Informazioni sul documento

Nome documento	Glossario di Ingegneria del Software
Data redazione	2014-11-07
Redattori	Tesser Paolo

Sommario

Lo scopo del documento è quello di fornire una distribuzione per i termini più utilizzati e significativi nel mondo dell'ingegneria del software.

Indice

1 A

- **Ambiente di lavoro**: è ciò che serve ai processi di produzione. É composto da:
  - persone
  - ruoli
  - procedure
  - infrastrutture

  Esso influisce sulla qualità del processo e del prodotto. Deve essere quindi: completo, ordinato e aggiornato;

- **Amministratore**: è uno dei ruoli di gestione di un progetto. Esso ha il compito di controllare l'ambiente di lavoro e di amministrare le infrastrutture necessarie allo svolgimento del progetto.
  Deve mettere in pratica ciò che le norme chiedono e attraverso sistemi automatizzati le deve fare eseguire senza renderle troppo ingombranti per gli altri membri. Questo lavoro va fatto in maniera preventiva e pro attiva rispetto l'inizio delle attività e del tutto trasparente agli altri componenti.
  A questa figura fanno quindi capo le configurazioni sui sistemi di versionamento e tutta la documentazione che viene redatta. Non attua scelte personali, ma segue quelle concordate con i responsabili;

- **Analista**: è uno dei ruoli di gestione di un progetto. Esso ha il compito di capire il problema per ottenere i requisiti. Non da la soluzione e spesso non segue fino in fondo la realizzazione del progetto, ma è presente principalmente nella fase iniziale.
  Per cercare i requisiti segue un approccio top-down. Parte infatti ad analizzare quelli espliciti del proponente fino a scinderli in vari sottogruppi gerarchici, trovandone nel frattempo anche di impliciti (il numero maggiore tra le due tipologie);

2 B

- **Best Practice (migliore prassi)**: modo di fare che per esperienza e per studio si è comportato bene in determinate circostanze, note e specifiche, e abbia mostrato di garantire i risultati migliori;
- **Broken Window Theory**: teoria sociologica, secondo la quale se si impiegano le risorse nella cura e nel rispetto delle cose che già ci sono si ottengono risultati migliori dell'attuazione di norme repressive. Questo perché la gente attua un emulazione del comportamento della comunità;

3 C

- **Casi d'uso**: tecnica usata nei processi di ingegneria del software per effettuare in maniera esaustiva e non ambigua, la raccolta dei requisiti al fine di produrre software di qualità. Consiste nel valutare ogni requisito, focalizzandosi sugli attori che interagiscono col sistema e valutandone le varie interazioni. Il documento dei casi d'uso, individua e descrive gli scenari elementari di utilizzo del sistema, da parte degli attori che si inter facciano con esso.

- **Ciclo di Deming** : è un modello studiato per il miglioramento continuo della qualità in un'ottica a lungo raggio. Serve per promuovere una cultura della qualità che è tesa al miglioramento continuo dei processi e all'utilizzo ottimale delle risorse. Questo strumento parte dall'assunto che per il raggiungimento del massimo della qualità sia necessaria la costante interazione tra ricerca, progettazione, test, produzione e vendita. Per migliorare la qualità e soddisfare il cliente, le quattro fasi devono ruotare costantemente, tenendo come criterio principale la qualità. La sequenza logica dei quattro punti ripetuti per un miglioramento continuo è la seguente:

  - P: Plan. Pianificazione;
  - D: Do. Esecuzione del programma, dapprima in contesti circoscritti;
  - C: Check. Test e controllo, studio e raccolta dei risultati e dei riscontri;
  - A: Act. Azione per rendere definitivo e/o migliorare il processo.

- **Closure**: nei linguaggi di programmazione è una astrazione che combina una funzione con le variabili libere presenti nell'ambiente in cui è definita.

- **Controllo di versione**: è la gestione di versioni multiple di un insieme di informazioni. I documenti che vengono gestiti attraverso questa tecnica sono tutti quelli che hanno o necessitano di avere una storia.
  Questi strumenti (Git, SVN, Mercurial) ci permettono di muoverci attraverso le varie modifiche che sono state effettuate su file digitali come del codice sorgente, disegni tecnici, documentazione o altro, e, in caso di necessità, ripristinare quelli antecedenti alla versione attuale.
  Tutto ciò serve per correggere errori o ritornare a stati in cui non ce ne erano. Ci permette inoltre di procedere su rami diversi in modo da non entrare in conflitto con sezioni stabili dello sviluppo;

4 D

- **Determinismo**: è la concezione della realtà secondo la quale tutti i fenomeni del mondo sono collegati l'un l'altro e si verificano secondo un ordine necessario e invariabile (il che esclude la presenza del libero arbitrio).
  In informatica possiamo applicare questo concetto al fatto che una funzione, o un determinato evento, dia sempre un certo output dato sempre lo stesso input e che quel risultato sarà disponibile in un tempo quantificabile;

- **Diagramma di GANTT**: è uno degli strumenti di supporto alle attività di pianificazione eseguite dal responsabile di progetto.
  Serve per gestire la dislocazione temporale delle attività, per rappresentarne la durata, preventivata e quella poi effettiva, la sequenzialità o il parallelismo.
  L'asse delle ascisse è destinata allo scorrere del tempo ed è lunga quanto la durata del progetto.
  L'asse delle ordinate invece è riservata alle mansioni che costituiscono il progetto.
  Le barre orizzontali rappresentate servono ad indicare la quantità di tempo riservato ad una determinata mansione. Ne deriva quindi un calendario delle attività.
  Questo metodo offre quindi un'idea più chiara sull'andamento del progetto e permette di controllare meglio i budget di risorse e conseguentemente i costi;

- **Diagramma di PERT**: (Programme Evaluation and Review Technique) è uno degli strumenti di supporto alle attività di pianificazione eseguite dal responsabile di progetto.
  Serve per arricchire le informazioni date dal diagramma di Gantt. In esso vengono gestite principalmente le dipendenze tra un'attività e un'altra, andando a segnare il tempo di inizio e quello di fine. Tra le attività correlate si potrà dunque verificare se è presente dello **Slack**, cioè del tempo di margine in caso l'attività precedente non riesca ad essere eseguita nei tempi prefissati. Bisogna stare attenti a come si gestisce ciò perché molto slack causa inattività e non consente ritardo finale.
  Questa tecnica, congiunta allo studio del Cammino critico, permette di visualizzare meglio i collegamenti tra le varie attività e consenti di valutare più facilmente i percorsi critici;

- **Disciplina**: insieme di regole e procedure che attuate conducono a un risultato conforme a determinate aspettative;

- **Disciplinato**: fissato un principio, lo segue sempre senza aggiungere o fare cose che non sono state regolamentate precedentemente da norme di gruppo;

- **Documentazione**: riguarda tutto ciò che documenta le attività di un progetto, riguardo al prodotto e al processo. Per essere utile deve sempre essere disponibile, corretta nei contenuti, verificata, approvata e aggiornata. La diffusione di questi documenti deve essere controllata. Ogni documento infatti deve avere una lista di distribuzione, coerente con i suoi scopi, per non generare offuscamento in chi la riceve.

- **Duck typing**: nei linguaggi di programmazione ad oggetti si riferisce ad uno stile di tipizzazione dinamica dove la semantica di un oggetto è determinata

dall'insieme corrente dei suoi metodi e dalla sue proprietà piuttosto che dal fatto di estendere una particolare classe o implementare una specifica interfaccia. Alcuni linguaggi a cui si riesce ad applicare bene questo principio sono JavaScript o PHP. Il nome si riferisce al duck test attribuito a James Whitcomb Riley il quale si può esporre attraverso la frase: "Quando vedo un uccello camminare come un'anatra, nuotare come un'anatra e fare il verso come l'anatra, posso chiamare quell'uccello anatra.".

5 F

- **Efficacia**: è la capacità di raggiungere l'obiettivo specifico che si cercava di raggiungere applicando una determinata azione.
  Viene determinata dal grado di conformità del prodotto rispetto alle norme vigenti e agli obiettivi prefissati;

- **Efficienza**: è la capacità di produrre qualcosa, impiegando il minor numero di risorse che siano esse tempo, denaro o persone durante l'esecuzione delle attività richieste;

6 F

7 G

8 H

9 I

- **Incremento**: procedere in questo modo significa aggiungere ad un impianto base. Si inserisce, ma non si toglie;
- **Iterazione**: procedere in questo modo significa operare raffinamenti o rivisitazioni. A differenza della tecnica incrementale, questa può essere distruttiva. Devo inoltre fissare un numero preciso di iterazioni da compiere e per farlo ha bisogno di condizioni di uscita forti dal ciclo;

10 L

- **Lambda Calcolo**: TO DO;

11 K

12 M

- **MEAN**: si riferisce alle iniziali delle quattro componenti (free e open source) per la costruzione di siti web dinamici.
  In particolare è uno stack JavaScript per creare applicazioni web.
  I componenti sono:
  - Mondo DB : a NoSQL database;
  - Express.js : un framework per applicazioni web costruito in Node.js;
  - Angular.js : un framework basato sul design pattern MVC per web app;
  - Node.js : una piattaforma software per gestire servizi web scalabili.
- **Metrica di progetto (Metrics)**: sistema di misurazione composto da valori e metodi per valutarli in maniera oggettiva;
- **Milestone**: TO DO;
- **Modelli agili**: è uno dei modelli di ciclo di vita del SW.
  Nell'idea pura della metodologia agile viene perso il legame con l'ingegneria del software. Quello che viene proposto infatti è che le persone sono più importanti dei processi e degli strumenti imposti. Di seguito sono riportati i quattro principi fondanti:
  1. gli individui e le interazioni tra di essi sono più importanti dei processi e degli strumenti;
  2. un software funzionante piuttosto che una documentazione comprensiva;
  3. collaborazione con il cliente piuttosto che una negoziazione contrattuale;
  4. adattarsi ai cambiamenti piuttosto che seguire un piano preciso e rigido.

  Questo modello ha come obiettivi quelli di riuscire a dimostrare costantemente al cliente quanto è stato fatto. Ciò da maggiore soddisfazione agli sviluppatori che vedono in breve tempo l'avanzamento del risultato finale.
  Alcune tecniche agili sono: Scrum (caos organizzato) e Kanban (just-in-time).
- **Modello a componenti**: è uno dei modelli di ciclo di vita del SW. Nasce dall'osservazione che molto di quello che ci serve esiste già e molto di quello che faremo ci servirà ancora.
  Generalmente costa meno come approccio e, se è stato fatto bene il componente che andremo ad integrare, ci sarà una buona probabilità che sia già testato.
- **Modello evolutivo**: è uno dei modelli di ciclo di vita del SW. Esso aiuta a rispondere a bisogni non inizialmente preventivabili. Non si sa dove si vuole andare, ma si procede migliorando.
  Può richiedere il rilascio e il mantenimento di più versioni esterne in parallelo che consentono una presenza molto vasta sul mercato ,ma ciò può essere estremamente costoso se il prodotto è di tipo commerciale. Alcuni esempi di ciò riguardano Google Chrome e Firefox.
- **Modello incrementale**: è uno dei modelli di ciclo di vita del SW. Prevede rilasci multipli e successivi nei quali avviene un incremento di funzionalità.
  Alcune fasi però, come l'analisi e la progettazione, non vengono ripetute. Solo la realizzazione assume carattere incrementale iniziando dai requisiti essenziali

per poi terminare con quelli desiderabile. In tal modo, già dopo non troppo tempo, si possono mostrare i risultati al committente e ricevere dei feedback. Così è più facile e meno costoso effettuare dei miglioramenti o dei raffinamenti se necessario;

- **Modello sequenziale (a cascata)**: è uno dei modelli di ciclo di vita del SW più consolidati e autoritari. Prevede una successione di fasi rigidamente sequenziali, non ammettendo ritorni a stati precedenti. Se si verificano eventi eccezionali, essi faranno ripartire dall'inizio.
  Per passare da uno stato ad un altro vengono introdotte delle PRE e delle POST condizioni molto forti definite all'origine. Le fasi previste sono: Analisi, Progettazione, Realizzazione, Manutenzione. Esse sono distinte e non si sovrappongono.
  Questo modello però pecca di eccessiva rigidità e spesso il committente non sa da subito quali sono le sue esigenze in maniera precisa.
  Ci sono quindi delle correzioni che si possono apportare per renderlo più flessibile. Si può fare della prototipazione solo per capire meglio i requisiti o attuare dei ritorni, in modo che ogni ciclo di ritorno raggruppo delle sotto sequenze di fasi;

- **Modello a spirale**: è uno dei modelli di ciclo di vita del SW. Viene impiegato in progetti fortemente innovativi perché si ha un miglior controllo dei rischi (risk driven).
  Si attua mediante cicli interni rapidi e ripetuti che consistono in analisi e sviluppo di prototipi, mentre per i cicli esterni si può aderire a un qualsiasi degli altri modelli.
  Richiede quindi forte interazione tra il committente e il fornitore. Le fasi presenti sono: definizione degli obiettivi, l'analisi dei rischi, sviluppo e validazione, pianificazione per poi riprendere dall'inizio;

13 N

14 O

15 P

- **Piano di Progetto**: è uno dei documenti esterni da redarre. Viene steso dal responsabile di progetto.
  In esso vengono fissate le risorse disponibili, la suddivisione delle attività e il loro calendario.
  Si deve cercare di organizzare le attività in modo da produrre risultati utili per valutare con efficacia il grado di avanzamento del lavoro.
  Serve inoltre fissare delle milestone come punti critici o finali delle attività.
  La struttura tipica di questo documento segue questo schema:

  1. Introduzione (scopo e struttura);
  2. Organizzazione del progetto;
  3. Analisi dei rischi, maggiori sono i rischi maggiori saranno i punti di controllo.
     Le tipologie di rischi possibili sono: a livello tecnologico, a livello del personale, a livello organizzativo, a livello dei requisiti (quello che causa più probabilità di fallimento) e a livello di valutazione dei costi;
  4. Risorse necessarie e risorse disponibili (HW e SW);
  5. Suddivisione del lavoro (Work Breakdown);
  6. Calendario delle attività;
  7. Meccanismi di controllo e di rendicontazione.

- **Pro attivi**: al contrario di re-attivi, si è pro attivi quando si cerca di risolvere un problema prima che esso si manifesti. In particolare riguarda gli errori che si potranno commettere e cercare di prevenirli.
  Agire prima permette di ridurre i costi oltre al fatto di capire meglio ciò che si sta facendo e garantire una migliore qualità del prodotto;

- **Procedura**: in generale è l'insieme di regole e norme da svolgere in maniera sequenziale per ottenere un determinato risultato.
  In informatica (funzione) è un blocco di istruzioni che a partire da un certo input restituiscono un determinato output;

- **Processo**: in generale è una rete di cambiamenti, attività o azioni collegate tra loro per la creazione, la verifica, la manutenzione, ecc di un prodotto.
  Durante il ciclo di vita del software tramite un processo si attua la transizione tra uno stato e l'altro. Nel particolare vengono chiamati processi di ciclo di vita.
  In Informatica si intende un'istanza di un programma in esecuzione in modo sequenziale.
  A differenza di un programma, esso è un'entità dinamica, che dipende dai dati che vengono elaborati, e dalle operazioni eseguite su di essi. Il processo è quindi caratterizzato, oltre che dal codice eseguibile, dall'insieme di tutte le informazioni che ne definiscono lo stato, come il contenuto della memoria indirizzata, i thread, i descrittori dei file e delle periferiche in uso;

- **Progettista**: è uno dei ruoli di gestione di un progetto.
  Esso ha il compito di trovare la migliore soluzione per i requisiti trovati dall'analista. Per fare ciò generalmente utilizza un approccio bottom-up.

Non si occupa dell'implementazione, compito spettante invece al programmatore. La sua presenza nel progetto dura in genere fino alla manutenzione.

- **Programmatore**: è uno dei ruoli di gestione di un progetto.
  Esso ha il compito di implementare le soluzioni trovate dal progettista. Non si deve inventare nulla proprio perché quello che deve fare è già stato fissato in precedenza. Partecipano sia allo sviluppo che alla manutenzione.
- **Programma**: è un insieme di istruzioni che, una volta eseguite su un computer, produce soluzioni per una data classe di problemi automatizzati;

16 Q

- **Quantificabile**: di cui è possibile misurarne l'efficacia e l'efficienza, anche a priori;

17 R

- **Responsabile**: è uno dei ruoli di gestione di progetto. Esso rappresenta il progetto di tutto il team presso il fornitore e il committente. Viene quindi accentrata la responsabilità di scelta e approvazione.
  Partecipa interamente al ciclo di vita del prodotto e sa sempre (senza bisogno di domandare nulla agli altri membri) a che punto è l'avanzamento delle attività per il raggiungimento degli obiettivi fissati.
  Ha quindi responsabilità sulla pianificazione, sulla gestione delle risorse umane tramite coordinamento e relazioni esterne con il committente.
  La pianificazione consiste nella definizione delle attività che il team dovrà svolgere e l'assegnamento dei tempi di esecuzione per quelle attività riuscendo a determinarne i costi di attuazione, da fornire poi come preventivo al proponente e come consuntivo nella fase finale.
  Ci sono diversi strumenti per lo svolgimento di questo ruolo, come l'utilizzo di Diagrammi di Gantt e di PERT .

- **Ruolo**: è una funzione aziendale assegnata a progetto. E' un servizio che viene svolto per la comunità. Ne esistono di diversi e non tutti ammettono parallelismo.
  Ci sono diversi ruoli che si possono raggruppare più macroscopicamente in quattro gruppi:

  1. Sviluppo: ha la responsabilità tecnica e realizzativa;
  2. Direzione: ha la responsabilità decisionale (molto importante);
  3. Amministrazione: ha la responsabilità della gestione dei processi;
  4. Qualità: ha la responsabilità di gestione della qualità.

18 S

- **Scrum**: TO DO;
- **SEMAT**: Software Engineering Method and Theory è una comunità industriale con lo scopo di migliorare le pratiche dell'ingegneria del software, ridefinendola come una disciplina rigorosa;
- **Servizio**: è un mezzo per fornire valore all'utente, agevolando il raggiungimento dei suoi obiettivi senza doverne sostenere gli specifici costi e rischi;
- **Sistematico**: costante, che fa sempre la stessa cosa in determinate situazioni, applicando un comportamento, che deve tendere ad esser best, ripetitivo;
- **Stakeholder (portatore di interesse)**: è l'insieme di persone coinvolte nel ciclo di vita del SW con influenza sul prodotto.
  Possono essere chi usa lo usa, chi lo paga o chi lo sviluppa;

19 T

20 U

- UML: acronimo di Unified Modeling Language, (linguaggio di modellazione unificato), è un linguaggio di modellazione e specifica basato sul paradigma object-oriented. Consente di costruire modelli object-oriented per rappresentare domini di diverso genere. Nel contesto dell'ingegneria del software, viene usato soprattutto per descrivere il dominio applicativo di un sistema software e/o il comportamento e la struttura del sistema stesso.

21 V

- **Verificatore**: è uno dei ruoli di gestione di un progetto. Necessità come per la categoria dei programmatori di molte unità. Partecipano all'intero ciclo di vita in quanto servono a mantenere la qualità e aiutare a rimuovere gli errori.

22 W

- **Work Breakdown Structure**: Breakdown significa esplosione, decomposizione. Questa attività serve per identificare tutti i compiti che dovranno essere svolti dai membri di un progetto. I compiti sono appunto delle attività che si compongono di altre sotto attività. Sono unicamente identificate e possono non essere svolte in ordine sequenziale.
  Si parte quindi da un macro obiettivo fino a decomporlo in compiti sempre più piccoli (ma non troppo) tali da poter essere assegnati a una singola persona e che essa possa comprenderli facilmente.
  Bisogna quindi stare molto attenti sia a non sottostimare il problema, sia a non sovra-stimarlo.

23 Y

24 Z