AtAVi

Piano di qualifica v1.0.0

Sommario
Documento contenente le strategie adottate dal gruppo Co.Code per garantire la qualità del *prodotto*$_g$ AtAVi.

Versione	1.0.0
Data di redazione	2017-01-08
Redazione	Nicola Tintorri
	Andrea Magnan
Verifica	Pier Paolo Tricomi
Approvazione	Luca Bertolini
Uso	Esterno
Distribuzione	prof. Tullio Vardanega
	prof. Riccardo Cardin
	Zero12

Diario delle modifiche

Versione	Riepilogo	Autore	Ruolo	Data
1.0.0	Approvazione	Luca Bertolini	*Responsabile*	2017-01-08
0.2.2	Inseriti dati resoconto fase AR	Andrea Magnan	*Amministratore*	2017-01-07
0.2.1	Correzione sezioni segnalate dalla verifica	Andrea Magnan	*Amministratore*	2016-12-24
0.2.0	Verifica	Pier Paolo Tricomi	*Verificatore*	2016-12-24
0.1.5	Aggiunta struttura resoconto fase AR	Andrea Magnan	*Amministratore*	2016-12-23
0.1.4	Aggiunta appendice A	Nicola Tintorri	*Amministratore*	2016-12-23
0.1.3	Conclusa stesura sezione 3	Nicola Tintorri	*Amministratore*	2016-12-22
0.1.2	Aggiunte appendici B	Andrea Magnan	*Amministratore*	2016-12-22
0.1.1	Correzione sezioni segnalate dalla verifica	Andrea Magnan	*Amministratore*	2016-12-22
0.1.0	Verifica	Pier Paolo Tricomi	*Verificatore*	2016-12-22
0.0.5	Inizio stesura sezione 3	Nicola Tintorri	*Amministratore*	2016-12-21
0.0.4	Conclusa stesura sezione 2	Andrea Magnan	*Amministratore*	2016-12-21
0.0.3	Inizio stesura sezione 2	Andrea Magnan	*Amministratore*	2016-12-20
0.0.2	Stesura introduzione	Nicola Tintorri	*Amministratore*	2016-12-19
0.0.1	Inizio stesura documento	Nicola Tintorri	*Amministratore*	2016-12-19

Indice

Elenco delle tabelle

1 Introduzione

1.1 Scopo del documento

Il documento ha lo scopo di definire gli obiettivi di qualità e le strategie che il gruppo Co.Code adotterà per raggiungerli. Verrà inoltre illustrato come il gruppo affronterà le varie fasi di verifica per poter garantire il miglior risultato qualitativo possibile.

1.2 Scopo del prodotto

Si vuole creare un'applicazione web che permetta ad un ospite, in visita all'ufficio di Zero12, di interrogare un assistente virtuale per annunciare la propria presenza, avvisare l'interessato del suo arrivo sul sistema di comunicazione aziendale (*Slack*$_g$) e nel frattempo essere intrattenuto con varie attività.

1.3 Glossario

Allo scopo di evitare ogni ambiguità nel linguaggio e rendere più semplice e chiara la comprensione dei documenti, viene allegato il "*Glossario v1.0.0*". Le parole in esso contenute sono scritte in corsivo e marcate con una 'g' a pedice (p.es. *Parola*$_g$).

1.4 Riferimenti

1.4.1 Normativi

- Norme di progetto: "*Norme di Progetto v1.0.0*";

1.4.2 Informativi

- Piano di progetto: "*Piano di Progetto v1.0.0*";
- Slide del corso di Ingegneria del software - Qualità del software : http://www.math.unipd.it/~tullio/IS-1/2016/Dispense/L10.pdf;
- Slide del corso di Ingegneria del software - Qualità di processo : http://www.math.unipd.it/~tullio/IS-1/2016/Dispense/L11.pdf;
- Slide del corso di Ingegneria del software - Analisi dinamica : http://www.math.unipd.it/~tullio/IS-1/2016/Dispense/L14.pdf;
- Indice Gulpease: https://it.wikipedia.org/wiki/Indice_Gulpease;
- Standard ISO/IEC 9126:2001;
- Standard ISO/IEC 12207:2008;
- Capability Maturity Model (CMM): https://en.wikipedia.org/wiki/Capability_Maturity_Model;
- Plan-Do-Check-Act (PDCA): https://en.wikipedia.org/wiki/PDCA.

2 Qualità di processo

Da processi scadenti derivano prodotti scadenti. La qualità di processo è quindi un fattore indispensabile per garantire la qualità dei prodotti. Assicurarla, inoltre, permette di:

- favorire l'ottimizzazione delle risorse;
- migliorare la stima dei rischi;
- ridurre i costi.

Per garantire la qualità di processo abbiamo deciso di adottare il Capability Maturity Model (CMM) che definisce una scala, suddivisa in cinque livelli, per misurarne la maturità.
Le misurazioni ottenute vengono utilizzate all'interno della strategia di miglioramento continuo della qualità, realizzata tramite Plan-Do-Check-Act (PDCA).
Per maggiori informazioni su CMM e PDCA consultare le rispettive appendici A e B.
Inoltre, sono stati individuati dallo standard ISO/IEC 12207:2008 i processi ritenuti più importanti nel ciclo di vita del prodotto. Per ciascuno di essi sono stati individuati gli obiettivi di qualità e i rispettivi intervalli di accettabilità e ottimalità.
Per quantificare gli obiettivi di qualità vengono utilizzate delle metriche, descritte nel documento "*Norme di Progetto v2.0.0*".
Viene assegnato un codice identificativo ad ogni obiettivo al fine di semplificarne il tracciamento. Il metodo di denominazione è descritto nel documento "*Norme di Progetto v2.0.0*".

2.1 Infrastructure Management Process

Questo processo si occupa di mantenere, monitorare e modificare l'infrastruttura per assicurare che essa continui ad eseguire i servizi necessari allo svolgimento del progetto. Con infrastruttura si intendono elementi hardware, software, metodi, strumenti, tecniche e standard impiegati nello sviluppo del prodotto.

2.1.1 Strategie

Per tutto l'arco del progetto l'infrastruttura dovrà possedere le seguenti caratteristiche:

- tutti gli strumenti e le procedure per utilizzarli verranno descritti esaustivamente nel documento "*Norme di Progetto v2.0.0*";
- la piattaforma PragmaDB sarà disponibile ogni qual volta un componente del gruppo ne richiederà l'accesso;
- i dati contenuti in PragmaDB saranno sempre coerenti e aggiornati;
- la piattaforma PragmaDB estenderà le proprie funzionalità in base alle esigenze del progetto;

2.1.2 Obiettivi di qualità

2.1.2.1 Disponibilità PragmaDB - OPC1

Indica la disponibilità di utilizzo della piattaforma di PragmaDB rispetto alla richiesta. Questo valore viene espresso in percentuale.

- **Metrica utilizzata**: percentuale di accessi avvenuti correttamente a PragmaDB (MPC1);
- **Soglia di accettabilità**: $80\% \leq X \leq 100\%$;
- **Soglia di ottimalità**: $90\% \leq X \leq 100\%$;

2.2 Project Planning, Assessment & Control Process

Questo processo (derivante dall'unione dei processi Project Planning Process e Project Assessment & Control Process) si occupa del necessario per la pianificazione del progetto il quale deve contenere la scelta del modello del ciclo di vita del prodotto, la pianificazione dei tempi e costi da sostenere, la descrizione dei compiti e delle attività associate, l'allocazione dei compiti e delle responsabilità e le metriche atte a misurare lo stato del progetto rispetto la pianificazione.

2.2.1 Strategie

Lo sviluppo del progetto dovrà seguire la pianificazione, in particolare:

- il progetto dovrebbe rispettare i tempi indicati nel documento "*Piano di Progetto v2.0.0*";
- il progetto dovrebbe rispettare i costi indicati nel documento "*Piano di Progetto v2.0.0*";
- ruoli e compiti verranno descritti esaustivamente nel documento "*Piano di Progetto v2.0.0*".

2.2.2 Obiettivi di qualità

2.2.2.1 Rispetto dei tempi - OPC2

Indica se i tempi pianificati sono stati rispettati.

- **Metrica utilizzata**: Schedule Variance (MPC2);
- **Soglia di accettabilità**: $\geq -5\%$;
- **Soglia di ottimalità**: $\geq 0\%$.

2.2.2.2 Rispetto dei costi - OPC3

Indica se i costi pianificati, in data corrente, sono rispettati.

- **Metrica utilizzata**: Cost Variance percentuale (MPC3);
- **Soglia di accettabilità**: $\geq -10\%$;
- **Soglia di ottimalità**: $\geq 0\%$.

Eventuali costi non accettabili dovranno essere compensati entro la fine dell'attività di progetto in quanto non è permesso eccedere i costi preventivati oltre la soglia definita.

2.3 Risk Management Process

Questo processo identifica, analizza, tratta e monitora continuamente i rischi che possono sorgere nel ciclo di vita del progetto.

2.3.1 Strategie

Il gruppo dovrà gestire correttamente i rischi, in particolare:

- all'inizio della fase di progetto, i rischi devono essere individuati e descritti nel documento "*Piano di Progetto v2.0.0*";
- per ogni rischio, devono essere definite strategie per il riconoscimento e il trattamento;
- all'inizio di ogni periodo, l'analisi dei rischi permetterà l'individuazione di eventuali nuovi rischi;

- il livello di probabilità che i rischi si presentino dovrà sempre essere tenuto sotto controllo.

2.3.2 Obiettivi di qualità

2.3.2.1 Rischi non preventivati - OPC4

Evidenzia il numero dei rischi presentati e non preventivati nell'arco del progetto; un valore molto alto potrebbe indicare una povera analisi dei rischi.

- **Metrica utilizzata**: numero dei rischi non preventivati (MPC4);
- **Soglia di accettabilità**: 0 - 3;
- **Soglia di ottimalità**: 0.

2.4 System/Software Requirements Analysis Process

Questo processo si occupa di trasformare le idee del proponente in un insieme di requisiti tecnici atti a guidare la progettazione del sistema.

2.4.1 Strategie

I requisiti identificati dal gruppo e successivamente inseriti nel documento "*Analisi dei Requisiti v2.0.0*" dovranno possedere le seguenti caratteristiche:

- dovranno essere inseriti nella piattaforma PragmaDB;
- si dovrà tenere traccia delle fonti da cui sono stati ricavati;
- si dovrà tenere traccia della loro implementazione;
- dovranno essere approvati dal proponente;
- nessun requisito dovrà risultare superfluo o ambiguo.

2.4.2 Obiettivi di qualità

2.4.2.1 Requisiti obbligatori soddisfatti - OPC5

Indica il numero dei requisiti obbligatori soddisfatti, espresso in percentuale.

- **Metrica utilizzata**: Numero dei requisiti obbligatori soddisfatti (MPC5);
- **Soglia di accettabilità**: 100%;
- **Soglia di ottimalità**: 100%.

2.4.2.2 Requisiti desiderabili soddisfatti - OPC6

Indica il numero dei requisiti desiderabili soddisfatti, espresso in percentuale.

- **Metrica utilizzata**: Numero dei requisiti desiderabili soddisfatti (MPC6);
- **Soglia di accettabilità**: 70%;
- **Soglia di ottimalità**: 100%.

2.4.2.3 requisiti facoltativi soddisfatti - OPC7

Indica il numero dei requisiti facoltativi soddisfatti, espresso in percentuale.

- **Metrica utilizzata**: Numero dei requisiti facoltativi soddisfatti (MPC7);
- **Soglia di accettabilità**: 0%;
- **Soglia di ottimalità**: 100%.

2.5 System/Software Architectural Design Process

Questo processo si occupa di identificare la corrispondenza tra requisiti di sistema ed elementi del sistema.

2.5.1 Strategie

L'architettura ottenuta svolgendo le attività di questo processo dovrà possedere le seguenti caratteristiche:

- il sistema dovrà presentare basso accoppiamento ed alta coesione;
- ogni componente dovrà essere progettato puntando su incapsulamento, modularizzazione e riuso di codice;
- per ogni elemento del sistema dovrà essere possibile tracciare il requisito associato.

2.5.2 Obiettivi di qualità

2.5.2.1 Structural Fan-In - OPC8

In riferimento ad un modulo del software, indica quanti altri moduli lo utilizzano durante la loro esecuzione; tale indicazione permette di stabilire il livello di riuso implementato.

- **Metrica utilizzata**: SF-IN (MPC8);
- **Soglia di accettabilità**: ≥ 0;
- **Soglia di ottimalità**: ≥ 2;

2.5.2.2 Structural Fan-Out - OPC9

In riferimento ad un modulo del software, indica quanti moduli vengono utilizzati durante la sua esecuzione; tale indicazione permette di stabilire il livello di accoppiamento implementato.

- **Metrica utilizzata**: SF-OUT (MPC9);
- **Soglia di accettabilità**: 0 - 6;
- **Soglia di ottimalità**: 0 - 2.

2.5.2.3 Design Structure Quality Index - OPC10

2.6 Software Detailed Design Process

Questo processo si occupa di fornire, dato un sistema, una sua progettazione di dettaglio che permetta codifica ed esecuzione di test.

2.6.1 Strategie

Le attività di questo processo dovranno possedere le seguenti caratteristiche:

- il livello di dettaglio della progettazione dovrà guidare la codifica e l'esecuzione dei test senza bisogno di informazioni aggiuntive;
- la progettazione di dettaglio, che include architettura di basso livello, relazioni fra le varie unità software concepite e la definizione dettagliata delle interfacce dovrà essere esposta chiaramente nel documento "*Definizione di Prodotto v1.0.0*";
- per ogni elemento dell'architettura a basso livello dovrà essere possibile tracciare il requisito associato.

2.6.2 Obiettivi di qualità

2.6.2.1 Numero di metodi per classe - OPC11

Indica il numero di metodi definiti in una classe; un valore molto alto potrebbe indicare una cattiva decomposizione delle funzionalità a livello di progettazione.

- **Metrica utilizzata**: numero di metodi per classe (MPC11);
- **Soglia di accettabilità**: 1 - 10;
- **Soglia di ottimalità**: 1 - 8.

2.6.2.2 Numero di parametri per metodo - OPC12

Indica il numero di parametri passati ad un metodo; un valore molto alto potrebbe indicare un metodo troppo complesso e ulteriormente scomponibile in metodi più semplici.

- **Metrica utilizzata**: numero di parametri per metodo (MPC12);
- **Soglia di accettabilità**: 0 - 6;
- **Soglia di ottimalità**: 0 - 4.

2.7 Software Construction Process

Questo processo si occupa di produrre unità di software eseguibili che riflettono la progettazione effettuata.

2.7.1 Strategie

Le unità software prodotte dovranno rispettare le seguenti caratteristiche:

- il codice dovrà risultare facilmente manutenibile;
- il codice prodotto dovrà essere facilmente comprensibile e testabile;
- per ogni unità di software dovrà essere possibile tracciare il requisito e l'elemento architetturale associato.

2.7.2 Obiettivi di qualità

2.7.2.1 Complessità ciclomatica - OPC13

Indica la complessità di funzioni, moduli, metodi o classi di un programma. Alti valori di complessità ciclomatica implicano una ridotta manutenibilità del codice.

- **Metrica utilizzata**: indice di complessità ciclomatica (MPC13);
- **Soglia di accettabilità**: 1 - 20;
- **Soglia di ottimalità**: 1 - 10.

2.7.2.2 Livelli di annidamento - OPC14

Indica il numero di procedure e funzioni annidate, ovvero richiamate all'interno di altre procedure o funzioni. Più livelli di annidamento potrebbero rendere il codice di difficile comprensione e potrebbero portare a commettere errori logici durante la realizzazione del codice. In caso di troppi livelli di annidamento sarebbe opportuno riscrivere il metodo, o la funzione, affinchè sia facilmente comprensibile.

- **Metrica utilizzata**: numero di livelli di annidamento (MPC14);
- **Soglia di accettabilità**: 1 - ;
- **Soglia di ottimalità**: 1 - 2.

2.7.2.3 Linee di commento per linee di codice - OPC15

Indica il numero di linee di commento rispetto alle linee totali del codice. Un valore basso indica un codice poco comprensibile e, di conseguenza, difficilmente manutenibile. Un valore troppo alto indica un eccesso di commenti e un appesantimento dei file.

- **Metrica utilizzata**: percentuale linee di commento per linee di codice (MPC15);
- **Soglia di accettabilità**: 10% - 40%;
- **Soglia di ottimalità**: 20% - 30%.

2.7.2.4 Complessità di una funzione - OPC16

Indica il livello di complessità di una funzione, ovvero la difficoltà della funzione di essere scritta o compresa.

- **Metrica utilizzata**: Halstead Difficulty per funzione (MPC16);
- **Soglia di accettabilità**: 0 - 25;
- **Soglia di ottimalità**: 0 - 15.

2.7.2.5 Dimensione dell'implementazione di una funzione - OPC17

Indica la dimensione dell'implementazione di una funzione.

- **Metrica utilizzata**: Halstead Volume per funzione (MPC17);
- **Soglia di accettabilità**: 20 - 1500;
- **Soglia di ottimalità**: 20 - 1000.

2.7.2.6 Costo di una funzione - OPC18

Rappresenta il costo per scrivere il codice di una funzione.

- **Metrica utilizzata**: Halstead Effort per funzione (MPC18);
- **Soglia di accettabilità**: 0 - 400;
- **Soglia di ottimalità**: 0 - 300.

2.7.2.7 Manutenibilità - OPC19

Permette di stabilire il grado di manutenibilità del codice prodotto.

- **Metrica utilizzata**: indice di manutenibilità (MPC19);
- **Soglia di accettabilità**: 10 - 100;
- **Soglia di ottimalità**: 20 - 100.

2.8 System/Software Integration Process

Questo processo si occupa di integrare le unità software tra loro, produrre software coerente con la progettazione e dimostrare che il prodotto soddisfi i requisiti identificati.

2.8.1 Strategie

Le attività previste da questo processo dovranno puntare a raggiungere un alto livello di automazione, in particolare:

- l'integrazione delle varie parti del sistema sarà completamente automatizzata utilizzando lo strumento di continuous integration Jenkins, come definito nel documento "*Norme di Progetto v2.0.0*";
- il livello di integrazione raggiunto del sistema sarà sempre consultabile grazie all'utilizzo dello strumento di continuous integration Jenkins, come definito nel documento "*Norme di Progetto v2.0.0*".

2.8.2 Obiettivi di qualità

2.8.2.1 Componenti integrate - OPC20

Indica il numero di componenti definite in progettazione che sono attualmente implementate e integrate nel sistema. Nel nostro caso, tutte le componenti progettate andranno a costituire il sistema.

- **Metrica utilizzata**: percentuale di componenti integrate nel sistema (MPC20);
- **Soglia di accettabilità**: 100%;
- **Soglia di ottimalità**: 100%.

2.9 System/Software Qualification Testing Process

Questo processo si occupa di assicurare che ogni requisito individuato sia stato implementato nel prodotto.

2.9.1 Strategie

Questo processo deve possedere le seguenti caratteristiche:

- le attività di test previste dal processo verranno svolte su un sistema le cui componenti sono verificate e correttamente integrate fra loro;
- le attività di test dovranno raggiungere il maggior livello di automazione nell'esecuzione tramite lo strumento di continuous integration Jenkins;
- le attività di test dovranno essere eseguite in numero sufficiente in modo tale da garantire un'ottima copertura dei requisiti;
- il software dovrà implementare tutti i requisiti obbligatori.

2.9.2 Obiettivi di qualità

2.9.2.1 Test di unità eseguiti - OPC21

Indica il numero di test di unità eseguiti tra quelli definiti dal gruppo.

- **Metrica utilizzata**: percentuale di test di unità eseguiti (MPC21);
- **Soglia di accettabilità**: 90% - 100%;
- **Soglia di ottimalità**: 100%.

2.9.2.2 Test di integrazione eseguiti - OPC22

Indica il numero di test di integrazione eseguiti tra quelli definiti dal gruppo.

- **Metrica utilizzata**: percentuale di test di integrazione eseguiti (MPC22);
- **Soglia di accettabilità**: 70% - 100%;
- **Soglia di ottimalità**: 80% - 100%.

2.9.2.3 Test di sistema eseguiti - OPC23

Indica il numero di test di sistema eseguiti in modo automatico tra quelli definiti dal gruppo.

- **Metrica utilizzata**: percentuale di test di sistema eseguiti (MPC23);
- **Soglia di accettabilità**: 70% - 100%;
- **Soglia di ottimalità**: 80% - 100%.

2.9.2.4 Test di validazione eseguiti - OPC24

Indica il numero di test di validazione eseguiti manualmente tra quelli definiti dal gruppo.

- **Metrica utilizzata**: percentuale di test di validazione eseguiti (MPC24);
- **Soglia di accettabilità**: 100%;
- **Soglia di ottimalità**: 100%.

2.9.2.5 Test superati - OPC25

Indica il numero di test superati.

- **Metrica utilizzata**: percentuale dei test superati (MPC25);
- **Soglia di accettabilità**: 90% - 100%;
- **Soglia di ottimalità**: 100%.

2.10 Software Verification Process

Questo processo si occupa di controllare che il software prodotto soddisfi correttamente i requisiti identificati.

2.10.1 Strategie

Questo processo deve possedere le seguenti caratteristiche:

- la documentazione verrà verificata mediante inspenction;
- i test dinamici sui vari elementi saranno al più possibile automatizzabili;
- i test dinamici sui vari elementi del software copriranno gran parte delle possibili casistiche di utilizzo;
- l'eisto di ogni test deve essere tracciabile.

2.10.2 Obiettivi di qualità

2.10.2.1 Branch coverage - OPC26

Indica il numero di rami decisionali percorsi nei test utilizzati. Questo obiettivo assicura che i branch derivanti da una condizione siano eseguiti da almeno un test.

- **Metrica utilizzata**: percentuale di rami decisionali percorsi (MPC26);
- **Soglia di accettabilità**: 75% - 100%;
- **Soglia di ottimalità**: 85% - 100%.

2.10.2.2 Function coverage - OPC27

Indica il numero di funzioni che sono state chiamate nei test utilizzati.

- **Metrica utilizzata**: numero di funzioni chiamate nei test (MPC27);
- **Soglia di accettabilità**: 70% - 100%;
- **Soglia di ottimalità**: 80% - 100%.

2.10.2.3 Statement coverage - OPC28

Indica il numero di istruzioni che sono state eseguite nei test utilizzati. Maggiore è il valore maggiore sarà il numero di statement eseguiti almeno una volta dai test.

- **Metrica utilizzata**: numero di istruzioni nei test (MPC28);
- **Soglia di accettabilità**: 75% - 100%;

- **Soglia di ottimalità**: 85% - 100%.

3 Qualità di prodotto

È prevista la realizzazione di due tipologie di prodotto: documenti e software. Per ciascuno di essi sono stati individuati degli obiettivi di qualità e i rispettivi intervalli di accettabilità e ottimalità. Per quantificare gli obiettivi di qualità vengono utilizzate delle metriche, descritte nel documento "*Norme di Progetto v2.0.0*".
Viene assegnato un codice identificativo ad ogni obiettivo al fine di semplificarne il tracciamento. Il metodo di denominazione è descritto nel documento "*Norme di Progetto v2.0.0*".

3.1 Documenti

I documenti prodotti dal gruppo Co.Code si dividono in interni ed esterni. I primi definiscono le strategie, gli strumenti e il metodo di lavoro (ways of working) del gruppo affinchè i membri realizzino prodotti simili tra loro secondo delle regole definite. I secondi definiscono tutto ciò che riguarda il software prodotto, partendo dalla progettazione fino a giungere all'analisi dei requisiti e la definizione di prodotto.
Poichè i documenti interni devono essere letti e compresi da tutti i membri del gruppo e quelli esterni devono essere comprensibili affinchè proponente e committente siano informati correttamente, il gruppo ha deciso di perseguire le strategie e gli obiettivi di qualità definiti di seguito.

3.1.1 Strategie

Tutti i documenti prodotti devono avere le seguenti caratteristiche:

- devono essere comprensibili da utenti con almeno la licenza superiore;
- i termini con significato ambiguo o poco chiaro dovranno essere inseriti nel "*Glossario v2.0.0*";
- saranno sempre aggiornati e allineati allo stato attuale del processo di sviluppo;
- dovranno essere dotati di numero di versione e diario delle modifiche.

3.1.2 Obiettivi di qualità - OPDD1

3.1.2.1 Leggibilità e comprensibilità

Indica il livello di leggibilità e comprensibilità del documento. Maggiore è il valore dell'indice maggiore sarà la leggibilità del documento.

- **Metrica utilizzata**: indice Gulpease (MPDD1);
- **Soglia di accettabilità**: 40 - 100;
- **Soglia di ottimalità**: 60 - 100.

3.2 Software

Sono state individuate dallo standard ISO/IEC 9126:2001 le principali caratteristiche che il software deve soddisfare. Sulla base di esse sono state definiti gl obiettivi di qualità e i relativi intervalli.

3.2.1 Functionality

È la capacità del prodotto software di fornire le funzionalità definite nei requisiti individuati nel documento "*Analisi dei Requisiti v2.0.0*".

3.2.1.1 Strategie

Il software deve soddisfare le seguenti caratteristiche:

- **Suitability**: fornire un appropriato insieme di funzionalità in base alle richieste dell'utente;
- **Accuracy**: fornire i corretti risultati con un adeguato grado di precisione;
- **Security**: proteggere le informazioni e i dati affinchè solo gli utenti autorizzati possano modificarli e/o leggerli.

3.2.1.2 Obiettivi di qualità

3.2.1.2.1 Implementazione funzionale - OPDS1

Indica quanti requisiti funzionali sono stati implementati.

- **Metrica utilizzata**: completezza dell'implementazione funzionale (MPDS1);
- **Soglia di accettabilità**: 100%;
- **Soglia di ottimalità**: 100%.

3.2.1.2.2 Accuratezza rispetto alle attese - OPDS2

Indica quanti risultati sono concordi alle attese.

- **Metrica utilizzata**: percentuale risultati concordi alle attese (MPDS2);
- **Soglia di accettabilità**: 90% - 100%;
- **Soglia di ottimalità**: 100%.

3.2.1.2.3 Controllo degli accessi - OPDS3

Indica quante operazioni illegali non sono state bloccate. Valori grandi indicano un sistema poco sicuro e facilmenete violabile. Valori bassi sono d'obbligo per poter garantire la sicurezza dei dati.

- **Metrica utilizzata**: percentuale operazioni illegali non bloccate (MPDS3);
- **Soglia di accettabilità**: 0% - 10%;
- **Soglia di ottimalità**: 0%.

3.2.2 Reliability

È la capacità del prodotto software di svolgere correttamente le sue funzioni in qualunque situazioni, anche in caso di situazioni anomale.

3.2.2.1 Strategie

Il software deve soddisfare le seguenti caratteristiche:

- **Maturity**: evitare che si verifichino malfunzionamenti, operazioni illegali e risultati errati in seguito ad errori;
- **Fault tolerance**: mantenere un certo livello di performance nonostante siano presenti errori e guasti o come conseguenza di un uso scorretto dell'applicativo.

3.2.2.2 Obiettivi di qualità

3.2.2.2.1 Densità di failure - OPDS4

Indica quante operazioni di testing sono concluse in failure.

- **Metrica utilizzata**: percentuale failure su test (MPDS4);
- **Soglia di accettabilità**: 0% - 10%;
- **Soglia di ottimalità**: 0%.

3.2.2.2.2 Blocco di operazioni non corrette - OPDS5

Indica quante funzionalità sono in grado di gestire correttamente gli errori che potrebbero verificarsi. Un valore alto è sinonimo di robustezza.

- **Metrica utilizzata**: numero di failure evitati (MPDS5);
- **Soglia di accettabilità**: 90% - 100%;
- **Soglia di ottimalità**: 100%;

3.2.3 Usability

È la capacità del prodotto software di essere capito, compreso ed usato dall'utente.

3.2.3.1 Strategie

Il software deve soddisfare le seguenti caratteristiche:

- **Understandability**: permettere all'utente di capire il grado di conformità del software e il suo dominio applicativo;
- **Learnability**: permettere all'utente di capire come utilizzarlo;
- **Operability**: permettere all'utente di utilizzarlo e controllarlo;
- **Attractiveness**: essere piacevole all'utente che lo utilizza.

3.2.3.2 Obiettivi di qualità

3.2.3.2.1 Comprensibilità delle funzioni offerte - OPDS6

Indica quante funzionalità sono state comprese immediatamente dall'utente senza la consultazione del manuale. L'assistente virtuale dovrebbe essere il più user friendly possibile. Infatti, l'interazione dell'utente dovrebbe essere il più fluida possibile senza dover consultare il manuale ad ogni operazione da eseguire.

- **Metrica utilizzata**: percentuale delle funzionalità comprese (MPDS6);
- **Soglia di accettabilità**: 85% - 100%;
- **Soglia di ottimalità**: 95% - 100%.

3.2.3.2.2 Consistenza operazionale in uso - OPDS7

Indica quante funzionalità rispettano le aspettative dell'utente.

- **Metrica utilizzata**: percentuale di funzionalità conformi alle aspettative (MPDS7);
- **Soglia di accettabilità**: 80% - 100%;
- **Soglia di ottimalità**: 90% - 100%.

3.2.4 Efficiency

È la capacità del prodotto software di fornire le proprie funzioni in modo appropriato, in relazione alla quantità di risorse utilizzate.

3.2.4.1 Strategie

Il software deve soddisfare le seguenti caratteristiche:

- **Time behaviour**: svolgere le proprie funzioni in tempi adeguati;
- **Resource utilisation**: eseguire le proprie funzioni utilizzando un'appropriata quantità di risorse.

3.2.4.2 Obiettivi di qualità

3.2.4.2.1 Tempo di risposta - OPDS8

Indica il periodo temporale medio trascorso tra la richiesta al software di una determinata funzionalità e la risposta all'utente.

- **Metrica utilizzata**: tempo medio di risposta (MPDS8);
- **Soglia di accettabilità**: 0 - 20 secondi;
- **Soglia di ottimalità**: 0 - 7 secondi;

3.2.5 Maintainability

È la capacità del prodotto software di essere modificato, ovvero corretto, migliorato o adattato in base a cambiamenti negli ambienti, nei requisiti o nelle specifiche funzionali.

3.2.5.1 Strategie

Il software deve soddisfare le seguenti caratteristiche:

- **Analysability**: poter essere analizzato per poter individuare cause di errori e/o parti da modificare;
- **Changeability**: permettere cambiamenti in alcune sue parti;
- **Stability**: evitare comportamenti indesiderati in seguito a modifiche;
- **Testability**: permettere l'esecuzione di test per validare le modifiche effettuate.

3.2.5.2 Obiettivi di qualità

3.2.5.2.1 Capacità analisi di failure - OPDS9

Indica il numero di failure di cui sono state individuate le cause. Maggiore è il valore più facile sarà individuare gli errori nel software e determinare delle soluzioni per correggerli.

- **Metrica utilizzata**: percentuale di failure con cause individuate (MPDS9)
- **Soglia di accettabilità**: 65% - 100%;
- **Soglia di ottimalità**: 85% - 100%;

3.2.5.2.2 Impatto delle modifiche - OPDS10

Indica il numero di modifiche effettuate in risposta alle failure che ne hanno introdotte di nuove. Valori alti indicano l'individuazione di soluzioni scandenti per la risoluzione delle failure.

- **Metrica utilizzata**: percentuale di failure introdotte con modifiche (MPDS10);
- **Soglia di accettabilità**: 0% - 20%;
- **Soglia di ottimalità**: 0% - 10%.

A Capability Maturity Model

Il Capability Maturity Model (CMM) è stato ideato e introdotto inizialmente dal Dipartimento della Difesa statunitense, per poi essere acquisito, sviluppato e sponsorizzato dalla SEI (Software Engineering Institute). Tale modello assume che la qualità del software dipende decisamente dal processo utilizzato per il suo sviluppo e per la successiva manutenzione, e consiste nell'applicare le migliori tecniche di gestione dei processi e del miglioramento della qualità. Si basa su:

- linee guida comuni per lo sviluppo e la manutenzione del software;
- struttura per la valutazione consistente dei livelli raggiunti.

A.1 Scopo

Lo scopo principale dell'adozione del modello in esame è quello di migliorare i processi di sviluppo del software in ottica di:

- miglioramento della qualità del software prodotto;
- aumento della produttività dell'organizzazione di sviluppo;
- riduzione dei tempi di sviluppo.

A.2 Struttura

Il CMM è costituito dalla seguente struttura:

- **Livelli di maturità**: Il modello definisce cinque livelli di maturità crescente del processo di sviluppo del software. Il più alto (il quinto) è uno stato ideale in cui i processi vengono sistematicamente gestiti da una combinazione di processi di ottimizzazione e di miglioramento continuo.
- **Aree chiave del processo**: identifica una serie di attività correlate che, se svolte collettivamente, realizzano un insieme di obiettivi considerati importanti;
- **Obiettivi**: indicano lo scopo, i confini e l'intento di ogni area chiave del processo;
- **Caratteristiche comuni**: includono le pratiche che implementano e regolamentano un'area chiave del processo. Ci sono cinque tipologie di caratteristiche comuni:
  - impegno nell'operare;
  - abilità nell'operare;
  - attività eseguite;
  - misurazioni ed analisi;
  - veriche dell'implementazione.
- **pratiche chiave**: descrivono gli elementi dell'infrastruttura e delle pratiche che contribuiscono maggiormente all'implementazione e la regolamentazione di un'area.

A.3 Livelli

I livelli di maturità che costituiscono il CMM sono:

- **Primo livello - iniziale (caotico)**: i processi che rientrano in questo livello sono disorganizzati. Il non essere sufficientemente definiti e documentati non permette loro di essere riutilizzati;

- **Secondo livello - ripetibile**: sono stabiliti processi base di gestione per tracciare i costi, la schedulazione delle attività e le funzionalità sviluppate. Il processo è stabilito per essere ripetibile.
- **Terzo livello - definito**: Il processo di sviluppo software, sia per la parte di gestione che per quella di sviluppo tecnico, è definito, documentato e standardizzato per il riutilizzo.
- **Quarto livello - gestito**: un organizzazione monitora e controlla i propri processi attraverso analisi e Data collection.
- **Quinto livello - ottimizzante**: i processi che rientrano in questo livello sono soggetti ad un continuo miglioramento delle proprie performance attraverso cambiamenti incrementali e miglioramenti tecnologici.

B PDCA

Il Plan-Do-Check-Act (PDCA), conosciuto anche come "Ciclo di Deming" o "Ciclo di miglioramento continuo", è un modello studiato per il miglioramento continuo della qualità in un'ottica a lungo raggio.
Questo modello permette di ricercare la qualità sui processi alla base del prodotto, e non sul prodotto stesso. Questo strumento permette di fissare degli obiettivi di miglioramento a partire dagli esiti delle misurazioni effettuate durante le varie attività di verifica. Una volta fissati gli obiettivi che si desiderano raggiungere, si iterano le quattro attività definite in seguito assicurando un incremento della qualità ad ogni ciclo.

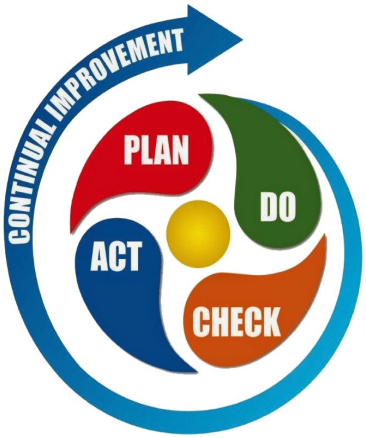

Figura 1: Continuous quality improvement with PDCA

- **Plan - Pianificare:** consiste nel definire gli obiettivi di miglioramento e le strategie da utilizzare per raggiungere la qualità attesa, in dettaglio:
  - identificare il problema o i processi da migliorare raccogliendo dati attraverso misurazioni;
  - analizzare il problema in modo tale da capire quali sono gli effetti negativi definendone l'importanza e la priorità di intervento;
  - definire gli obiettivi di massima in modo chiaro e quantitativo, indicando i benefici ottenibili con il suo raggiungimento. Devono essere definiti anche i tempi, gli indicatori e gli strumenti di controllo.
- **Do - Eseguire:** consiste nell'esecuzione di ciò che è stato pianificato nel punto precedente e nella raccolta dati necessaria all'analisi effettuata nei punti successivi;
- **Check - Verificare:** consiste nel verificare l'esito del processo (per efficienza ed efficacia) confrontandolo con i risultati attesi, così da poter definire se si va nella direzione giusta. Vanno considerate metriche come la Schedule Variance e la completezza dei risultati attesi soddisfatti, vanno elaborati grafici e tabelle per avere una visione chiara di quanto rilevato. Una volta raggiunto l'obiettivo definito nella attività di Plan si può passare a quella di Act, mentre se questo non è soddisfatto è necessario ripetere un nuovo ciclo PDCA sullo stesso problema analizzando i vari stadi del ciclo precedente individuandone le cause del non raggiungimento dell'obiettivo stabilito;
- **Act - Agire:** si standardizza la soluzione individuata ed ogni membro del gruppo di lavoro viene formato e informato. Una volta terminato questo stadio si proseguirà nuovamente dallo stadio 1, con un nuovo problema.

Bisogna tener presente che se l'obiettivo è il miglioramento continuo, le attività devono essere analizzabili, ripetibili e tracciabili. Unendo queste tre caratteristiche è possibile individuare eventuali errori e correggerli.

C Test

Al fine di produrre software di qualità, il gruppo ha strutturato dei test atti a verificare che le funzionalità del software prodotto corrispondano alle attese. Tali test sono ottenuti dall'applicazione delle tecniche di analisi dinamica descritte nel documento "*Norme di Progetto v2.0.0*". Inoltre, devono possedere le seguenti caratteristiche:

- devono essere ripetibili al fine di fornire informazioni utili per poter eseguire operazioni di correzione, ove sia necessario;
- devono essere tracciabili al fine di classificare le informazioni ottenute per garantire una più facile consultazione;

C.1 Test di validazione

I test di validazione hanno lo scopo di verificare che tutte le funzionalità richieste dal proponente siano soddisfatte. A questo scopo, attraverso una serie di azioni, si andrà a simulare il comportamento generale del software e dell'utente che interagisce con esso.
I test di validazione saranno identificati secondo quanto riportato nelle "*Norme di Progetto v2.0.0*".

C.2 Test di unità

I test di unità hanno lo scopo di verificare il corretto funzionamento delle unità. Le unità, individuate duarante la fase di progettazione, sono le più piccole parti del sistema dotate di funzionamento proprio. Questo si traduce nel verificare metodi e classi scritte dai *Programmatori*.
I test di unità saranno identificati secondo quanto riportato nelle "*Norme di Progetto v2.0.0*".

C.3 Test di integrazione

I test di integrazione hanno lo scopo di verificare il corretto funzionamento delle varie componenti. In particolare, l'obiettivo è quello di testare i vari package prodotti dall'unione delle unità. I test di integrazione saranno identificati secondo quanto riportato nelle "*Norme di Progetto v2.0.0*".

C.4 Test di sistema

I test di sistema hanno lo scopo di verificare il corretto funzionamento del prodotto software. Inoltre verranno verificate la sua robustezza in presenza di possibili malfunzionamenti e il suo comportamento di fronte a possibili violazioni (sicurezza).
I test di sistema saranno identificati secondo quanto riportato nelle "*Norme di Progetto v2.0.0*".

D Resoconto delle attività di verifica - fase AR

All'interno di questa sezione sono riportati gli esiti di tutte le attività di verifica effettuate nell'arco della fase AR, come previsto dal documento "*Piano di Progetto v1.0.0*". Ove necessario sono state tratte conclusioni sui risultati e su come essi possano essere migliorati.

D.1 Verifica sui processi

D.1.1 Processo di documentazione

D.1.1.1 Miglioramento costante

Per rendere le performance dei processi costantemente migliorabili e perseguire gli obiettivi quantitativi di miglioramento viene utilizzato il modello Capability Maturity Model (CMM_g).
All'inizio della fase i processi si trovavano al livello 1 della scala CMM_g. In seguito, grazie alla stesura del documento "*Norme di Progetto v1.0.0*" sono state definite regole per ogni tipo di documentazione, strumenti da utilizzare e procedure da seguire. Questo ha permesso un maggiore controllo del processo di documentazione, che ha ottenuto la ripetibilità, proprietà che caratterizza il livello 2 della scala CMM_g. Si può quindi affermare che il processo di documentazione ha raggiunto tale livello. Non si può ancora affermare di aver raggiunto il livello 3 del modello perchè al processo manca ancora la sua caratteristica principale, la proattività.
Secondo le metriche definite il valore raggiunto rappresenta la **soglia minima accettabile**, ma nelle prossime fasi il gruppo si impegnerà a raggiungere la soglia ottimale (sfruttando $PDCA_g$).

D.1.1.2 Rispetto della pianificazione

Per capire se l'attività di un processo rispetta i tempi stabiliti dalla pianificazione all'interno del "*Piano di Progetto v1.0.0*" viene utilizzata la metrica Schedule Variance. Si desidera, come soglia minima accettabile, che un processo sia in ritardo non più del 5% rispetto alla pianificazione. Sarebbe ottimale, invece, non avere ritardi rispetto alla pianificazione o, ancora meglio, essere in anticipo.
Di seguito sono riportati i valori ottenuti calcolando la Schedule Variance sui tempi di stesura di ogni documento nella fase AR:

Documento	**Schedule Variance**	**Esito**
"*Piano di Progetto v1.0.0*"	0%	ottimale
"*Norme di Progetto v1.0.0*"	0%	ottimale
"*Analisi dei Requisiti v1.0.0*"	-10%	ottimale
"*Piano di Qualifica v1.0.0*"	0%	ottimale
"*Glossario v1.0.0*"	0%	ottimale
"*Analisi SDK dei principali Assistenti Virtuali v1.0.0*"	0%	ottimale
"*Studio di Fattibilità v1.0.0*"	0%	ottimale

Tabella 1: Esiti del calcolo della Schedule Variance sul processo di documentazione durante la fase AR

D.1.1.3 Rispetto del budget

Per capire se i costi di un processo rientrano nel budget stabilito dalla pianificazione all'interno del "*Piano di Progetto v1.0.0*" viene utilizzata la metrica Cost Variance. Si desidera, come soglia minima accettabile, che un processo non superi il 10% del budget pianificato. Sarebbe ottimale, invece, non superare i costi pianificati o, ancora meglio, spendere meno.
Di seguito sono riportati i valori ottenuti calcolando la Cost Variance sui tempi di stesura di ogni documento nella fase AR:

Documento	Cost Variance	Esito
"*Piano di Progetto v1.0.0*"	0%	ottimale
"*Norme di Progetto v1.0.0*"	+10%	accettabile
"*Analisi dei Requisiti v1.0.0*"	+11%	non accettabile
"*Piano di Qualifica v1.0.0*"	+16%	non accettabile
"*Glossario v1.0.0*"	+6%	accettabile
"*Analisi SDK dei principali Assistenti Virtuali v1.0.0*"	+14%	non accettabile
"*Studio di Fattibilità v1.0.0*"	+10%	accettabile

Tabella 2: Esiti del calcolo della Cost Variance sul processo di documentazione durante la fase AR

Tali valori sono dovuti al fatto che l'attività degli *Amministratori* ha richiesto più tempo del previsto in quanto è stato necessario modificare alcune funzioni del software utilizzato per il tracciamento dei requisiti e dei casi d'uso, inoltre l'attività degli *Analisti* ha richiesto più tempo del previsto, in quanto si è dovuta fare un'analisi più approfondita rispetto a quella prefissata per una corretta stesura dei requisiti e dei casi d'uso. Questo è dovuto, in parte, all'interfaccia vocale da progettare, non convenzionale.

D.1.2 Processo di verifica

D.1.2.1 Miglioramento costante

Per rendere le performance dei processi costantemente migliorabili e perseguire gli obiettivi quantitativi di miglioramento viene utilizzato il modello Capability Maturity Model (CMM_g).
All'inizio della fase i processi si trovavano al livello 1 della scala CMM_g. In seguito, grazie alla stesura del documento "*Norme di Progetto v1.0.0*" sono state definite regole per ogni tipo di documentazione, strumenti da utilizzare e procedure da seguire, oltre che alla definizione di metriche in questo documento. Questo ha permesso un maggiore controllo del processo di verifica, che ha ottenuto la ripetibilità, proprietà che caratterizza il livello 2 della scala CMM_g. Si può quindi affermare che il processo di documentazione ha raggiunto tale livello. Non si può ancora affermare di aver raggiunto il livello 3 del modello perchè al processo manca ancora la sua caratteristica principale, la proattività.
Secondo le metriche definite il valore raggiunto rappresenta la **soglia minima accettabile**, ma nelle prossime fasi il gruppo si impegnerà a raggiungere la soglia ottimale (sfruttando $PDCA_g$).

D.1.2.2 Rispetto della pianificazione

Per capire se l'attività di un processo rispetta i tempi stabiliti dalla pianificazione all'interno del "*Piano di Progetto v1.0.0*" viene utilizzata la metrica Schedule Variance. Si desidera, come soglia minima accettabile, che un processo sia in ritardo non più del 5% rispetto alla pianificazione.

Sarebbe ottimale, invece, non avere ritardi rispetto alla pianificazione o, ancora meglio, essere in anticipo.
Di seguito è riportato il valore ottenuto calcolando la Schedule Variance sul processo di verifica nella fase AR:

Processo	**Schedule Variance**	**Esito**
Processo di verifica	-6%	ottimale

Tabella 3: Esiti del calcolo della Schedule Variance sul processo di verifica durante la fase AR

D.1.2.3 Rispetto del budget

Per capire se i costi di un processo rientrano nel budget stabilito dalla pianificazione all'interno del "*Piano di Progetto v1.0.0*" viene utilizzata la metrica Cost Variance. Si desidera, come soglia minima accettabile, che un processo non superi il 10% del budget pianificato. Sarebbe ottimale, invece, non superare i costi pianificati o, ancora meglio, spendere meno.
Di seguito è riportato il valore ottenuto calcolando la Cost Variance sul processo di verifica nella fase AR:

Processo	**Cost Variance**	**Esito**
Processo di verifica	-25%	ottimale

Tabella 4: Esiti del calcolo della Cost Variance sul processo di verifica durante la fase AR

D.2 Verifica sui prodotti

D.2.1 Documenti

D.2.1.1 Leggibilità e comprensibilità

Per determinare il grado di leggibilità e comprensibilità del documento, il gruppo ha deciso di utilizzare l'indice Gulpease. Si desidera come soglia minima accettabile un indice maggiore o uguale a 40 e, come soglia ottimale, un indice maggiore di 60.
Di seguito sono riportati i valori ottenuti calcolando l'indice Gulpease sui documenti della fase AR:

Documento	**Gulpease**	**Esito**
"*Piano di Progetto v1.0.0*"	49	accettabile
"*Norme di Progetto v1.0.0*"	58	accettabile
"*Analisi dei Requisiti v1.0.0*"	66	ottimale
"*Piano di Qualifica v1.0.0*"	54	accettabile
"*Glossario v1.0.0*"	50	accettabile
"*Analisi SDK dei principali Assistenti Virtuali v1.0.0*"	67	ottimale
Verbale esterno 2016-12-17	66	ottimale
Verbale interno 2016-12-10	61	ottimale
Verbale interno 2016-12-19	62	ottimale

Tabella 5: Esiti del calcolo dell'indice Gulpease sui documenti della fase AR

D.2.1.2 Correttezza ortografica

Per determinare il grado di correttezza ortografica del documento, il gruppo ha deciso di utilizzare la seguente metrica: percentuale di errori ortografici rinvenuti e non corretti. Pertanto, la soglia minima accettabile e la soglia ottimale coincidono e corrispondono a una correzione totale degli errori rinvenuti.
Di seguito è riportato il numero di errori ortografici trovati:

Errori ortografici	43

Tabella 6: Errori ortografici rinvenuti durante la fase AR

Tutti gli errori ortografici rinvenuti sono stati corretti, quindi è stato raggiunto l'obiettivo **ottimale**.

D.2.1.3 Correttezza concettuale

Per determinare il grado di correttezza concettuale del documento, il gruppo ha deciso di utilizzare la seguente metrica: percentuale di errori concettuali rinvenuti e non corretti. Si desidera come soglia minima accettabile che non più del 5% degli errori concettuali rinvenuti non siano stati corretti e, come soglia ottimale, che tutti gli errori concettuali rinvenuti siano stati corretti.
Di seguito è riportato il numero di errori concettuali trovati:

Errori concettuali	24

Tabella 7: Errori concettuali rinvenuti durante la fase AR

Tutti gli errori concettuali rinvenuti sono stati corretti, quindi è stato raggiunto l'obiettivo **ottimale**.