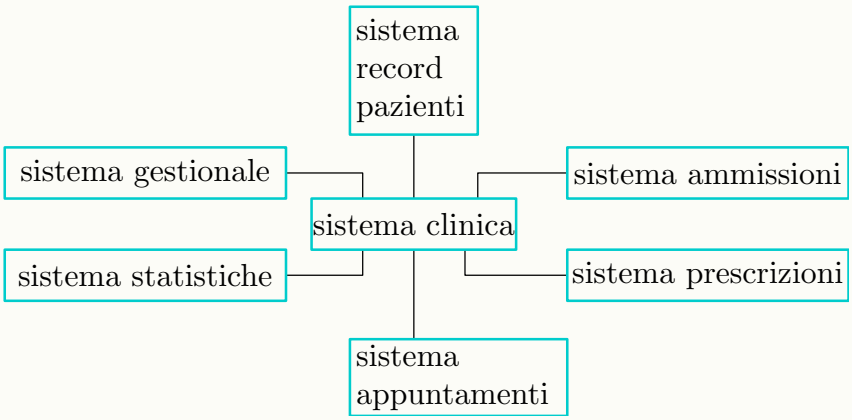


Figura 1.1: context diagram

l'evoluzione di una certa attività/task che il sistema deve poter implementare, si considere l'esempio in figura 1.2 riguardante l'inserimento di un paziente nella clinica.

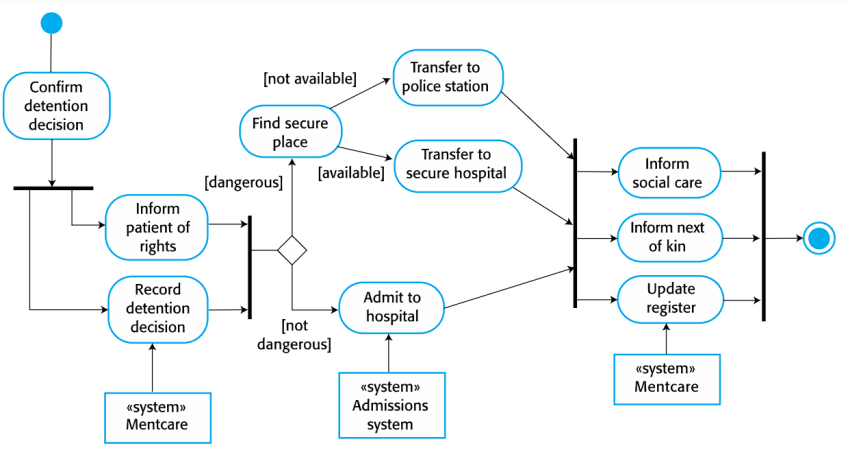


Figura 1.2: activity diagram

1.2 Catene di Markov per la Progettazione

La moderna progettazione di sistemi complessi si basa sul concetto di digital twin, in modo da poter definire un modello che simula il sistema su cui verranno eseguiti determinati esperimenti al fine di collaudo prima della progettazione vera e propria. Questo capitolo si concetra sull'aspetto *predittivo* dei digital twin : Si vuole utilizzare il modello per fare predizioni sull'andamento della progettazione.

Definizione (DTMC) : Una **Catena di Markov Discreta** è una tupla (U, X, Y, p, g) tale che

- U è un insieme (possibilmente vuoto, finito o non) denotato *input/ingresso* del sistema.
- X è un insieme non vuoto (finito o non) denotato *stato* del sistema.

- Y è un insieme non vuoto (finito o non) denotato *output/uscita* del sistema.
- p è la *probabilità di transizione*, è una funzione

  $$p : X \times X \times U \to [0, 1]$$

  tale che $p(x'|x, u)$ rappresenta la probabilità che la DTMC si sposti dallo stato x allo stato x' quando l'ingresso è u. Inoltre, per ogni $\tilde{x} \in X$ e per ogni $\tilde{u} \in U$, si ha che

  $$\int_{x' \in X} p(x'|\tilde{x}, \tilde{u}) = 1$$

  informalmente, per ogni possibile coppia stato-ingresso, esiste uno stato successivo.
- g è una funzione

  $$g : X \to Y$$

  detta *funzione di output*.

Data una catena di markov M, denotiamo $M(U), M(X), M(Y)$ rispettivamente : gli ingressi, gli stati e le uscite.

Esempio tempi di completamento : Si consideri il processo di sviluppo mostrato in figura 1.3, si vuole calcolare la probabilità che il processo termini in esattamente 10 mesi.

Si identificano i possibili cammini la cui somma dei sotto processi impiega 10 mesi

$$0 \to 1 \to 1 \to 2 \to 3 \to 4$$
$$0 \to 1 \to 2 \to 3 \to 3 \to 4$$

La probabilità che ciò accada è la somma delle probabilità sui due percorsi

$$\begin{matrix} 1 \cdot 0.3 \cdot 0.7 \cdot 0.6 \cdot 0.6 = 0.0756 \\ 1 \cdot 0.7 \cdot 0.6 \cdot 0.1 \cdot 0.6 = 0.02520 \end{matrix} \implies p(\text{finire in 10 mesi}) = 0.1008$$

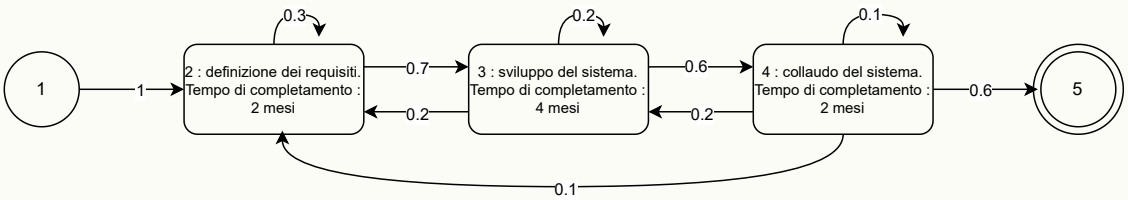


Figura 1.3: Esempio 1

Qual'è invece la probabilità di finire il processo in esattamente 12 mesi? Si identificano i cammini

$$0 \to 1 \to 1 \to 1 \to 2 \to 3 \to 4$$
$$0 \to 1 \to 1 \to 2 \to 3 \to 3 \to 4$$
$$0 \to 1 \to 2 \to 2 \to 3 \to 4$$
$$0 \to 1 \to 2 \to 3 \to 3 \to 3 \to 4$$

la probabilità è

$$1 \cdot 0.3 \cdot 0.3 \cdot 0.7 \cdot 0.6 \cdot 0.6 +$$
$$1 \cdot 0.3 \cdot 0.7 \cdot 0.6 \cdot 0.1 \cdot 0.6 +$$
$$1 \cdot 0.7 \cdot 0.2 \cdot 0.6 \cdot 0.6 +$$
$$1 \cdot 0.7 \cdot 0.6 \cdot 0.1 \cdot 0.1 \cdot 0.6 =$$
$$0.022680 + 0.007560 \cdot 0.05040 \cdot 0.002520 = 0.08316$$

Esempio costi di completamento : Si consideri il processo di sviluppo mostrato in figura 1.4, si vuole calcolare la probabilità che il processo abbia un costo toale di esattamente 80.000 euro.

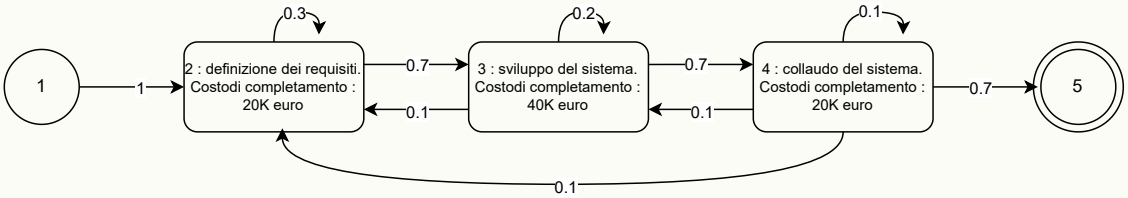


Figura 1.4: Esempio 2

L'unico cammino possibile è

$$0 \to 1 \to 2 \to 3 \to 4$$

La probabilità è

$$1 \cdot 0.7 \cdot 0.7 \cdot 0.7 = 0.343$$

Esempio costi di design : Si consideri il processo di sviluppo mostrato in figura 1.5, quale deve essere il minimo valore di p tale che la probabilità che il processo termini in esattamente 8 mesi sia almeno 0.5?

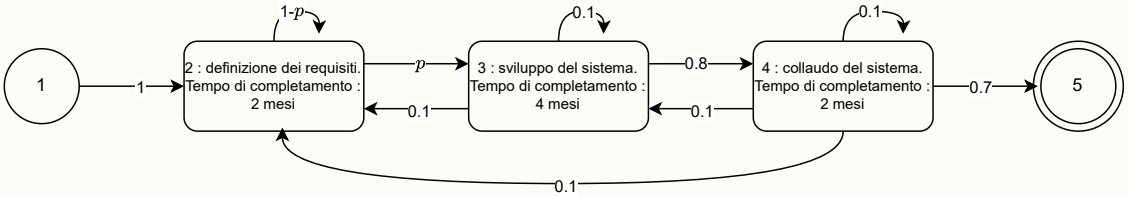


Figura 1.5: Esempio 3

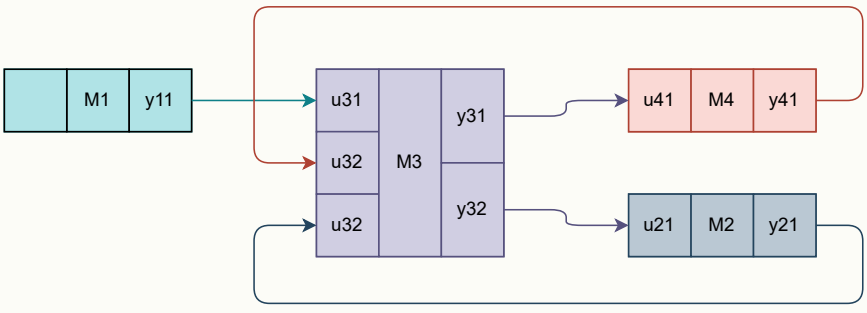
L'unico cammino possibile è

$$0 \to 1 \to 2 \to 3 \to 4$$

la cui probabilità deve essere

$$1 \cdot p \cdot 0.8 \cdot 0.7 \geq 0.5 \implies$$
$$p \geq \frac{0.5}{0.8 \cdot 0.7} = 0.8928$$

Una **rete di catene di Markov** è un insieme di catene di Markov interconnesse tramite gli ingressi e le uscite.



Inoltre, una rete di catene di Markov è rappresentabile come una catena di Markov singola, si considerino le catene di Markov K, P in figura 1.6, formalmente

$$K = (U_1 \times V, X_1, Y_1, p_1, g_1)$$
$$P = (U_2, X_2, W \times Z, p_2, (g_{2_w}, g_{2_z}))$$

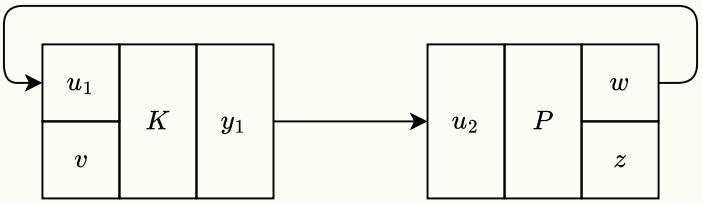


Figura 1.6: K e P interconnesse

Nota : u_1 rappresenta un elemento dell'insieme U_1 (ingresso di K), $v \in V$, $y_1 \in Y_1 \ldots$ analogo per gli altri elementi.

Un ingresso di K è un uscita di P

$$M_1(U_1) = M_2(W)$$

e l'ingresso di P è l'uscita di K

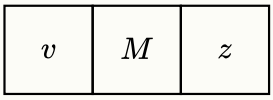
$$M_2(U_2) = M_1(Y_1)$$

Si può definire il sistema globale equivalente

$$M = (V, X_1 \times X_2, Z, p, g)$$

Dove

- $p((x_1', x_2')|(x_1, x_2), v) = p_1(x_1'|x_1, (g_{2_w}(x_2), v)) \cdot p_2(x_2'|x_2, g_1(x_1))$
- $g(x_1, x_2) = g_{2_z}(x_2)$



CAPITOLO

2

PLANNING DEL PROGETTO

Il planning di un progetto, consiste nel suddividere il carico di lavoro in diverse parti, da assegnare ai vari membri del team, cercando di prevedere possibili problemi che potrebbero insorgere durante lo sviluppo, pensando e preparando eventuali modi per risolverli.

Il *piano del progetto* viene preparato all'inizio dei lavori, viene utilizzato per comunicare ai vari membri del team ed al cliente come esso è stato suddiviso. Le fasi in cui il planning viene definito sono

- Durante la proposta, quando avviene la contrattazione con il cliente riguardo lo sviluppo del software
- Durante la fase di avvio dei lavori, quando si decide come e a chi assegnare il lavoro, e quali risorse dovranno essere allocate
- Periodicamente durante lo sviluppo, monitorando ed appositamente modificando i piani in base all'andamento del progetto e alle esperienze pregresse

Lo scopo del planning è quello di avere un idea chiara sul progetto in modo che si possa decidere un prezzo d'accordo con il cliente, valutando e stimando quanto denaro sarà necessario per lo sviluppo considerando variabili del tipo

- costo dei dipendenti
- costo dell'hardware necessario
- costo del software

Durante la fase di planning, sono noti i requisiti del sistema, ma non è ancora chiara la struttura del software, tanto meno la sua implementazione, il planning deve essere preciso a sufficienza per far si che sia possibile definire il budget e lo staff necessario. Inoltre bisogna definire i meccanismi con la quale verrà monitorato lo sviluppo.

2.1 Sviluppo Plan-Driven

Con *plan driven development*, si intende un approccio all'ingegneria del software in cui i processi di sviluppo sono programmati in principio, in maniera dettagliata. Si basa sulle tecniche di gestione, tipiche dei progetti ingegneristici, e sulle tecniche "classiche" di gestione di grandi progetti software.

Un project plan ha l'obiettivo di definire e monitorare il lavoro da svolgere, in che modo deve essere

svolto, da chi, e quali prodotti sono necessari. È scopo del manager, servirsi di un project plan (che da ora chiameremo semplicemente "piano") per supportare le decisioni da prendere durante lo sviluppo, e per misurarne il progresso.

- Un lato favorevole di tale approccio, è che una pianificazione a monte permette di aggirare problemi organizzativi in principio, determinando potenziali problemi e dipendenze da soddisfare prima che il progetto sia avviato, piuttosto che durante la lavorazione.

  meglio prevenire che curare

- Un lato sfavorevole, è che molte decisioni prese in principio vanno riviste dati possibili cambiamenti dell'ambiente in cui il software deve essere adoperato.

Un piano deve definire, le risorse disponibili per il progetto, la suddivisione del carico di lavoro, una schedule per portare a termine il lavoro. Precisamente, un piano consiste nelle seguenti sezioni

1. introduzione
2. organizzazione del progetto
3. analisi dei rischi
4. risorse hardware e software necessarie
5. struttura di scomposizione del lavoro
6. schedule del progetto
7. meccanismi di monitoraggio e report

Ci sono inoltre altri tipi di "piano" che possono essere aggiunti a quello principale come supporto:

Piano	Descrizione
Configuration managment plan	descrive la configurazione delle procedure di gestione, la loro struttura ed il loro utilizzo
Deployment plan	descrive come il software deve essere integrato con l'apposito hardware di riferimento del cliente, con eventuali piani di migrazione dei dati in uso su sistemi precedentemente adoperati
Maintenance plan	predizione dei requisiti, costi e lavoro per la manutenzione del software
Quality plan	descrizione delle procedure e standard di qualità utilizzati nel progetto
Validation plan	descrizione degli approcci, risorse e schedule utilizzati per la convalida del sistema

Il planning del progetto è un processo *iterativo* che viene sottoposto ad inevitabili cambiamenti durante lo sviluppo progressivo. Con l'aumentare delle informazioni relative al sistema durante lo sviluppo, è doveroso revisionare i requisiti iniziali, dei cambiamenti nel business possono portare a grandi cambiamenti nei requisiti del progetto, portando ad un eventuale ri-pianificazione totale.

Assuzioni del planning

- Le assunzioni da fare durante la definizione del project plan non devono essere ottimistiche, bensì realistiche.
- Durante lo sviluppo insorgeranno inevitabilmente dei problemi che causeranno dei ritardi nella consegna.
- Le assunzioni iniziali e lo scheduling saranno inevitabilmente soggetti a problemi inaspettati.

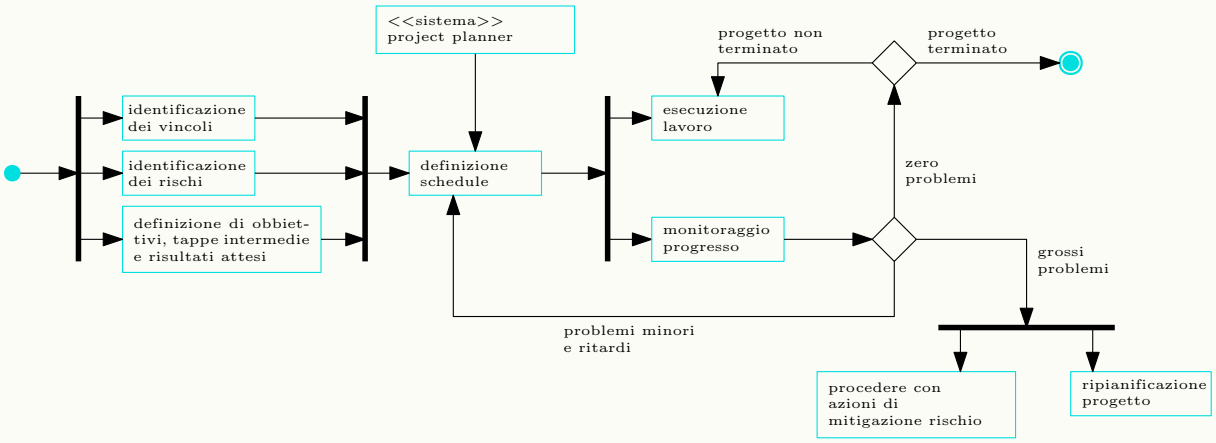


Figura 2.1: Processi del project planning

- Bisogna sempre considerare ogni imprevisto, in modo che eventuali problemi non siano troppo gravanti sulla schedule.

In caso di seri problemi durante lo sviluppo, è necessario applicare delle procedure di *mitigazione del rischio*, onde evitare il fallimento del progetto. In congiunzione a ciò, può essere necessaria la ripianificazione del progetto.
Ciò può comportare la rinegoziazione con il cliente dei risultati attesi e dei vincoli da rispettare. Una nuova schedule, e data di consegna dovrà essere definita in modo da stabilire un accordo con il cliente.

2.2 Scheduling del Progetto

Definizione : : Con *scheduling del progetto* si intende il processo di decisione riguardante il come il carico di lavoro di un progetto deve essere organizzato in differenti attività, ed in che ordine queste attività devono essere eseguite.

Viene stimato un calendario con i tempi necessari al completamento delle attività, lo sforzo necessario ed il personale al quale delegarlo. È anche necessaria una stima delle risorse necessarie, come lo spazio su disco necessario per il software, ed il budget da muovere.

1. Suddivisione del progetto in varie attività e stima delle risorse per ognuna di esse
2. Organizzazione delle attività da eseguire in concorrenza per ottimizzare i tempi
3. Minimizzazione delle dipendenze fra le varie attività, in modo da ridurre i ritardi dovuti ad attese

Questi ultimi fattori dipendono anche dall'intuizione e dalla esperienza del project manager.

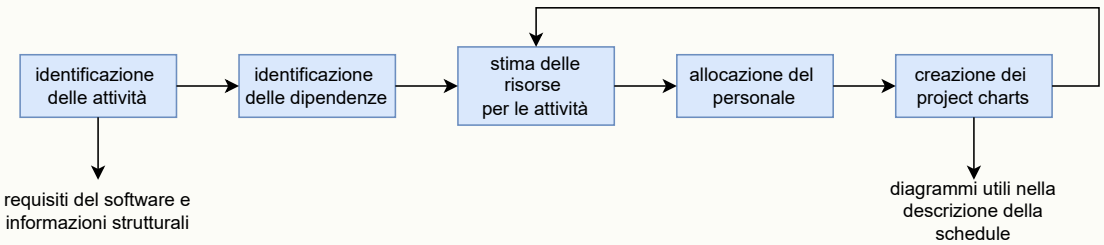


Figura 2.2: Scheduling

Problemi nello scheduling

- Stimare la difficoltà del lavoro ed i costi di sviluppo è molto difficile
- La produttività non è proporzionale al numero di persone coinvolte nel lavoro
- L'aggiunta di personale a progetto già avviato può causare ritardi
- Accade sempre l'inaspettato, bisogna considerare ogni evenienza nel planning

Esiste un annotazione grafica utile nella rappresentazione della schedule, mostra le diverse attività, il periodo in cui vanno terminate e le varie dipendenze fra esse, il diagramma a barre mostra le attività come risorse da disporre sull'asse dei tempi. Le "project activities" sono gli elementi di base del grafico, comprendono

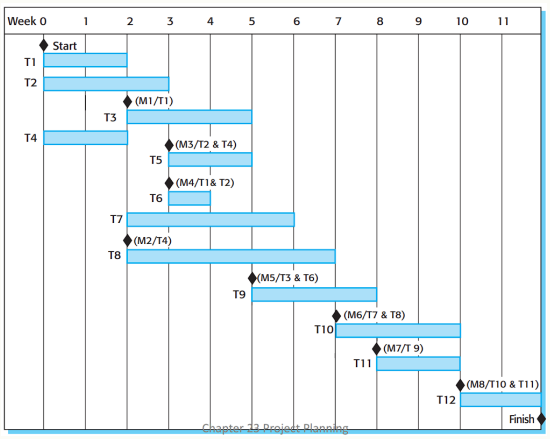
- una durata sul calendario, di giorni o mesi
- un carico di lavoro stimato, misurato in numero di impiegati al giorno necessari
- una deadline per ogni attività che ne vincola il completamento entro una certa data
- un punto specifico che descrive il terminamento di un'attività, può essere un documento , una riunione o il completamento di tutti i test

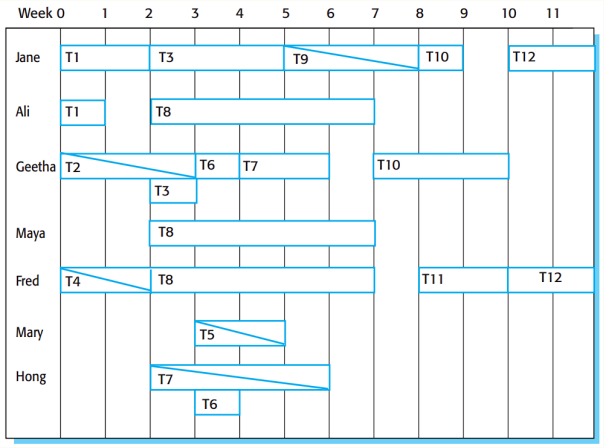
Una **milestone** non è altro che una "tappa fondamentale" durante lo svolgimento delle varie attività, può rappresentare un momento in cui si valuta la progressione del progetto.

Con **deliverables** si definiscono dei risultati ottenuti durante la lavorazione da presentare al cliente.

task	personale necessario (persone/giorno)	durata in giorni	dipendenze
T1	15	10	
T2	8	15	
T3	20	15	T1 (M1)
T4	5	10	
T5	5	10	T2,T4 (M3)
T6	10	5	T1,T2 (M4)
T7	25	20	T1 (M1)
T8	75	25	T4 (M2)
T9	10		T3,T6 (M5)
T10	20	15	T7,T8 (M6)
T11	10	10	T9 (M7)
T12	20	10	T10, T11 (M8)

Nella tabella sopramostrata, sono riportate diverse attività di un processo di sviluppo, ogni attività T_i (o task) ha associato un certo numero di persone necessarie ed una durata in giorni. Inoltre, un task può dipendere da uno o più task. Con M_i si identificano le milestones.





Un *activity bar chart* è un diagramma in cui sull'asse temporale vengono rappresentate le varie attività ed il loro scheduling, in modo che le varie dipendenze siano rispettate.

Analogamente, uno *staff allocation chart* mostra la distribuzione del personale suddiviso nelle varie settimane, indicando il progetto alla quale lavorano.

2.3 Sviluppo Agile

Con sviluppo agile, si denota una metodologia di approccio iterativa in cui il processo di sviluppo ed i requisiti vengono rimodellati dinamicamente, in quanto il progetto in fase di sviluppo viene considerato dal cliente prima del suo completamento. Diversamente dallo sviluppo plan driven, questo approccio non è predeterminato e le decisioni vengono prese durante lo sviluppo, possibile aggiunte o rimozioni dipendono da come il progetto procede e dalle nuove priorità del cliente.

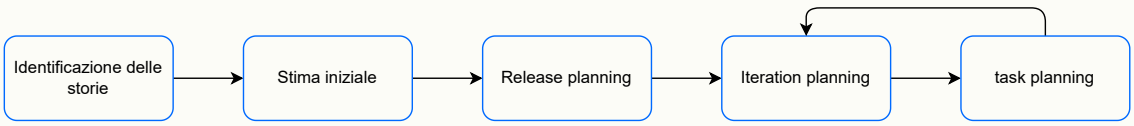
> Le priorità e le esigenze del cliente cambiano, quindi ha senso avere un piano flessibile che possa accogliere questi cambiamenti.

Lo sviluppo agile prevede due stadi

- *Release planning* : prevede la pianificazione a lungo termine delle funzionalità che dovranno essere implementate nei mesi a venire.
- *Iteration planning* : ci si concentra sugli incrementi a breve termine del sistema, tipicamente si tratta di aggiunte che richiedono dalle 2 alle 4 settimane.

Una delle metodologie dello sviluppo agile è la pianificazione basata su storie (**story-based planning**). Consiste nel suddividere le funzionalità di un sistema in piccole unità chiamate "storie utente" (*user stories*). In particolare, sono previsti i seguenti step

1. Creazione delle storie: Le storie utente sono brevi descrizioni di una funzionalità del sistema dal punto di vista dell'utente finale. Solitamente seguono un formato semplice: "Come utente, voglio [azione] in modo da [obiettivo]".
2. Stima dello sforzo: Il team di sviluppo stima il tempo necessario per implementare ciascuna storia, assegnandole un valore numerico (effort points) che riflette la complessità e la dimensione della storia.
3. Prioritizzazione: Le storie vengono ordinate in base alla priorità, tenendo conto di fattori come il valore commerciale, l'urgenza e le dipendenze.
4. Pianificazione delle iterazioni: Le storie prioritarie vengono raggruppate in iterazioni (sprint) di durata definita.
5. Misurazione della velocità: Viene calcolata la "velocità" del team, ovvero la quantità di lavoro (misurata in effort points) che il team riesce a completare in un'iterazione.
6. Previsione: Sulla base della velocità, il team può stimare il tempo totale necessario per completare il progetto.



Il Release Planning si occupa di selezionare e raffinare le storie utente (features) che verranno implementate in una specifica versione (release) del prodotto. Questo processo definisce quali funzionalità saranno incluse nel rilascio e l'ordine in cui verranno sviluppate.

L'Iteration planning si concentra sulla scelta delle storie utente che verranno implementate in ciascuna iterazione (spesso chiamate sprint), che sono periodi di tempo fissi (solitamente 2 o 3 settimane) durante

i quali il team si impegna a completare un insieme di lavoro. Il numero di storie scelte per ogni iterazione dipende dalla capacità del team (velocità) di completare il lavoro.

Durante la fase di pianificazione dei task, gli sviluppatori suddividono le storie utente in compiti di sviluppo più piccoli.

- Un task di sviluppo dovrebbe richiedere tra le 4 e le 16 ore.
- Tutti i task necessari per completare tutte le storie di quell'iterazione vengono elencati.
- I singoli sviluppatori si offrono volontari per i task specifici che desiderano implementare.

Benefici di questo approccio:

- L'intero team ha una visione d'insieme dei task da completare in un'iterazione.
- Gli sviluppatori si sentono responsabili dei loro task e questo li motiva a completarli.

Un incremento del software[1] viene sempre consegnato alla fine di ogni iterazione del progetto. Se le funzionalità da includere nell'incremento non possono essere completate nel tempo previsto, si riducono gli obbiettivi prefissati del lavoro, ma la tempistica di consegna non viene mai prolungata.

Svantaggi dell'agile planning

- La pianificazione Agile dipende fortemente dal coinvolgimento e dalla disponibilità del cliente.
- può essere difficile da organizzare, poiché i clienti devono spesso dare priorità ad altro lavoro e non sono sempre disponibili per le sessioni di pianificazione.
- alcuni clienti potrebbero essere più abituati ai piani di progetto tradizionali e trovare difficile partecipare a un processo di pianificazione agile.

La pianificazione agile funziona bene con team di sviluppo piccoli e stabili che possono riunirsi e discutere le storie da implementare. Tuttavia, quando i team sono grandi e/o geograficamente distribuiti, o quando la composizione del team cambia frequentemente, è praticamente impossibile coinvolgere tutti nella pianificazione collaborativa che è essenziale per la gestione agile di un progetto.

> Lo sviluppo agile è un approccio molto efficace per team piccoli e coesi, richiede adattamenti e strumenti specifici per funzionare bene in contesti più grandi e complessi.

[1] un incremento del software rappresenta una porzione funzionante e valutabile del prodotto software che viene consegnata alla fine di ogni iterazione

CAPITOLO

3

PROCESSI SOFTWARE

3.1 Definizione e Modelli

Definizione : Il processo software è un insieme strutturato di attività necessarie per sviluppare un sistema software. Esistono diversi processi software, ma tutti coinvolgono specifiche, progettazione e implementazione, validazione ed evoluzione del sistema per rispondere ai bisogni dei clienti.

Quando descriviamo e discutiamo di processi, di solito parliamo delle attività che li compongono, ad esempio, la specifica di un modello di dati, la progettazione dell'interfaccia utente, ecc., e dell'ordine in cui queste attività vengono eseguite. Le descrizioni dei processi possono includere anche:

- Prodotti: ovvero i risultati di un'attività del processo;
- Ruoli: che riflettono le responsabilità delle persone coinvolte nel processo;
- Pre-condizioni e post-condizioni: ossia affermazioni che sono vere prima e dopo che un'attività del processo è stata eseguita o un prodotto è stato creato.

Come già accennato, i processi software possono essere di due tipi

- **processi plan-driven** : Sono processi in cui tutte le attività vengono pianificate in anticipo e l'avanzamento del progetto viene misurato in base a questo piano iniziale. È come seguire una ricetta precisa, dove ogni passo è definito fin dall'inizio
- **processi agili** : In questi processi, la pianificazione è più flessibile e si adatta ai cambiamenti. Si lavora a iterazioni brevi (sprint) e si può modificare il piano di volta in volta in base alle nuove esigenze del cliente. È come cucinare senza una ricetta precisa, ma adattando i piatti ai gusti degli ospiti man mano che si procede.

Nella pratica La maggior parte dei progetti software combina elementi di entrambi gli approcci. A volte è necessario un piano dettagliato per le fasi iniziali, mentre in altre si può essere più flessibili.

continua dalle slide "Ch2 SW Processes" - dalla slide 7 fino alla slide 19