Glossario

7DOS - 10 Gennaio 2019

Informazioni sul documento	
Versione	1.0.0
Responsabile	Lorenzo Busin
Verifica	Giacomo Barzon Michele Roverato Giovanni Sorice
Redazione	Marco Costantino Nicolò Tartaggia Andrea Trevisin
Stato	Approvato
Uso	Esterno
Destinato a	Prof.Tullio Vardanega Prof.Riccardo Cardin Zucchetti 7DOS
Email	7dos.swe@gmail.com

Descrizione

Questo documento rappresenta il glossario dei termini utilizzati nella documentazione di progetto*G&B*.

Diario delle modifiche

Modifica	Autore	Ruolo	Data	Versione
Approvazione del documento	Lorenzo Busin	Responsabile	2019-01-10	1.0.0
Verifica del documento	Michele Roverato	Verificatore	2019-01-09	0.4.0
Verifica del documento	Giovanni Sorice	Verificatore	2019-01-07	0.3.0
Verifica del documento	Giacomo Barzon	Verificatore	2019-01-05	0.2.0
Stesura termini	Marco Costantino	Analista	2019-01-04	0.1.6
Stesura termini	Marco Costantino	Analista	2018-12-29	0.1.5
Stesura termini	Nicolò Tartaggia	Analista	2018-12-21	0.1.4
Stesura termini	Andrea Trevisin	Analista	2018-12-18	0.1.3
Stesura termini	Andrea Trevisin	Analista	2018-12-13	0.1.2
Stesura termini	Marco Costantino	Analista	2018-12-10	0.1.1
Stesura termini	Nicolò Tartaggia	Analista	2018-12-05	0.1.0
Stesura dello scheletro del documento	Andrea Trevisin	Analista	2018-12-05	0.0.1

Indice

1 A . . . 6
1.1 Attività . . . 6
1.2 Analisi dinamica . . . 6
1.3 Analisi dei requisiti . . . 6
1.4 Analisi statica . . . 6
1.5 API Rest . . . 6

2 B . . . 7
2.1 Baseline . . . 7
2.2 Best Practice . . . 7
2.3 Brainstorming . . . 7

3 C . . . 8
3.1 Capitolato d'appalto . . . 8
3.2 Ciclo di vita del software . . . 8
3.3 Cloud . . . 8
3.4 CoCoMo . . . 8
3.5 Commit . . . 8
3.6 Compito . . . 9
3.7 Code-'n-Fix . . . 9
3.8 Conditional Probability Table(CPT) . . . 9
3.9 Configurazione . . . 9
3.10 Continuous Delivery . . . 9
3.11 Continuous Integration . . . 9
3.12 Controllo di configurazione . . . 9

4 D . . . 10
4.1 Dashboard . . . 10
4.2 DevOps . . . 10
4.3 Disciplinato . . . 10
4.4 Docker . . . 10

5 E . . . 10
5.1 Economicità . . . 10
5.2 Efficacia . . . 10
5.3 Efficienza . . . 10

6 G . . . 11
6.1 Git . . . 11
6.2 GitLab . . . 11
6.3 Grafana . . . 11

7 H . . . 11
7.1 HTML . . . 11

8 I **11**
8.1 IEE 830-1998 11
8.2 Incremento 11
8.3 InfluxDB 11
8.4 Intelligenza artificiale 12
8.5 ISO/IEC 12207 12
8.6 ISO/IEC 15504 12
8.7 ISO/IEC 25010 12
8.8 ISO/IEC/IEEE 42010:2011 12
8.9 ISO 90003:2004 13
8.10 Issue Tracking System 13
8.11 Iterazione 13

9 J **14**
9.1 Java 14
9.2 JavaScript 14
9.3 JSON 14

10 L **15**
10.1 Liveliness 15
10.2 Likelihood 15

11 M **15**
11.1 Management 15
11.2 Manuale della qualità 15
11.3 Manutenzione 15
11.3.1 M. correttiva 15
11.3.2 M. adattiva 15
11.3.3 M. evolutiva 15
11.4 Miglioramento continuo 15
11.5 Minimizzazione 16
11.6 Modello di ciclo di vita 16
11.6.1 Sequenziale (A cascata) 16
11.6.2 Incrementale 16
11.6.3 Evolutivo 16
11.6.4 A componenti 16
11.6.5 A spirale 16
11.6.6 Agili 16
11.6.7 Scrum 17

12 N **18**
12.1 Node.js 18

13 O **19**
13.1 Obiettivo SMART 19
13.2 Open source 19

14 P **20**
14.1 Panel 20
14.2 PDCA 20
14.3 Performance 20
14.4 Perl 20
14.5 Pianificazione delle attività 20
14.6 Piano di progetto 20
14.7 Plug-in 21
14.8 Processo 21
14.9 Processo software 21
14.10 Processo specializzato per progetto 21
14.11 Prodotto software 21
14.11.1 Commessa 21
14.11.2 Pacchetto 21
14.11.3 Componente 22
14.11.4 Servizio 22
14.12 Prodotti documentali 22
14.13 Produttività 22
14.14 Progettazione Software 22
14.15 Progetto 22
14.16 Project Manager 22
14.17 Prototipo 23
14.18 Publish/subscribe 24
14.19 Python 24

15 Q **25**
15.1 Quantificabile 25
15.2 Qualità 25
15.2.1 Qualità del prodotto 25
15.2.2 Qualità dei processi 25

16 R **26**
16.1 Raintank 26
16.2 Redmine 26
16.3 Repository 26
16.4 Requisito 26
16.5 Rete Bayesiana 26
16.6 Rischio 27
16.7 Riuso 27
16.8 Ruolo 27

17 S **28**
17.1 Script 28
17.2 Sistematico 28
17.3 Slack (tempo) 28
17.4 Slack (software) 28

17.5 SonarQube . . . 28
17.6 Software Engineering . . . 28
17.7 Stima dei costi . . . 28
17.8 Studio di fattibilità . . . 28

18 T . . . 29
18.1 Telegram . . . 29
18.2 Tracciamento dei requisiti . . . 29

19 V . . . 30
19.1 Validare . . . 30
19.2 Verificare . . . 30
19.3 Verificatore di requisiti . . . 30
19.4 Versionamento . . . 30
19.5 Versione . . . 30

20 W . . . 31
20.1 Way of working . . . 31

21 Z . . . 32
21.1 Zero-latency . . . 32
21.2 Zero-laxity . . . 32

1 A

1.1 Attività

L'attività è una componente essenziale di un progetto. Un'attività prevede dell'intenzionalità: le specifiche dell'attività sono determinate da chi la svolge.

1.2 Analisi dinamica

Tecnica di analisi di un prodotto software in cui il quest'ultimo viene eseguito in un ambiente reale o virtuale; coincide con i test:

- Di unità: verificano il corretto funzionamento dell'unità.
- Di integrazione: verifica la corretta interazione tra più unità che interagiscono dando luogo ad un *build*.
- Di sistema: verificano la corretta interazione tra più *build* che compongono un sistema complesso.

Il collaudo, test finale eseguito con il committente, è solo un attività formale se i test precedenti sono stati eseguiti rigorosamente. I test devono essere ripetibili e deterministici, perciò vanno documentati: le condizioni iniziali di esecuzione, l'esecuzione, il risultato atteso ed il risultato ottenuto. I test inoltre devono essere automatizzati per ridurre al minimo la possibilità di errore umano.

1.3 Analisi dei requisiti

L'obiettivo dell'analisi dei requisiti è quello di individuare e definire i requisiti di progetto. Il risultato di tale processo è un documento contrattuale che andrà fornito all'azienda appaltatrice nell'ambito di una gara d'appalto. Sulla base di questo documento l'azienda deciderà a chi affidare l'appalto.

1.4 Analisi statica

Tecnica di analisi di un prodotto software in cui questo non viene eseguito, bensì ne viene esaminato il codice. *Walkthrough* e *Inspection* sono le due tecniche di analisi statica prevalenti: *Walkthrough* prevede un'analisi a tappeto del codice sorgente alla ricerca di possibili criticità, mentre *Inspection* somiglia di più all'analisi a campione e prevede l'analisi specifica di parti considerate di importanza critica.

1.5 Apache Kafka

Piattaforma di *stream processing* open source scritta in Java e Scala.

1.6 API Rest

Note anche come *RESTful API*, costituiscono un metodo per permettere la comunicazione tra un client web e un server che implementa un'architettura REST ("*REpresentational State Transfer*").

2 B

2.1 Baseline

La *baseline* è in generale un punto di partenza, il piano di progetto originale. Essa ha diverse declinazioni che indirizzano un obiettivo strategico e il cui scopo è aiutare a misurare l'avanzamento del processo nella direzione degli obiettivi prefissati. Questi vengono concordati con il committente, in modo di dimostrare l'avanzamento del progetto. La *baseline* è suddivisa in parti, definite nel modo migliore per aiutare a raggiungere gli obiettivi. Ogni parte evolve nel tempo ed ha quindi un numero di versioni, a cura dell'*owner* (responsabile) della parte. Il mantenimento della *baseline* avviene tramite la gestione della configurazione.

2.2 Best Practice

Una *best practice* è il modo migliore di approcciare un problema. Le *best practice* prevedono l'applicazione di principi noti ed autorevoli. In ingegneria è richiesto conoscere ed usare le *best practice* esistenti, non crearne di nuove.

2.3 Brainstorming

Discussioni collaborative e creative, in cui viene data voce ad ogni persona a turno. Hanno l'intento di far emergere più spunti e concetti possibili, per poi operare una selezione di quelli più interessanti.

3 C

3.1 Capitolato d'appalto

Documento, prodotto da un azienda, che descrive un prodotto o sistema che questa desidera sia realizzato. Il capitolato è una chiamata ai fornitori, una notifica di bisogno che andrà esaminata per verificare se è il caso di partecipare al bando d'appalto per la realizzazione del prodotto richiesto. Costituisce la prima fonte di informazioni da analizzare per l'analisi dei requisiti. Il dominio dell'azienda appaltatrice influenzerà le informazioni contenute nel capitolato che quindi dovrà essere attentamente analizzato tenendo conto del contesto professionale da cui arriva; sarà necessario un dialogo con l'azienda appaltatrice per la corretta comprensione del capitolato.

3.2 Ciclo di vita del software

In ingegneria del software, modo in cui l'attività di realizzazione di un prodotto software viene scomposta in sotto-attività coordinate tra loro, e il cui risultato é il prodotto software e tutta la documentazione necessaria. Il ciclo di vita del software varia a seconda della metodologia di sviluppo adottata.

3.3 Cloud

Abbreviativo di "*cloud computing*" ("computazione sulle nuvole", sta ad indicare un paradigma di erogazione di servizi informatici da un fornitore ad un cliente attraverso Internet. In particolare, il fornitore permette al cliente di accedere e configurare delle risorse fisiche in remoto, *on demand*.

3.4 CoCoMo

Modello algoritmico per la stima di costi e risorse. Esso stima le risorse necessarie e le esprime in mesi/persona (MP). Ha come input:

- La complessità del progetto;
- Le dimensioni del SW da sviluppare;
- Il peso della complessità sullo sviluppo;
- Un coefficiente moltiplicativo (parte da 1);
- Un fattore di espansione del tempo (parte da 2.5);
- Un coefficiente di complessità.

3.5 Commit

Termine inglese che significa "impegnarsi", "prendere l'impegno". In ambito informatico indica l'azione di aggiungere gli ultimi cambiamenti alla repository.

3.6 Compito

Un compito è una componente essenziale di un progetto. I compiti vengono assegnati e non lasciano spazio alla decisione di chi li riceve.

3.7 Code-'n-Fix

Vecchia pratica nell'ambito di produzione del software, autodescrittiva: l'intera attività di produzione consisteva nel codificare e riparare software.

3.8 Conditional Probability Table(CPT)

Una Conditional Probability Table (CPT) è una tabella in cui:

- Ogni colonna rappresenta l'i-esimo stato in cui il nodo corrente può risiedere. Ad ogni stato è identificato da un nome ed un intervallo di valori associato. Un nodo non può possedere più stati con intervalli di valori sovrapposti tra loro.
- Ogni riga rappresenta la j-esima combinazione esistente di tutti i possibili stati dei nodi predecessori
- le celle interne indicano la probabilità, condizionata dai nodi predecessori, che il nodo corrente si trovi in uno specifico stato data una combinazione esistente dei possibili stati dei nodi predecessori

3.9 Configurazione

Insieme di regole che determina come assemblare le sezioni di un software, quali versioni usare per ogni sezione, come le sezioni interagiscono, quali sezioni con quali versioni producono una baseline , etc. etc..

3.10 Continuous Delivery

Approccio di software engineering che prevede la produzione di prodotti software in brevi cicli, assicurandosi che sia possibile rilasciare il software manualmente in qualsiasi momento. Ha come obiettivo l'incremento delle velocità di sviluppo, testing e rilascio del prodotto software.

3.11 Continuous Integration

Pratica di sviluppo che richiede che gli sviluppatori integrino il codice in una repository condivisa più volte al giorno. Ogni integrazione è verificata mediante un build automatizzato, che permette di rilevare errori immediatamente.

3.12 Controllo di configurazione

Gestione e controllo della configurazione che permette di assemblare le varie componenti di un software.

4 D

4.1 Dashboard

Una dashboard (in italiano cruscotto) è una schermata che permette di monitorare in tempo reale l'andamento dei report e delle metriche aziendali. Una dashboard può presentare uno o più panel al suo interno.

4.2 DevOps

Termine derivante dalla contrazione delle parole inglesi *development* ("sviluppo") e *operations* ("operazioni" nel senso di "messa in produzione"). Indica un metodo di sviluppo software basato sulla comunicazione e collaborazione tra sviluppatori e addetti alle "*IT operations*", per sviluppare e mantenere velocemente ed efficacemente prodotti e servizi software.

4.3 Disciplinato

Soggetto ad un insieme di regole pensate per garantire la massima efficienza ed efficacia.

4.4 Docker

Prodotto software open source che automatizza il rilascio di applicazioni all'interno di "contenitori" software, grazie alla virtualizzazione a livello di sistema operativo di Linux.

5 E

5.1 Economicità

Efficacia raggiungibile con efficienza. Garantita dall'uso di standard.

5.2 Efficacia

L'abilità di un entità di portare a il compito assegnatole.

5.3 Efficienza

Data la misura del consumo di risorse che avviene nel compimento di un obiettivo, minore il consumo di risorse, maggiore l'efficienza.

6 G

6.1 Git

Sistema di controllo versione, permette di organizzare e tracciare il lavoro di più persone su file condivisi.

6.2 GitLab

Applicazione web per il la gestione di repository. Include una wiki, issue tracking e servizi di pipeline CD/CI sotto licenza open source.

6.3 Grafana

Grafana è un compositore di grafici, che viene eseguito come applicazione web. Supporta InfluxDB.

7 H

7.1 HTML

In informatica l'HyperText Markup Language (HTML) è un linguaggio di markup nato per la formattazione e impaginazione di documenti ipertestuali disponibili nel web. HTML è un linguaggio di pubblico dominio, la cui sintassi è stabilita dal *World Wide Web Consortium* (W3C).

8 I

8.1 IEE 830-1998

Best practice raccomandate per la specifica dei requisiti di prodotti software.

8.2 Incremento

Procedura che prevede avanzamento per aggiunta ad un impianto base.

8.3 InfluxDB

InfluxDB è un database di serie temporali open source. È ottimizzato per la memorizzazione e il recupero rapido e ad alta disponibilità di dati di serie temporali in campi come il monitoraggio delle operazioni, le metriche dell'applicazione, i dati di sensori e l'analisi in tempo reale.

8.4 Intelligenza artificiale

Disciplina nel ramo dell'informatica, si occupa dello studio e della progettazione di sistemi informatici in grado di funzionare in maniera simile alla mente umana. Per raggiungere questo obiettivo, si occupa anche di elaborare modelli matematici che approssimino le modalità di pensiero ed apprendimento caratteristiche dei cervelli biologici.

8.5 ISO/IEC 12207

Standard riferiti ai processi di ciclo di vita, raggruppati in 3 categorie: primari, di supporto, organizzativi.

8.6 ISO/IEC 15504

Standard riferiti alla qualità dei processi, collettivamente noti anche come SPICE (*Software Process Improvement and Capability Determination*). Definiscono una scala di maturità di processo a cinque livelli (più il livello base, detto "livello 0").

8.7 ISO/IEC 25010

Standard riferito alla qualità di prodotto, noto anche come SQuaRE (*Systems and software Quality Requirements and Evaluation*). Descrive un modello di qualità di prodotto che individua 8 caratteristiche principali da rispettare:

- Functional Suitability: idoneità funzionale, ossia quanto il prodotto rispetta i requisiti;
- Performance Efficiency: efficienza della performance, ossia quanto il prodotto è in grado di sfruttare efficientemente le risorse disponibili;
- Compatibility: compatibilità, ossia quanto il prodotto riesce ad interfacciarsi con altri prodotti o sistemi;
- Usability: usabilità, ossia quanto il prodotto risulta di semplice utilizzo;
- Reliability: affidabilità, ossia quanto il prodotto risulta stabile nell'esecuzione e nei risultati;
- Security: sicurezza, ossia quanto il prodotto protegge i dati trattati;
- Maintainability: manutenibilità, ossia quanto il prodotto si presta alla risoluzione di bug e problemi;
- Portability: portabilità, ossia quanto il prodotto risulta utilizzabile su ambienti diversi tra loro.

8.8 ISO/IEC/IEEE 42010:2011

Standard di best practice concernenti la progettazione software e la definizione dell'architettura del software. Punti essenziali sono:

- La decomposizione del sistema in componenti (utile ad aumentare il parallelismo);

- Definizione delle interfacce dei componenti;
- Definizione dell'organizzazione delle interfacce che permettono l'interazione dei componenti;
- Paradigmi vari per la composizione dei componenti.

8.9 ISO 90003:2004

Standard di best practice per la valutazione della qualità di processi dei fornitori. I principi fondamentali sono:

- L'orientamento al cliente;
- L'obiettivo di leadership sul mercato;
- Il coinvolgimento del personale;
- L'approccio per processi;
- L'obiettivo del miglioramento continuo;
- La presa di decisioni basate su evidenze;
- La gestione delle relazioni.

A garantire l'adesione a questi principi dev'essere la documentazione verticale (specifica di progetto) ed orizzontale (specifica dell'azienda).

8.10 Issue Tracking System

Prodotto software che gestisce e mantiene liste di problematiche. Utile in situazioni di sviluppo collaborativo da parte di un team.

8.11 Iterazione

Procedura che prevede avanzamento per raffinamento e rivisitazioni.

9 J

9.1 Java

Linguaggio di programmazione concorrente, orientato agli oggetti, basato su classi, che si pone lo scopo di ridurre al minimo le dipendenze da implementazione. Programmi scritti in Java vengono poi compilati in *bytecode*, che può venire eseguito su JVM ("*Java Virtual Machine*") indipendentemente dall'architettura del sistema.

9.2 JavaScript

Linguaggio di scripting orientato agli oggetti e agli eventi, comunemente utilizzato nella programmazione web lato client per la creazione, in siti web e applicazioni web, di effetti dinamici interattivi tramite funzioni di script invocate da eventi innescati a loro volta in vari modi dall'utente sulla pagina web in uso.

9.3 JSON

JSON(JavaScript Object Notation) è un formato adatto all'interscambio di dati fra applicazioni client/server basato sul linguaggio JavaScript.

10 L

10.1 Liveliness

Parola inglese, significa "vitalità" in italiano. In ambito informatico si riferisce ad un insieme di proprietà dei sistemi concorrenti, per cui un sistema deve funzionare nonostante i suoi componenti concorrenti debbano "fare a turni" in sezioni critiche.

10.2 Likelihood

Parola inglese, significa "verosimiglianza" in italiano. In ambito probabilistico si usa per indicare la probabilità che un evento si verifichi.

11 M

11.1 Management

L'insieme delle funzioni amministrative, direttive e gestionali di un'impresa o di un'azienda; il compito del manager.

11.2 Manuale della qualità

Documento che specifica le strategie che un organizzazione adotta per operare processi di qualità.

11.3 Manutenzione

Processo correttivo e di sviluppo che avviene dopo il rilascio della versione finale di un prodotto software. Se ne distinguono diversi tipi, come di seguito.

11.3.1 M. correttiva

Ha come scopo la correzione di errori, bug, inesattezze, inefficienze, etc..

11.3.2 M. adattiva

Ha come scopo l'adattamento a diverse tecnologie, ambiti, contesti.

11.3.3 M. evolutiva

Ha come scopo l'adattamento a nuove tecnologie, ambiti, contesti, l'aggiunta di funzionalità, etc..

11.4 Miglioramento continuo

Principio attorno al quale organizzare i processi per ottenere un miglioramento continuo, prevede 4 macro-fasi: *Plan (keep track of what you're going to do)*, *Do (as planned)*, *Check*, *Act (keep what works, throw what doesn't)*.

11.5 Minimizzazione

Chiedere a Giovanni!!!

11.6 Modello di ciclo di vita

Il ciclo di vita di un software non è univocamente determinato, ma ne vengono individuati diversi modelli possibili. La scelta del modello dipende da 3 macro-fattori: cosa vuole il committente, dipendenza da terze parti, livello di coinvolgimento del committente nell'accertamento dello stato di avanzamento.

11.6.1 Sequenziale (A cascata)

Ha per principio cardine la ripetibilità dei processi. Il ciclo di vita sequenziale è lineare, le fasi si susseguono, e la direzione ammessa è una sola. Il modello fa forte uso di documentazione il che rende il sistema organizzato e tracciabile, prevede pre e post per ogni fase e associa ad ogni fase date di inizio e fine. ISO 12207 definisce cosi' le fasi del ciclo di vita sequenziale: analisi, progettazione (intesa come *design*), realizzazione, manutenzione.

11.6.2 Incrementale

Prevede un approccio che fa uso di incrementi, ha come vantaggi:

- Il valore aggiunto di ogni incremento
- La riduzione del rischio di fallimento portata da ogni incremento
- L'uso di abilitatori che facilitano il lavoro

11.6.3 Evolutivo

Fa uso massiccio della fase di manutenzione viene fatta ad ogni versione rilevante del prodotto.

11.6.4 A componenti

Si sviluppa sull'idea di riutilizzare componenti . Prevede quindi tecniche di adattamento delle componenti e dei requisiti finalizzate al riuso del software.

11.6.5 A spirale

Modello di ciclo di vita utilizzato quando il progetto è innovativo e non esistono best practice applicabili per lo sviluppo del progetto.

11.6.6 Agili

Metodi che si rifanno a 4 principi fondamentali:

- Focus sugli *stakeholder* e le loro interazioni, piuttosto che su processi e strumenti;
- Focus sul software piuttosto che sulla documentazione;

- Rapporto collaborativo piuttosto che contrattuale con il cliente;
- Risposta veloce anche se non pianificata al cambiamento.

Tali metodi offrono degli svantaggi poiché la documentazione è cruciale nella fase di manutenzione del software e l'assenza di pianificazione in faccia ai cambiamenti comporta dei rischi.

11.6.7 Scrum

Scrum ha come focus fondamentale le *user-stories*: una visione di ciò che vuole il cliente, da esse si ricava un *product backlog*, insieme di attività/feature da svolgere/realizzare. L'attività di *Sprint Backlog* é un passo iterativo che impone controlli frequenti (*Daily Scrum*) sulla correttezza delle azioni intraprese, tramite *Sprint Review* e *Retrospective*.

12 N

12.1 Node.js

Ambiente run-time open source e multipiattaforma che permette di eseguire codice JavaScript al di fuori di un browser.

13 O

13.1 Obiettivo SMART

- Specific: formalmente definiti;
- Measurable: il cui stato di ottenimento è misurabile;
- Achievable: raggiungibili;
- Realistic: pratici (raggiungibili);
- Time-Bound: costretti da un vincolo temporale.

13.2 Open source

Tipo di prodotto software il cui codice sorgente viene rilascia al pubblico mediante una licenza che ne permette a chiunque lo studio, la modifica e la distribuzione p qualsiasi scopo.

14 P

14.1 Panel

Un panel è una rappresentazione visiva di dati elaborati da un sistema mostrati sotto forma di grafici.

14.2 PDCA

Acronimo di *Plan, Do, Check, Act*, anche noto come ciclo di Deming. Si tratta di un ciclo di miglioramento continuo dei processi e dei prodotti, suddiviso in quattro fasi, che sono appunto *Plan, Do, Check, Act*.

14.3 Performance

Prestazione o prestazioni, a seconda del contesto.

14.4 Perl

Perl è un linguaggio di programmazione ad alto livello, dinamico, procedurale e interpretato, creato nel 1987 da Larry Wall. Perl ha un singolare insieme di funzionalità ereditate da molti altri linguaggi di programmazione, compresi alcuni linguaggi funzionali.

14.5 Pianificazione delle attività

Pianificare le attività è uno dei compiti del project manager. La pianificazione include l'allineamento delle attività su un asse temporale, l'assegnazione delle attività a delle persone.
Il project manager si avvale di diversi strumenti per pianificare le attività:

- Diagrammi di Gantt (Rappresentano durata prevista vs durata effettiva delle attività);
- Diagrammi PERT (*Programme Evaluation and Review Technique*, tiene conto delle dipendenze tra attività e mostra lo slack);
- WBS (*Work Breakdown Structure*, divisione delle attività fino a raggiungere il compito minore assegnabile ad una persona. Diagramma che rappresenta l'assegnazione di tali attività alle persone nel tempo, rispettando e rappresentando dipendenze e vincoli temporali.).

14.6 Piano di progetto

Include:

- Introduzione (scopo, struttura);
- Organizzazione del progetto;
- Analisi dei rischi;
- Risorse disponibili (tempo, persone);

- Suddivisione del lavoro;
- Calendario delle attività;
- Meccanismi di controllo e rendicontazione.

14.7 Plug-in

Un plug-in in campo informatico è un programma non autonomo che interagisce con un altro programma per ampliarne o estenderne le funzionalità originarie. Ad esempio, un plug-in per un software di grafica permette l'utilizzo di nuove funzioni non presenti nel software principale.

14.8 Processo

Insieme di attività correlate (includono tutto ciò che è attinente) e coese (sono tutte necessarie) che trasformano ingressi (bisogni) in uscite (prodotti) secondo regole date dal controllo processo a seguito di decisioni prese sulla base di misurazioni delle risorse consumate dal processo. I processi si differenziano in base alla specificità dell'ambito di applicazione, possono essere standard (riferimento base generico), definito (standard adattato alle esigenze aziendali), di progetto (definito adattato al progetto).

14.9 Processo software

Insieme di attività che devono essere svolte per far avanzare un prodotto software nel suo ciclo di vita (nell'ambito SWE, vedi "Processo" per una voce più generale).

14.10 Processo specializzato per progetto

Prevedono una fase di pianificazione, una di definizione, attenzione nella conduzione e analisi critica del funzionamento del processo.

14.11 Prodotto software

Una "entità" software progettata per essere rilasciata ad un cliente. Prende forme diverse in base al soggetto richiedente:

14.11.1 Commessa

Si tratta di un prodotto software le cui specifiche ed obiettivi sono indicati da un committente.

14.11.2 Pacchetto

Si tratta di un prodotto software le cui specifiche ed obiettivi sono indicati per la replicazione in o per altri software.

14.11.3 Componente

Si tratta di un prodotto software le cui specifiche ed obiettivi sono indicati per la composizione con altri software.

14.11.4 Servizio

Si tratta di un prodotto software le cui specifiche ed obiettivi sono determinati per la risoluzione di un problema.

14.12 Prodotti documentali

- Capitolato d'appalto;
- Studio di fattibilità;
- Analisi dei requisiti;

14.13 Produttività

Una misurazione di efficienza, rapporto tra quantità di prodotto realizzato e risorse consumate.

14.14 Progettazione Software

Insieme di attività che precede la realizzazione il cui scopo è quello definire l'architettura del software con l'obiettivo di sviluppare un prodotto corretto per costruzione piuttosto che per correzione.

14.15 Progetto

Insieme di attività e compiti, atti a raggiungere obiettivi SMART, che hanno una data di inizio ed una di fine, e che consumano un pool di risorse limitate.

14.16 Project Manager

Individuo che ha il compito di gestione del progetto. Ciò prevede:

- Istanziare i processi nel progetto (quelli standard aziendali e quelli istanziati dai processi aziendali);
- Stimare costi e risorse necessarie;
- Pianificare attività (organizzarle nel tempo e a chi assegnarle);
- Controllare le attività e verificare i risultati.

14.17 Prototipo

Esemplare di prova del prodotto, serve a provare e scegliere soluzioni; può avere carattere usa e getta (in metodi iterativi) oppure essere la base per incrementi successivi (in metodi incrementali). Un prototipo di tipo usa e getta comporta dei costi che producono poco valore aggiunto.

14.18 Publish/subscribe

In ambito informatico, design pattern utilizzato per la comunicazione asincrona tra diversi processi, oggetti o altri agenti.

14.19 Python

Linguaggio di programmazione ad alto livello, interpretato, ha come scopo la buona leggibilità del codice. Questa viene garantita usando la spaziatura come elemento fondamentale di sintassi. Notoriamente meno efficiente di altri linguaggi, ma di facile comprensione e apprendimento.

15 Q

15.1 Quantificabile

Misurabile.

15.2 Qualità

Misura in cui un prodotto software soddisfa un certo numero di aspettative rispetto sia al suo funzionamento sia alla sua struttura interna.

15.2.1 Qualità del prodotto

Caratteristiche di un prodotto di qualità:

- Sufficienza: Capacità di soddisfare tutti i requisiti;
- Comprensibilità: Facilità di utilizzo, comprensione del funzionamento da parte degli *stakeholder*;
- Modularità: Suddivisione in parti chiaramente distinte e sconnesse, principio fondamentale dell' *information hiding.*;
- Robustezza: Agli input di utilizzo;
- Sicurezza: Affidabilità in caso di malfunzionamento (hardware o software);
- Sicurezza bis: Resistenza alla intrusioni;
- Flessibilità: Misura della facilità con la quale il sistema può essere modificato o adattato per soddisfare nuovi requisiti;
- Riusabilità: Misura della facilità di riutilizzo in altri contesti dei moduli che formano il prodotto;
- Disponibilità: Più inconveniente il tempo di *downtime* in caso di guasti o aggiornamenti, meno disponibile risulta il prodotto;
- Efficienza;

15.2.2 Qualità dei processi

Una buona qualità dei processi induce una buona qualità del prodotto di questi. Per assicurarsi che un processo sia di qualità, è necessario un meccanismo di controllo che effettui modifiche sul way of working ove ritenuto necessario sulla base di misurazioni sulla qualità dei processi. Per definire gli strumenti di valutazione della qualità dei processi dei fornitori in genere, non solo in ambito di sviluppo software, è stato messo a punto lo standard ISO 9000, in ambito specifico di ingegneria software si ha invece ISO 90003:2004.

16 R

16.1 Raintank

Raintank è una piattaforma di monitoraggio open source per raccogliere, archiviare e analizzare enormi quantità di dati diagnostici, da un'ampia varietà di fonti, in molti formati diversi.

16.2 Redmine

Applicazione web gratuita e open source per attività di *project management* e *issue tracking*.

16.3 Repository

Termine inglese che significa "deposito", "ripostiglio". In ambito informatico viene usato per indicare una struttura dati in cui vengono gestiti i metadati per un insieme di file o una strutture di cartelle. Esempi di metadati gestiti da una repository sono lo storico dei cambiamenti avvenuti, o l'insieme dei commit (modifiche).

16.4 Requisito

I requisiti sono le capacità che il sistema dovrà avere per svolgere le funzioni volute dal committente. È best practice definire requisiti:

- Non ambigui: Interpretati univocamente da tutti .
- Corretti.
- Completi.
- Variabili.
- Consistenti: che non si contraddicono a vicenda o da sè.
- Modificabili.
- Tracciabili
- Ordinati: tra obbligatori, desiderabili ed opzionali.

16.5 Rete Bayesiana

Una rete Bayesiana è un modello grafico probabilistico che rappresenta un insieme di variabili stocastiche con le loro dipendenze condizionali attraverso l'uso di un grafo. Per esempio, una rete Bayesiana potrebbe rappresentare la relazione probabilistica esistente tra dei sintomi e delle malattie. Dati i sintomi, la rete può essere usata per calcolare la probabilità della presenza di diverse malattie.

16.6 Rischio

I rischi di progetto sono: sforamento costi, tempi, o risultati insoddisfacenti. Fonti di rischio principali sono: tecnologie usate, rapporti interpersonali, organizzazione del lavoro, requisiti e rapporti con *stakeholder*, tempi e costi. La gestione del rischio si fa tramite: Identificazione (nel progetto, prodotto, mercato), Analisi (probabilità e conseguenze), Pianificazione (come evitare e o mitigare gli effetti), Controllo(attenzione agli indicatori di rischio, raffinamento strategie).

16.7 Riuso

Del codice, può essere di due tipi: opportunistico (copia e incolla), che ha un basso costo e poco impatto, o sistematico, che ha un maggior costo e un maggior impatto.

16.8 Ruolo

Indica l'area di specializzazione. In un progetto informatico i ruoli principali sono:

- Analista: Si occupa dell'analisi dei requisiti, in generale: individua e definisce il "problema" in termini formali (in modo che sia possibile verificare se la soluzione è tale);
- Progettista: Si occupa del design del prodotto. In generale: si occupa della soluzione del "problema";
- Programmatore: Realizza il prodotto, non ha libertà di scelta, segue le direttive del progettista;
- Verificatore: Testano la qualità del prodotto, danno feedback;
- Amministratore: Garantisce il funzionamento dell'apparato informatico aziendale.

Tutte queste figure professionali sono coordinate nell'ambito di un progetto da un project manager.

17 S

17.1 Script

Il termine script, in informatica, designa un tipo particolare di programma, scritto in una particolare classe di linguaggi di programmazione, detti linguaggi di scripting.

17.2 Sistematico

Metodico, rigoroso.

17.3 Slack (tempo)

Margine di tempo tra la fine di un attività e la scadenza di fine attività.
Se è positivo, l'attività è prevista terminare prima della scadenza quindi vi è un margine per eventuali imprevisti. Se è vicino o 0 tale margine è piccolo o assente, si introducono quindi dei rischi. Se è negativo, la scadenza è passata e l'attività non è stata portata al termine.

17.4 Slack (software)

Insieme di strumenti Cloud per la comunicazione e la collaborazione in un team. Il nome è un acronimo per "*Searchable Log of All Conversation and Knowledge*".

17.5 SonarQube

Piattaforma open source per ispezione continua della qualità del codice tramite analisi statica.

17.6 Software Engineering

Disciplina il cui scopo è la realizzazione di prodotti software; nota anche con l'acronimo SWE. SWE si occupa dell'organizzazione e della gestione delle attività, compiti e interazioni di un team di sviluppatori, le quali hanno come obiettivo per l'attività di sviluppo e il software risultate Efficacia ed Efficienza. SWE si occupa inoltre delle metodologie di cura del progetto per l'intero ciclo di vita del software. SWE prevede un approccio sistematico, disciplinato e quantificabile all'attività di sviluppo.

17.7 Stima dei costi

Influenzata dalle dimensioni del progetto, dalle esperienze pregresse, dalla familiarità con le tecnologie adottate, dalla produttività dell'ambiente di lavoro, dalla qualità attesa.
Un modello algoritmico che può aiutare in ciò è il CoCoMo.

17.8 Studio di fattibilità

Studio economico realizzato per determinare se è vantaggioso partecipare ad una gara d'appalto.

18 T

18.1 Telegram

Servizio Cloud per VoIP e messaggistica istantanea.

18.2 Tracciamento dei requisiti

Attività di monitoraggio dell'evoluzione e della scoperta dei requisiti che possono cambiare durante lo svolgersi dello sviluppo del progetto.Tale attività viene svolta tramite apparati informatici appositi, le informazioni importanti sui requisiti di cui viene tenuta traccia sono: *status*, *origin*, assegnatari.

19 V

19.1 Validare

Attività di confronto tra i risultati ottenuti e quelli aspettati. Quest'attività si svolge al termine del progetto.

19.2 Verificare

Attività di verifica di presenza di errori e di rispetto del way of working. La verifica si manifesta tangibilmente sotto forma di test, effettuati sui moduli che compongono il prodotto software. Le forme di verifica sono 2: analisi dinamica e analisi statica. L'analisi dinamica prevede l'esecuzione del software mentre l'analisi statica prevede un analisi del codice sorgente senza eseguirlo.

19.3 Verificatore di requisiti

Individuo che ha il compito di verificare i requisiti, utilizza le tecniche descritte in analisi statica.

19.4 Versionamento

Organizzazione e gestione delle versioni del software prodotte nel corso dello sviluppo.

19.5 Versione

Vedi baseline.

20 W

20.1 Way of working

Insieme di regole, pratiche, che caratterizzano l'attività di sviluppo. Il way of working è pensato per agevolare lo sviluppo ed evitare sprechi di risorse ed errori.

21 Z

21.1 Zero-latency

Riferito al lavoro: senza ritardo. L'approccio prevede l'inizio della attività di sviluppo non appena possibile. Permette di avere un margine di tempo tra termine del progetto e consegna.

21.2 Zero-laxity

Riferito al lavoro: senza margine. L'approccio prevede l'inizio delle attività di sviluppo nel momento in cui la data di consegna del progetto meno il tempo richiesto dallo sviluppo equivale zero o meno.