Don't Panic

3DMob: Grafica 3D su device mobili

Piano di Qualifica

Informazioni sul documento

Versione	1.2.0
Redazione	Pezzutti Marco Cesarato Fabio
Verifica	Busato Luca Lain Daniele
Responsabile	Cesarato Fabio
Uso	Esterno
Lista di distribuzione	Prof. Vardanega Tullio Prof. Cardin Riccardo Mentis srl

Descrizione

Documento riguardante le strategie di verifica adottate da Don't Panic volte al perseguimento costante di requisiti qualitativi per il progetto 3DMob

Diario delle modifiche

Descrizione modifica	Autore	Data	Versione
Approvazione documento	Cesarato Fabio	2012-12-18	1.2.0
Verifica documento	Lain Daniele	2012-12-16	1.1.1
Verifica documento	Busato Luca	2012-12-15	1.1.0
Strutturato resoconto delle attività di verifica	Cesarato Fabio	2012-12-12	1.0.7
Stesura sezione analisi statica e dinamica	Pezzutti Marco	2012-12-01	1.0.6
Stesura gestione amministrativa della revisione	Cesarato Fabio	2012-11-30	1.0.5
Stesura responsabilità e risorse	Pezzutti Marco	2012-11-30	1.0.4
Stesura organizzazione e pianificazione strategica	Pezzutti Marco	2012-11-29	1.0.3
Descritti strumenti, metriche e metodi	Cesarato Fabio	2012-11-28	1.0.2
Stesura sezioni introduzione e obiettivi di qualità	Pezzutti Marco	2012-11-27	1.0.1
Creazione struttura principale	Pezzutti Marco	2012-11-26	1.0.0

Indice

Elenco delle tabelle

Elenco delle figure

1 Introduzione

1.1 Scopo del documento

Il *Piano di Qualifica* ha lo scopo di descrivere le strategie che il gruppo di lavoro ha deciso di adottare per perseguire obiettivi qualitativi da applicare al proprio prodotto. Per ottenere tali obiettivi è necessario un processo di verifica continua sulle attività svolte; questo consentirà di rilevare e correggere anomalie e incongruenze in modo tempestivo e senza spreco di risorse.

1.2 Scopo del Prodotto

Lo scopo del progetto è la realizzazione di un applicazione in grado di convertire file prodotti da programmi di grafica 3D in file in formato JSON$_G$ in grado di essere visualizzati su dispositivi mobile senza perdita di informazione. L'obiettivo è quello di semplificare il workflow attuale necessario a rendere compatibili i file.

1.3 Glossario

Al fine di evitare ogni ambiguità di linguaggio e massimizzare la comprensione dei documenti, i termini tecnici, di dominio, gli acronimi e le parole che necessitano di essere chiarite, sono riportate nel documento *Glossario v1.2.0*.
Ogni occorrenza di vocaboli presenti nel *Glossario* è marcata da una "G" maiuscola in pedice.

1.4 Riferimenti

1.4.1 Normativi

- **Norme di Progetto:** *Norme di Progetto v1.2.0*;
- **Capitolato d'appalto C2:** 3DMob: Grafica 3D su device mobili http://www.math.unipd.it/~tullio/IS-1/2012/Progetto/C2.pdf.

1.4.2 Informativi

- **Piano di Progetto:** *Piano di Progetto v1.2.0*;
- **Slide dell'insegnamento Ingegneria del Software modulo A:** http://www.math.unipd.it/~tullio/IS-1/2012/;
- **SWEBOK - Version 3 (2004):** capitolo 11 - Software Quality http://www.computer.org/portal/web/swebok/html/ch11;
- **Ingegneria del software - Ian Sommerville - 8ª Edizione (2007)**:
  - Capitolo 27 - Gestione della qualità;
  - Capitolo 28 - Miglioramento dei processi.
- **Standard ISO$_G$/IEC$_G$ TR 15504:** Software process assessment http://en.wikipedia.org/wiki/ISO/IEC_15504;
- **Standard ISO$_G$/IEC$_G$ 9126:** Product quality http://en.wikipedia.org/wiki/ISO/IEC_9126;

- **Indice Gulpease:**
  - http://www.eulogos.net/ActionPagina_1021.do#IndiceGULPEASE;
  - http://it.wikipedia.org/wiki/Indice_Gulpease.
- **Complessità ciclomatica:**
  http://it.wikipedia.org/wiki/Complessit%C3%A0_ciclomatica.

2 Visione generale della strategia di verifica

2.1 Definizione obiettivi

La seguente sezione intende descrivere sia gli obiettivi di qualità relativi al prodotto richiesto dal committente, che quelli relativi ai processi necessari al suo completamento.

2.1.1 Qualità di processo

Affinché la qualità del prodotto sia garantita è necessario perseguire la qualità dei processi che lo definiscono. Per fare questo si è quindi deciso di adottare lo standard ISO_G/IEC_G 15504 denominato SPICE[1] il quale fornisce gli strumenti necessari a valutare l'idoneità di quest'ultimi. Questo modello descrive come ogni processo debba essere

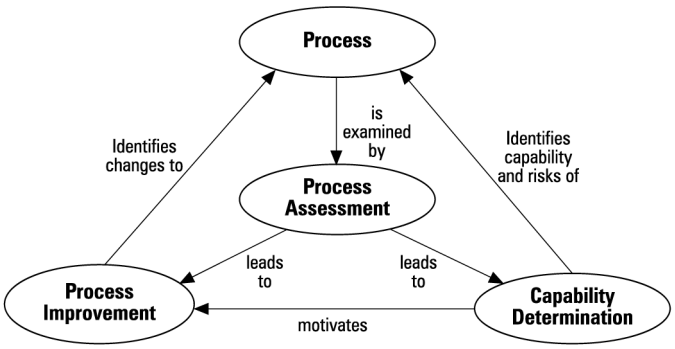

Figura 1: Modello SPY - Software Process Assessment and Improvement

controllato continuamente con lo scopo di rilevare possibili rischi e debolezze intrinseci che impediscono di raggiungere gli obiettivi prefissati, e di individuare le possibili cause in modo da migliorare l'efficienza di tali processi. I risultati delle singole valutazioni devono essere ripetibili, oggettivi e comparabili in contesti simili affinché contribuiscano al miglioramento effettivo dei processi in esame.
Il modello SPICE definisce 6 possibili livelli di maturità del processo i quali possiedono degli attributi utili per misurarla:

0. **Incomplete process:** il processo non viene attuato o non riesce a raggiungere i suoi risultati;
1. **Performed:** il processo attuato raggiunge i suoi risultati
   (a) **Process performance attribute:** capacità di un processo di raggiungere gli obiettivi trasformando input identificabili in output identificabili.
2. **Managed process:** il processo viene eseguito in modo controllato in base a obiettivi definiti
   (a) **Performance management attribute:** capacità del processo di elaborare un prodotto coerente con gli obiettivi fissati;
   (b) **Work product management attribute:** capacità del processo di elaborare un prodotto documentato, controllato e verificato.

[1] Software Process Improvement Capability dEtermination

3. **Established process:** il processo viene eseguito basandosi su principi dell'ingegneria del software ed è in grado di raggiungere i risultati fissati
   (a) **Process definition attribute:** l'esecuzione del processo si basa su standard di processo per raggiungere i propri obiettivi;
   (b) **Process resource attribute:** capacità del processo di attingere a risorse tecniche e umane appropriate per essere attuato efficacemente.
4. **Predictable process:** il processo viene eseguito costantemente entro limiti definiti per raggiungere i risultati attesi
   (a) **Measurement attribute:** gli obiettivi e le misure di prodotto e di processo vengono usati per garantire il raggiungimento dei traguardi definiti in supporto ai target aziendali;
   (b) **Process control attribute:** il processo viene controllato tramite misure di prodotto e processo per effettuare correzioni migliorative al processo stesso.
5. **Optimizing process:** il processo cambia e si adatta dinamicamente per raggiungere gli obiettivi aziendali
   (a) **Process change attribute:** i cambiamenti strutturali, di gestione e di esecuzione vengono gestiti in modo controllato per raggiungere i risultati fissati;
   (b) **Continuous improvement attribute:** le modifiche al processo sono identificate e implementate per garantire il miglioramento continuo nella realizzazione degli obiettivi di business dell'organizzazione.

Ogni attributo di processo precedentemente descritto è misurabile e lo standard predispone 4 differenti livelli:

N non posseduto (0% - 15%);

P parzialmente posseduto (16% - 50%);

L largamente posseduto (51% - 85%);

F completamente posseduto (86% - 100%).

Appare evidente che per applicare correttamente questo modello si deve utilizzare il ciclo di Deming$_G$ (ciclo PDCA$_G$) il quale definisce una metodologia di controllo dei processi durante il loro ciclo di vita che consente di migliorarne in modo continuativo la qualità. Esso si suddivide nelle seguenti 4 fasi:

- **Plan:** fase di pianificazione in cui vengono definite attività, risorse, scadenze e responsabilità;
- **Do:** fase di esecuzione delle attività pianificate;
- **Check:** fase di verifica in cui vengono controllati i risultati ottenuti nella fase *Do* e confrontati con quelli pianificati nella fase *Plan*;
- **Act:** fase in cui si mette in pratica il miglioramento continuo dei processi utilizzando i risultati della verifica per modificare gli aspetti critici dei processi in esame.

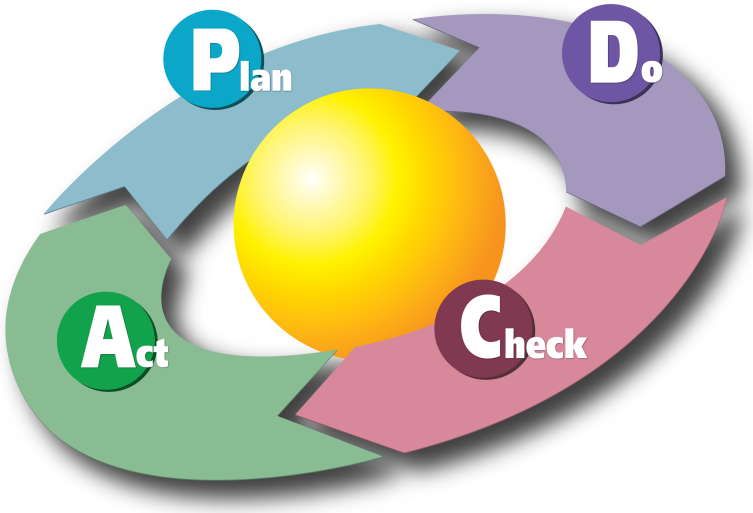

Figura 2: Ciclo PDCA

2.1.2 Qualità di prodotto

Al fine di aumentare il valore commerciale di un prodotto software e di garantirne il corretto funzionamento è necessario fissare degli obiettivi qualitativi e di garantire che questi vengano effettivamente rispettati.
Lo standard ISO_G/IEC_G 9126 è stato redatto con lo scopo di descrivere questi obiettivi e delineare delle metriche capaci di misurare il raggiungimento di tali obiettivi. Esso divide i criteri qualitativi in 3 aree diverse:

- **Qualità in uso:** è la qualità del prodotto software dal punto di vista dell'utilizzatore che ne fa uso all'interno di uno specifico sistema e contesto;
- **Qualità esterna:** è la qualità del prodotto software vista dall'esterno nel momento in cui esso viene eseguito e testato in un ambiente di prova;
- **Qualità interna:** è la qualità del prodotto software vista dall'interno, fa quindi riferimento alle caratteristiche implementative del software quali l'architettura e il codice che ne deriva.

Non avendo la possibilità di testare la qualità in uso del prodotto software da sviluppare, si è deciso di concentrarsi sulla qualità interna ed esterna. Lo standard ISO_G/IEC_G 9126 prevede 6 caratteristiche qualitative principali, suddivise in ulteriori sotto caratteristiche che possono essere misurate quantitativamente tramite metriche specifiche:

- **Funzionalità:** capacità del prodotto software di fornire funzioni che rispondano a determinate esigenze
  - **Idoneità:** capacità del prodotto software di fornire un insieme appropriato di funzioni per attività specifiche;
  - **Accuratezza:** capacità del prodotto software di fornire risultati corretti o concordati con il grado di precisione necessario;

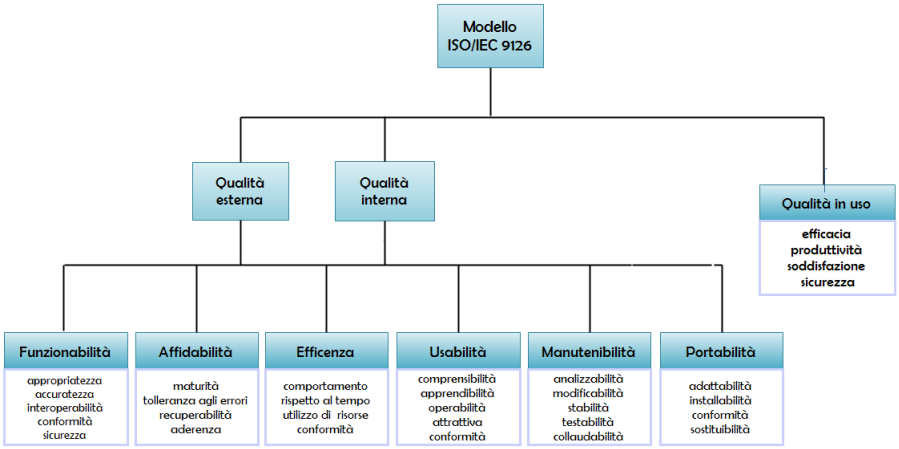

Figura 3: Caratteristiche qualitative modello ISO/IEC 9126

  - **Interoperabilità:** capacità del prodotto software di interagire con uno o più sistemi specifici;
  - **Sicurezza:** capacità del prodotto software di proteggere dati e informazioni in modo da impedire l'accesso e la modifica a persone o sistemi non autorizzati e a consentire l'uso di tali dati ai soli autorizzati;
  - **Conformità funzionale:** capacità del prodotto software di aderire a standard, convenzioni o regolamentazioni e prescrizioni in materia di funzionalità.

- **Affidabilità:** capacità del prodotto software di mantenere un adeguato livello di prestazioni
  - **Maturità:** capacità del prodotto software di evitare fallimenti a causa di errori nel software;
  - **Tolleranza agli errori:** capacità del prodotto software di mantenere un adeguato livello di prestazioni in caso di errori software o di violazioni alla propria interfaccia;
  - **Capacità di recupero:** capacità del prodotto software di ristabilire un adeguato livello di performance e di recuperare i dati interessati in caso di errori;
  - **Conformità di affidabilità:** capacità del prodotto software di aderire a standard, convenzioni o regolamentazioni in materia di affidabilità.

- **Usabilità:** capacità del software di essere capito, imparato, usato e di essere allettante per l'utente
  - **Intelligibilità:** capacità del prodotto software di consentire all'utente di capire se il software è adeguato e come può essere utilizzato per compiti particolari;
  - **Apprendibilità:** capacità del prodotto software di consentire all'utente di imparare le sue applicazioni;

  - **Operabilità:** capacità del prodotto software di consentire all'utente di usarlo e controllarlo;
  - **Attrattività:** capacità del prodotto software di creare interesse nell'utente;
  - **Conformità di usabilità:** capacità del prodotto software di aderire a standard, convenzioni o regolamentazioni in materia di usabilità.

- **Efficienza:** capacità del software di fornire prestazioni appropriate in relazione alla quantità di risorse usate
  - **Comportamento temporale:** capacità del software di fornire tempi di risposta e di elaborazione adeguati quando svolge le sue funzioni;
  - **Utilizzo di risorse:** capacità del prodotto software di utilizzare tipologia e quantità di risorse adeguate durante la sua esecuzione;
  - **Conformità di efficienza:** capacità del prodotto software di aderire a standard, convenzioni o regolamentazioni in materia di efficienza.

- **Manutenibilità:** capacità del prodotto software di essere modificato. Le modifiche comprendono correzioni, miglioramenti o adattamenti del software a cambiamenti ambientali, nelle specifiche o nelle funzionalità
  - **Analizzabilità:** capacità del prodotto software di poter essere studiato alla ricerca di carenze e difetti;
  - **Modificabilità:** capacità del prodotto software di consentire l'implementazione di una specifica modifica;
  - **Stabilità:** capacità del prodotto software di evitare effetti indesiderati causati da una o più modifiche;
  - **Testabilità:** capacità del prodotto software di consentire la validazione di una versione modificata del software;
  - **Conformità di manutenibilità:** capacità del prodotto software di aderire a standard, convenzioni o regolamentazioni in materia di manutenibilità.

- **Portabilità:** capacità del prodotto software di poter essere trasferito da un ambiente di lavoro ad un altro
  - **Adattabilità:** capacità del prodotto software di essere adattato a diversi ambienti senza la necessità di effettuare modifiche diverse da quelle fornite;
  - **Installabilità:** capacità del prodotto software di poter essere installato in specifici ambienti;
  - **Coesistenza:** capacità del prodotto software di coesistere in ambienti comuni con altri software indipendenti condividendo risorse comuni;
  - **Sostituibilità:** capacità del prodotto software di poter sostituire un software analogo con la stessa finalità e nello stesso ambiente;
  - **Conformità di portabilità:** capacità del prodotto software di aderire a standard, convenzioni o regolamentazioni in materia di portabilità.

2.2 Organizzazione

Il processo di **verifica** viene istanziato nel momento in cui l'output di un processo passa dalla versione $X.0.Z$ alla versione $X.1.0$.
Attraverso il diario delle modifiche è possibile focalizzare l'attenzione maggiormente sulle sezioni che hanno subito dei cambiamenti dall'ultima verifica, riducendo il tempo necessario al controllo.
Ognuna delle 5 fasi del progetto descritte nel *Piano di Progetto v1.2.0* necessita di diverse attività di verifica a causa della differente natura degli output ottenuti:

- **Analisi:** in tale fase si devono seguire i metodi di verifica descritti nelle sezioni 2.9.2 e 2.9.1 sui documenti prodotti e sui processi attuati. L'attuazione di tali attività di verifica, essendo antecedenti alla Revisione dei Requisiti, sono descritte nella sezione 4.1.1

Concluso il processo di verifica ha inizio il processo di **approvazione**.Tale processo viene istanziato nel momento in cui l'output di un processo passa dalla versione $X.1.Z$ alla versione $X.2.0$.
Durante tale processo sarà compito del *Responsabile di Progetto* accertarsi che a livello macroscopico i prodotti risultino conformi a quanto pianificato e progettato.

2.3 Pianificazione strategica e temporale

Avendo l'obiettivo di rispettare le scadenze fissate nel *Piano di Progetto v1.2.0*, è necessario che l'attività di verifica della documentazione e del codice sia sistematica e ben organizzata. Applicando tali principi l'individuazione e la correzione degli eventuali errori avverrà il prima possibile, impedendo una rapida diffusione degli stessi.
Ogni fase di redazione dei documenti e di codifica deve essere preceduta da una fase di studio preliminare. Tale fase ha lo scopo di ridurre la possibilità di commettere imprecisioni di natura concettuale e tecnica favorendo l'attività di verifica, dove saranno necessari minori interventi di correzione.
Di seguito vengono riportate le scadenze fissate:

- **Revisioni formali:**
  - **Revisione dei Requisiti:** 2012-12-21;
  - **Revisione di Accettazione:** 2013-03-16.
- **Revisioni di progresso:**
  - **Revisione di Progettazione:** 2013-02-03;
  - **Revisione di Qualifica:** 2013-03-06.

2.4 Responsabilità

- ***Responsabile di Progetto*:** ha il compito di assicurarsi che le attività di verifica vengano svolte sistematicamente seguendo le *Norme di Progetto*, vengano rispettati i ruoli e le competenze assegnate nel *Piano di Progetto*, non vi siano conflitti di interesse tra redattori e verificatori. Egli è l'unico a poter decidere l'approvazione di un documento e a sancirne la distribuzione.

  Ha inoltre l'incarico di gestire la creazione e l'assegnazione dei ticket delle macrofasi e di assegnare ad un membro del gruppo il ruolo di responsabile di quest'ultima;

- ***Verificatore*:** ha il compito di effettuare la verifica dei documenti utilizzando gli strumenti e i metodi proposti dal *Piano di Qualifica* e attenendosi a quanto descritto nelle *Norme di Progetto*.

  Nel momento in cui viene riscontrato un errore, esso deve essere segnalato tramite ticket come descritto nelle *Norme di Progetto*, e qualora il documento risulti essere giunto ad una versione finale, dovrà essere sottoposto all'approvazione del *Responsabile di Progetto*.

2.5 Risorse

Per la realizzazione del prodotto software sono necessarie sia risorse tecnologiche che umane:

- **Risorse umane:** vengono descritte dettagliatamente nel *Piano di Progetto*
  - *Responsabile di Progetto*;
  - *Amministratore di Progetto*;
  - *Analista*;
  - *Progettista*;
  - *Programmatore*;
  - *Verificatore*.
- **Risorse software:** sono necessari tutti gli strumenti software utili:
  - alla gestione di documentazione in LaTeX;
  - alla creazione dei diagrammi in UML$_G$;
  - allo sviluppo del codice nel linguaggio di programmazione scelto;
  - a semplificare ed automatizzare la verifica;
  - a gestire i test e le analisi sul codice.

  Tali risorse sono descritte nelle *Norme di Progetto* e nella sezione 2.6 del presente documento;
- **Risorse hardware:** sono necessari dei computer dotati di tutti gli strumenti software descritti nel *Piano di Qualifica* e nelle *Norme di Progetto*. È inoltre necessario avere a disposizione uno o più luoghi dove effettuare le riunioni interne del gruppo di lavoro.

I membri del gruppo Don't Panic hanno a disposizione almeno un computer personale, dotato degli strumenti necessari, su cui poter svolgere i propri compiti.
Viene messo a disposizione l'appartamento di un membro del gruppo come sede operativa dotata di connessione a banda larga e ospitante il server che il gruppo utilizzerà per la gestione del repository$_G$.

2.6 Strumenti

Le risorse software che si utilizzeranno durante il processo di verifica sono:

- **Correttore automatico TeXstudio**: come segnalato nelle *Norme di Progetto v1.2.0* per la scrittura di documenti è consigliato utilizzare l'ambiente grafico TeXstudio. Tale strumento integra i dizionari di OpenOffice.org e segnala i potenziali errori ortografici presenti nel testo;

- **Aspell**: strumento per la correzione tipografica dei documenti redatti in LaTeX;
- **Tracy**: strumento realizzato dal gruppo Don't Panic.
  Tale software contiene ed associa:
  - *Fonti* di requisiti individuate, inclusi anche i casi d'uso;
  - *Requisiti* individuati durante l'analisi.

  Permette inoltre di esportare automaticamente:
  - Codice LaTeX per la descrizione dei casi d'uso;
  - Anteprima del diagramma dei casi d'uso (indicativo, non UML$_G$);
  - Tabella in LaTeX per il tracciamento fonti-requisiti.

  Permette quindi un facile ed automatizzato **tracciamento dei requisiti**.
  Raggiungibile al sito `http://dtp.myopenproject.it/requisiti/`;
- **Gulpease-script**: script, scritto dai componenti del gruppo, che calcola automaticamente l'indice Gulpease del documento scritto;
- **Glossario-script**: script, scritto dai componenti del gruppo, che marca i termini presenti nel glossario con la simbologia descritta nelle *Norme di Progetto v1.2.0*;
- **CppCheck**: analizzatore statico per individuare errori sintattici comuni del codice C++;
- **CCCC** (C and C++ Code Counter): Misura metriche riguardanti codice sorgente in C++. Le metriche generate da questo programma, utilizzate per la qualità, sono: Complessità ciclomatica, Numero di livelli di annidamento, Linee di codice per linee di commento, Flusso di informazioni, Accoppiamento;
- **Clang**: compilatore che segnala errori e warnings in modo molto espressivo;
- **GCC**: compilatore che segnala errori e warnings in modo dettagliato e chiaro;
- **Valgrind**: strumento per l'analisi dinamica. Rileva problemi di memoria, ricerca i memory leak$_G$ e restituisce il tempo richiesto per l'esecuzione di ogni funzione;
- **QtTest**: framework$_G$ per test di unità su codice C++.

Ulteriori ed approfondite specifiche dei programmi sono disponibili nelle *Norme di Progetto v1.2.0*.

2.7 Tecniche di analisi

2.7.1 Analisi statica

L'analisi statica è una tecnica di analisi, applicabile sia alla documentazione che al codice, che permette di effettuare la verifica di quanto prodotto individuando errori ed anomalie; essa può essere svolta in due modi distinti ma complementari.

2.7.1.1 Walkthrough

Si svolge effettuando una lettura critica a largo spettro. È una tecnica che viene utilizzata soprattutto nelle prime fasi del progetto, quando ancora non è presente una adeguata esperienza da parte dei membri del gruppo, che permetta di attuare una verifica più mirata e precisa.
Con l'utilizzo di questa tecnica, il *Verificatore* sarà in grado di stilare una lista di controllo con gli errori più frequenti in modo da favorire il miglioramento di tale attività nelle fasi future.
Questa è un'attività onerosa e collaborativa che richiede l'intervento di più persone per essere efficace ed efficiente. Dopo una prima fase di lettura e individuazione degli errori, segue una fase di discussione con la finalità di esaminare i difetti riscontrati e di proporre le dovute correzioni. L'ultima fase consiste nel correggere gli errori rilevati e nello scrivere un rapporto che elenchi le modifiche effettuate.

2.7.1.2 Inspection

Questa tecnica consiste nell'analisi mirata di alcune parti del documento o del codice che sono ritenute fonti maggiori di errore. La *lista di controllo*, che deve essere seguita per svolgere efficacemente questo processo, deve essere redatta anticipatamente ed è frutto dell'esperienza maturata dai verificatori attraverso la tecnica di walkthrough.
L'inspection è una strategia più rapida del walkthrough in quanto consente l'analisi di alcune parti dei prodotti ritenute critiche dalla checklist e non necessità della lettura integrale dei documenti in oggetto.
Diversamente dal walkthrough, tale tecnica viene svolta esclusivamente dai verificatori che dopo aver individuato gli errori procedono alla loro correzione e alla redazione di un rapporto di verifica che tenga traccia del lavoro svolto.

Durante l'applicazione del walkthrough ai documenti, sono state riportate le tipologie di errori più frequenti. La lista di controllo risultante è la seguente:

- **Norme stilistiche:**
  - Elenco puntato: non inizia con la lettera maiuscola;
  - Elenco puntato: non termina con il punto e virgola oppure con il punto se è l'ultimo elemento;
  - Elenco numerato: non termina con il punto e virgola oppure con il punto se è l'ultimo elemento;
  - Nome proprio di persona: non rispetta la norma Cognome Nome;
  - Nome ruolo di progetto: non viene utilizzata la macro predisposta;
  - Nome documento: non viene utilizzata la macro predisposta;
  - Parole Proponente e Committente: non vengono scritte con la maiuscola iniziale.
- **Italiano:**
  - Parola destinatario: viene erroneamente scritta come destinario;
  - Virgola tra soggetto e verbo: rende di difficile comprensione la frase;
  - Carattere È: non viene scritto correttamente utilizzando il comando apposito;

  - Periodi: frasi troppo lunghe rendono i concetti di difficile comprensione;
  - Doppie negazioni: evitare l'utilizzo di doppie negazioni perché complicano la comprensione della frase;
  - Punto e virgola: evitare l'uso del punto e virgola quando è necessario usare il punto;
  - Proponente e Committente: non si deve confondere il loro significato.

- **LaTeX:**
  - Lettere accentate nelle variabili: non viene utilizzato il comando apposito;
  - Carattere di spaziatura: non deve essere utilizzato all'interno dei tag;
  - Macro LaTeX: non viene scritta usando l'apposito comando \LaTeX{}.

- **UML$_G$:**
  - Il sistema non deve mai essere un attore;
  - Controllo ortografico: deve essere effettuato in modo dettagliato a causa dell'impossibilità di automatizzare i controlli sui diagrammi;
  - Direzione delle frecce non corrette;
  - Consistenza della nomenclatura tra i diagrammi e le descrizioni testuali nei documenti.

La seguente lista di controllo vuole riassumere invece gli errori più frequenti rilevati durante il walkthrough del tracciamento requisiti effettuato mediante il software Tracy:

- **Tracciamento requisiti:**
  - Ad ogni caso d'uso deve corrispondere almeno un requisito;
  - Ad ogni requisito deve corrispondere almeno una fonte;
  - La fonte "Capitolato" non deve comparire nei requisiti interni;
  - Deve esserci copertura totale del capitolato nei requisiti;
  - La checkbox appointed deve essere spuntata solo nei requisiti opzionali e desiderabili presi in carico;
  - Devono essere impostate le corrette relazioni di parentela tra requisiti in Tracy;
  - Devono essere impostate le corrette relazioni di parentela tra casi d'uso in Tracy;
  - In Tracy deve essere indicato almeno un attore in ogni caso d'uso;
  - Controllare in Tracy che le fonti dei requisiti siano le fonti corrette;
  - I codici dei casi d'uso nei diagrammi e in Tracy devono corrispondere.

2.7.2 Analisi dinamica

L'analisi dinamica si applica solamente al prodotto software e viene svolta durante l'esecuzione del codice mediante l'uso di test predisposti per verificarne il funzionamento e rilevare possibili difetti di implementazione.
Affinché tale attività sia utile e generi risultati attendibili è necessario che i test effettuati siano *ripetibili*: questa caratteristica è fondamentale in quanto solo un test che, dato

un certo input, produce sempre lo stesso output su uno specifico ambiente, è capace di riscontrare problemi e verificare la correttezza del prodotto software.
Di conseguenza devono essere definiti a priori:

- **Ambiente:** consiste sia del sistema hardware che di quello software sui quali è stato pianificato l'utilizzo del prodotto software sviluppato; di essi deve essere specificato uno stato iniziale dal quale poter iniziare ad eseguire i test;
- **Specifica:** consiste nel definire quali input sono necessari per l'esecuzione del test e quali devono essere gli output attesi;
- **Procedure:** consiste nel definire come devono essere svolti i test, in quale ordine e come devono essere analizzati i risultati ottenuti.

Sono definibili 5 tipi diversi di test: *test di unità*, *test di integrazione*, *test di sistema*, *test di regressione* e *test di accettazione*.

2.7.2.1 Test di unità

Consiste nella verifica di ogni singola unità del prodotto software tramite l'utilizzo di stub_G, driver_G e logger_G.
Per unità si intende la più piccola quantità di software che è utile verificare singolarmente e che viene prodotta da un singolo programmatore.
Attraverso tali test si vuole verificare il corretto funzionamento dei moduli che compongono l'intero sistema in modo da eliminare possibili errori di implementazione da parte dei programmatori.

2.7.2.2 Test di integrazione

Consiste nella verifica dei componenti del sistema che vengono aggiunti incrementalmente al prodotto e si prefigge quindi di analizzare che la combinazione di 2 o più unità software funzioni come previsto.
Questo tipo di test serve ad individuare errori residui nella realizzazione dei singoli moduli, modifiche delle interfacce e comportamenti inaspettati di componenti software preesistenti forniti da terze parti che non si conoscono a fondo.
Per effettuare tali test devono essere aggiunte delle componenti fittizie che non sono ancora state sviluppate in modo da non influenzare negativamente l'esito dell'analisi.

2.7.2.3 Test di sistema

Consiste nella validazione del prodotto software nel momento in cui vengono aggiunti tutti i componenti e lo si ritiene giunto ad una versione definitiva.
Tale test deve verificare che la copertura dei requisiti software stabiliti in fase di **Analisi di Dettaglio** sia totale.

2.7.2.4 Test di regressione

Consiste nell'eseguire nuovamente i test di unità e di integrazione su componenti software che hanno subito modifiche, in modo da controllare che i cambiamenti apportati non pregiudichino il funzionamento di componenti che non sono stati aggiornati e che precedentemente non erano soggetti ad errori.

2.7.2.5 Test di accettazione

È il test di collaudo del prodotto software che viene eseguito in presenza del proponente. Se tale collaudo viene superato positivamente si può procedere al rilascio ufficiale del prodotto sviluppato.

2.8 Misure e metriche

Il processo di verifica, per essere informativo, deve esse quantificabile. Le misure rilevate dal processo di verifica devono quindi essere basate su metriche stabilite a priori. Qualora ci fossero metriche incerte ed approssimate, grazie al ciclo di vita adottato, descritto nel *Piano di Progetto v1.2.0*, si miglioreranno in modo incrementale.
Essendo la natura delle metriche molto variabile, vi possono essere due tipologie di range:

- **Accettazione**: valori richiesti affinché il prodotto sia accettato;
- **Ottimale**: valori entro cui dovrebbe collocarsi la misurazione. Tale range non è vincolante, ma fortemente consigliato. Scostamenti da tali valori necessitano una verifica approfondita.

2.8.1 Metriche per i processi

Come metrica per i processi si è scelto di utilizzare indici che ne analizzino i costi e tempi.
Tali indici vengono utilizzati anche per mantenere il controllo sui processi durante il loro svolgimento. Sono quindi descritti nel *Piano di Progetto v1.2.0*.

2.8.1.1 Schedule Variance (SV)

Indica se si è in linea, in anticipo o in ritardo rispetto alla pianificazione temporale delle attività nella baseline_G.
È un indicatore di efficacia.
Se $\text{SV}_G > 0$ significa che il gruppo di lavoro sta producendo con maggior velocità rispetto a quanto pianificato, viceversa se negativo.
Essendo stati inseriti slack durante la pianificazione della baseline_G dei processi, il valore di tale indice è inizialmente positivo. Tale scelta è descritta nel *Piano di Progetto v1.2.0*.

2.8.1.2 Budget Variance (BV)

Indica se alla data corrente si è speso di più o di meno rispetto a quanto pianificato.
È un indicatore che ha un valore contabile e finanziario.
Se $\text{BV}_G > 0$ significa che l'attuazione del progetto sta consumando il proprio budget con minor velocità rispetto a quanto pianificato, viceversa se negativo.

2.8.2 Metriche per i documenti

Come metrica per i documenti redatti si è scelto di utilizzare l'*indice di leggibilità*.
Vi sono a disposizione molti indici di leggibilità, ma i più importanti sono per la lingua inglese. Si è deciso quindi di adottare un indice di leggibilità per la lingua italiana.

2.8.2.1 Gulpease

L'indice Gulpease è un indice di leggibilità di un testo tarato sulla lingua italiana. Rispetto ad altri ha il vantaggio di utilizzare la lunghezza delle parole in lettere anziché in sillabe, semplificandone il calcolo automatico. Permette di misurare la complessità dello stile di un documento.
L'indice Gulpease considera due variabili linguistiche: la lunghezza della parola e la lunghezza della frase rispetto al numero delle lettere.
L'indice è calcolato secondo la seguente formula:

$$89 + \frac{300 * (numero\ delle\ frasi) - 10 * (numero\ delle\ lettere)}{numero\ delle\ parole}$$

I risultati sono compresi tra 0 e 100, dove il valore 100 indica la leggibilità più alta e 0 la leggibilità più bassa. In generale risulta che testi con un indice:

- Inferiore a 80 sono difficili da leggere per chi ha la licenza elementare;
- Inferiore a 60 sono difficili da leggere per chi ha la licenza media;
- Inferiore a 40 sono difficili da leggere per chi ha un diploma superiore.

Parametri utilizzati:

- Range-accettazione: [40-100];
- Range-ottimale: [50-100].

2.8.3 Metriche per il software

Si vorrebbe, per poter perseguire degli obbiettivi di qualità software, poter applicare le metriche che ora verranno descritte.
Questa sezione è da comprendersi come una *dichiarazione di intenti*, che verrà rivista nelle prossime revisioni.

2.8.3.1 Complessità ciclomatica

Utilizzata per misurare la complessità di funzioni, moduli, metodi o classi di un programma.
La complessità ciclomatica è calcolata utilizzando il grafo di controllo di flusso del programma: i nodi del grafo corrispondono a gruppi indivisibili di istruzioni, mentre gli archi connettono due nodi se il secondo gruppo di istruzioni può essere eseguito immediatamente dopo il primo gruppo.
Alti valori di complessità ciclomatica implicano una ridotta manutenibilità del codice.
Valori bassi di complessità ciclomatica potrebbero però determinare scarsa efficienza dei metodi (ad esempio: per ridurre il valore di tale metrica si eliminano blocchi condizionali che garantirebbero scorciatoie in casi di esecuzione comuni).
Questo paramento inoltre è un indice del carico di lavoro richiesto per il testing. Un modulo con complessità elevata richiede più testing di un modulo a complessità inferiore.

Parametri utilizzati:

- Range-accettazione: [1-15];
- Range-ottimale: [1-10][2].

[2] Il valore 10 come massimo di complessità ciclomatica fu raccomandato da T. J. McCabe, l'inventore di tale metrica

2.8.3.2 Numero di livelli di annidamento

Rappresenta il numero di livelli di annidamento dei metodi, cioè l'inserimento di una struttura di controllo all'interno di un'altra.
Un valore elevato di tale indice implica un'alta complessità ed un basso livello di astrazione del codice.

Parametri utilizzati:

- Range-accettazione: [1-6];
- Range-ottimale: [1-3].

2.8.3.3 Attributi per classe

Un numero elevato di attributi interni ad una classe potrebbe evidenziare la necessità di suddividere la classe in più classi da mettere in relazione tra di loro, seguendo il principio dell'incapsulamento.
Un alto valore di questo attributo indica un possibile errore di progettazione.

Parametri utilizzati:

- Range-accettazione: [0-16];
- Range-ottimale: [3-8].

2.8.3.4 Numero di parametri per metodo

Un numero elevato di parametri per un metodo potrebbe evidenziare la necessità di ridurre le funzionalità associate a tale metodo.
Un alto valore di questo attributo, indica un possibile errore di progettazione.

Parametri utilizzati:

- Range-accettazione: [0-8];
- Range-ottimale: [0-4].

2.8.3.5 Linee di codice per linee di commento

Indica il rapporto tra linee di codice e linee di commento.
È utile per stimare la manutenibilità del codice.
Parametri utilizzati:

- Range-accettazione: [>0.25];
- Range-ottimale: [>0.30][3].

[3]Il valore 0.30 è ricavato da rapporto 22/78. I valori utilizzati nel rapporto derivano dalle medie di Ohloh https://www.ohloh.net/p/firefox/factoids#FactoidCommentsLow

2.8.3.6 Flusso di informazioni

Misura il flusso di informazioni come suggerito da S. Henry e D. Kafura.
Definiti:

- Fan-in: numero di moduli che passano informazioni dentro al modulo in esame;
- Fan-out: numero di moduli a cui il modulo in esame passa informazioni.

Il valore è calcolato come:

$$(lunghezzafunzione)^2 \times fan\text{-}in \times fan\text{-}out$$

2.8.3.7 Accoppiamento

Accoppiamento Afferente Indica il numero di classi esterne ad un package che dipendono da classi interne ad esso.
Un alto valore di tale indice implica un alto grado di dipendenza del resto del software dal package.
Alti valori di tale indice non implicano necessariamente una progettazione errata o di bassa qualità, ma possono rappresentare la criticità del package in esame. Tale indice può quindi essere utilizzato per evidenziare la robustezza richiesta dal package.
Contrariamente, un valore eccessivamente basso poterebbe essere segnale di un package che fornisce poche funzionalità. Tale package potrebbe essere scarsamente utile.

Accoppiamento Efferente Indica il numero di classi interne al package che dipendono da classi esterne ad esso.
Mantenendo un basso valore di tale indice, è possibile mantenere il package in grado di garantire funzionalità di base indipendentemente dal resto del sistema.
I valori di range di tale indice verranno scelti durante la progettazione di dettaglio.

2.8.3.8 Copertura del codice

Indica la percentuale di istruzioni che sono eseguite durante i test.
Maggiore è la percentuale di istruzioni coperte dai test eseguiti, maggiore sarà la probabilità che le componenti testate abbiano una ridotta quantità di errori.
Il valore di tale indice può essere abbassato da metodi molto semplici che non richiedono testing. Esempi di questi metodi sono: get e set.

Parametri utilizzati:

- Range-accettazione: [42%-100%];
- Range-ottimale: [65%-100%].

2.9 Metodi

Vengono qui esplicitate le procedure con cui si eseguono l'analisi statica e dinamica per la verifica dei documenti.

2.9.1 Analisi dei processi

Per eseguire la verifica dei processi compiuti è necessario seguire il seguente protocollo:

1. **Controllo delle metriche**
   Alla conclusione di ogni fase del progetto, per ogni macro-attività, definita nel *Piano di Progetto v1.2.0*, si calcolano gli indici definiti nella sezione 2.8.1. Al fine di avere un indice complessivo di fase dovrà essere inoltre calcolato il valore medio di tali indici.
   Tali indici verranno calcolati utilizzando in modo complementare il software per la pianificazione ed il sistema di ticketing descritti nelle *Norme di Progetto v1.2.0*;

2. **Analisi grafico PDCA**
   Avendo a disposizione il grafico PDCA della fase del progetto in esame, verranno tratte delle considerazioni di carattere generale sullo svolgimento dei processi. Tale grafico viene utilizzato e descritto nel *Piano di Progetto v1.2.0*.
   Dal grafico si possono identificare in modo visuale, quindi generico e non numericamente quantificabile, dei dati sulla fase in esame:

   - Errori di pianificazione, rappresentati da variazioni delle attività nello stato Plan;
   - Velocità con cui il gruppo di lavoro porta avanti i processi tra i vari stati del ciclo PDCA_G.

2.9.2 Analisi dei documenti

Per eseguire un'accurata verifica dei documenti redatti è necessario seguire il seguente protocollo:

1. **Controllo sintattico e del periodo**
   Utilizzando TeXstudio e GNU Aspell vengono evidenziati e corretti gli errori di grammatica più evidenti. Gli errori di sintassi, di sostituzione di lettere che provocano la creazione di parole grammaticalmente corrette ma sbagliate nel contesto ed i periodi di difficile comprensione necessitano dell'intervento di un verificatore umano. Per questa ragione ciascun documento dovrà essere sottoposto ad un walkthrough da parte dei verificatori per individuare tali errori;

2. **Rispetto delle norme di progetto**
   Nelle *Norme di Progetto v1.2.0* sono definite norme tipografiche di carattere generale. Impongono una struttura dei documenti che non può essere verificata in maniera automatica. Per la verifica di alcune norme si possono utilizzare degli strumenti automatici, descritti nella sezione 2.6. La verifica delle norme per cui non è stato definito uno strumento automatico richiede che i *Verificatori* eseguano inspection sul rispetto di quelle norme in ciascun documento;

3. **Lista di controllo**
   Il *Verificatore* dovrà utilizzare la lista di controllo per i documenti, descritta nella sezione 2.7.1.2, e verificare che gli errori tipici non siano presenti;

4. **Verifica delle proprietà di glossario**
   Seguendo le *Norme di Progetto v1.2.0* nella redazione del *Glossario* è possibile automatizzare tale verifica. Utilizzando l'apposito script, vengono individuate tutte le segnalazioni di una parola del glossario in un documento in maniera automatica,

ed applicato l'opportuno stile.
Il *Verificatore* dovrà quindi controllare che l'output finale sia conforme a quanto precisato nelle *Norme di Progetto v1.2.0* relativamente al *Glossario*;

5. **Calcolo dell'indice Gulpease**
   Su ogni documento redatto il *Verificatore* deve calcolare l'indice di leggibilità.
   Nel caso in cui l'indice risultasse troppo basso, sarà necessario eseguire un walkthrough del documento alla ricerca delle frasi troppo lunghe o complesse.
   Per il calcolo di tale indice vi è a disposizione uno strumento automatico;

6. **Miglioramento processo**
   Per avere un miglioramento del processo di verifica, quando i *Verificatori* eseguono walkthrough di un documento, dovranno riportare gli errori più frequentemente trovati. Grazie a tale pratica sarà possibile eseguire inspection su tali errori nelle verifiche future.

3 Gestione amministrativa della revisione

3.1 Comunicazione e risoluzione di anomalie

Una *anomalia* corrisponde a:

- Violazione delle norme tipografiche da parte di un documento;
- Uscita dal range di accettazione degli indici di misurazione, descritti nella sezione 2.8;
- Incongruenza del prodotto con funzionalità indicate nell'analisi dei requisiti;
- Incongruenza del codice con il design del prodotto.

Nel caso in cui un *Verificatore* individui un'anomalia, dovrà aprire un ticket nel sistema di ticketing secondo le modalità specificate nelle *Norme di Progetto v1.2.0*.

3.2 Procedure di controllo di qualità di processo

La qualità dei processi verrà garantita dall'applicazione del principio PDCA, descritto nella sezione 2.1.1. Grazie a tale principio, sarà possibile garantire un *miglioramento continuo* della qualità di tutti i processi, inclusa la verifica, e con conseguente miglioramento dei prodotti risultanti.

Per avere controllo dei processi, e conseguentemente qualità, è necessario che:

- I processi siano pianificati dettagliatamente;
- Nella pianificazione siano ripartite in modo chiaro le risorse;
- Vi sia controllo sui processi.

L'attuazione di tali punti è descritta dettagliatamente nel *Piano di Progetto v1.2.0*.

La qualità dei processi viene inoltre monitorata mediante l'analisi costante della qualità del prodotto. Un prodotto di bassa qualità indica un processo da migliorare.

Per quantificare la qualità dei processi si possono inoltre utilizzare le metriche descritte nella sezione 2.8.1.

3.3 Procedure di controllo di qualità di prodotto

Il controllo di qualità del prodotto verrà garantito da:

- **Quality Assurance**: insieme di attività realizzate per garantire il raggiungimento degli obbiettivi di qualità. Prevede l'attuazione di tecniche di analisi statica e dinamica, descritte nella sezione 2.7;
- **Verifica**: processo che determina se l'output di una fase è consistente, completo e corretto. La verifica andrà eseguita costantemente durante l'intera durata del progetto. I risultati delle attività di verifica eseguiti nelle varie fasi del progetto sono riportati nella sezione 4;
- **Validazione**: conferma in modo oggettivo che il sistema risponda ai requisiti.

4 Resoconto delle attività di verifica

4.1 Riassunto delle attività di verifica

4.1.1 Revisione dei Requisiti

Nel periodo antecedente la consegna di tale revisione sono stati verificati i documenti ed i processi.

I **documenti** sono stati verificati applicando la procedura descritta nella sezione 2.9.2. L'*analisi statica* è stata applicata secondo i criteri e le modalità indicate nella sezione 2.7.1. Effettuando *walkthrough* sono stati riscontrati degli errori. Sono state quindi avviate le procedure per la segnalazione, descritte nella sezione 3.1.
Noti gli errori, si è provveduto a:

- Correggere le imperfezioni rilevate;
- Segnalare gli errori frequenti, e riportarli nella sezione 2.7.1.2 relativa al metodo di *inspection*. Si è quindi applicato il ciclo $PDCA_G$ per rendere più efficiente ed efficace il processo di verifica.

È stata in seguito applicata l'*inspection* utilizzando la lista di controllo stilata durante la verifica dei documenti precedentemente verificati, ponendo particolare attenzione ai grafici dei casi d'uso.
Si sono poi calcolate per i documenti le *metriche* descritte nel punto 2.8.2.
Il *tracciamento* (requisiti - fonti, use-case - requisiti) è stato effettuato tramite l'applicativo Tracy.
I **processi** sono stati verificati applicando la procedura descritta nella sezione 2.9.1.
Si sono calcolate le *metriche* di processo, descritte nella sezione 2.8.1. Sono quindi stati riportati i valori di BV_G ed SV_G, ed è stato riportato e valutato il grafico PDCA aggiornato al termine di tutti i processi della fase.

4.2 Tracciamento componenti - requisiti

4.3 Dettaglio delle verifiche tramite analisi

4.3.1 Processi

Vengono qui riportati i valori degli indici indice SV_G e BV_G, descritti nella sezione 2.9.1, per macro-attività della fase di **Analisi**.

Macro-attività	SV	BV
Norme di Progetto	0€	20€
Studio Fattibilità	0€	130€
Analisi Requisiti	105€	-105€
Piano di Progetto	0€	-135€
Piano di Qualifica	0€	0€
Glossario	0€	0€

Tabella 2: Esiti verifica processi, Analisi

Complessivamente, la fase di **Analisi** ha:

- SV_G = 105€;

- BV_G = -90€.

Da valori di tali indici possiamo dedurre che:

- Grazie agli slack inseriti durante la pianificazione, le attività sono state svolte entro i tempi pianificati, descritti nel *Piano di Progetto v1.2.0*;
- Il costo della fase non si è discostato molto dal pianificato, descritto nel *Piano di Progetto v1.2.0*. Tale ammortamento si può imputare a valori positivi di alcune macro-attività, come lo Studio Fattibilità, che hanno scaricato il costo aggiuntivo apportato in altre, come il Piano di Progetto.

Il grafico PDCA della fase di **Analisi** è:

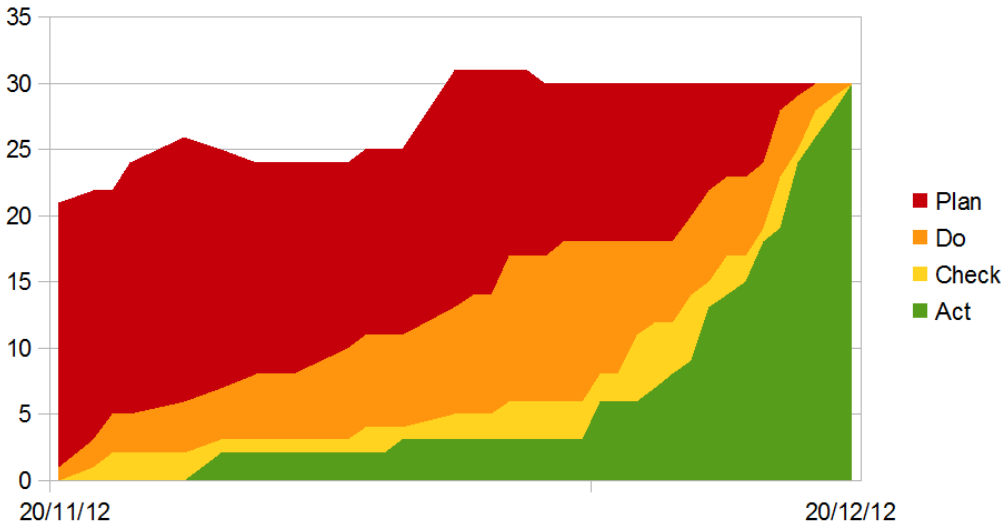

Figura 4: Grafico PDCA, fase di Analisi

Da tale grafico si può asserire che:

- Sono visibili dei mutamenti dei processi pianificati. Tali mutamenti sono imputabili ad errori di pianificazione. Tali errori sono causati dall'inesperienza del gruppo di lavoro;
- A tre quarti della fase è chiaramente visibile un plateau nell'avanzamento dei processi. Tale rallentamento è imputabile alla sovrapposizione degli impegni universitari dei componenti del gruppo con la realizzazione del progetto;
- Alla fine della fase è visibile una accelerazione nell'avanzamento. Tale accelerazione è conseguenza dal totale impegno del gruppo nel portare avanti i processi.

4.3.2 Documenti

Vengono qui riportati i valori dell'indice Gulpease per ogni documento durante la fase di **Analisi**. Un documento è considerato valido soltanto se rispetta le metriche descritte su 2.8.2.1.

Documento	Valore indice	Esito
Piano di Progetto v1.2.0	63.30	superato
Norme di Progetto v1.2.0	65.41	superato
Analisi dei Requisiti v1.2.0	66.78[4]	superato
Piano di Qualifica v1.2.0	56.19	superato
Studio di Fattibilita v1.2.0	60.26	superato
Glossario v1.2.0	52.03	superato

Tabella 3: Esiti verifica documenti, Analisi

Come si può notare dalla tabella, tutti gli indici Gulpease dei documenti rientrano nel range ottimale precedentemente definito e quindi i documenti redatti hanno raggiunto la leggibilità desiderata.

4.4 Dettaglio dell'esito delle revisioni

Durante lo sviluppo del progetto vi saranno quattro revisioni del committente a cui sottoporsi.
Il committente segnalerà le problematiche riscontrate fornendo una valutazione globale dell'andamento del progetto ed una dettagliata per ciascun documento.
Resi noti i problemi e le criticità del lavoro svolto, sarà possibile correggere quanto indicato. Dopo aver eseguito le opportune correzioni, si potrà procedere su una base verificata e corretta.

[4]Tale valore è stato ricavato eliminando le tabelle dal calcolo. Includendo tali tabelle il valore è pari 100 in quanto sono presenti molti identificatori che sfalsano il valore reale.