Don't Panic

3DMob: Grafica 3D su device mobili

Piano di Qualifica

Informazioni sul documento

Versione	5.2.0
Redazione	Sciarrone Riccardo Basaglia Mattia
Verifica	Pezzutti Marco Lain Daniele
Responsabile	Busato Luca
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	Ruolo	Data	Versione
Approvazione documento	Busato Luca	Responsabile	2013-03-19	5.2.0
Verifica documento	Pezzutti Marco	Verificatore	2013-03-18	5.1.1
Verifica documento	Lain Daniele	Verificatore	2013-03-18	5.1.0
Strutturato resoconto delle attività di verifica	Sciarrone Riccardo	Verificatore	2013-03-17	5.0.2
Stesura sezione relativa ai test di validazione	Basaglia Mattia	Progettista	2013-03-15	5.0.1
Approvazione documento	Rampazzo Federico	Responsabile	2013-02-27	4.2.0
Verifica documento	Busato Luca	Verificatore	2013-02-26	4.1.1
Verifica documento	Lain Daniele	Verificatore	2013-02-26	4.1.0
Strutturato resoconto delle attività di verifica	Cesarato Fabio	Verificatore	2013-01-25	4.0.2
Stesura sezione relativa ai test di sistema	Sciarrone Riccardo	Progettista	2013-01-17	4.0.1
Revisione correttiva struttura su segnalazione del committente	Pezzutti Marco	Analista	2013-01-07	4.0.0
Approvazione documento	Pezzutti Marco	Responsabile	2013-01-29	3.2.0
Verifica documento	Sciarrone Riccardo	Verificatore	2013-01-28	3.1.1
Verifica documento	Cesarato Fabio	Verificatore	2013-01-27	3.1.0
Strutturato resoconto delle attività di verifica	Basaglia Mattia	Verificatore	2013-01-26	3.0.3
Stesura pianificazione ed esecuzione del collaudo	Basaglia Mattia	Progettista	2013-01-25	3.0.2
Incremento sezione strategia di verifica	Lain Daniele	Responsabile	2013-01-25	3.0.1
Revisione correttiva struttura e contenuti basata su segnalazione del committente	Busato Luca	Analista	2013-01-12	3.0.0
Approvazione documento	Busato Luca	Responsabile	2013-01-08	2.2.0
Verifica documento	Sciarrone Riccardo	Verificatore	2013-01-07	2.1.0
Strutturato resoconto delle attività di verifica	Lain Daniele	Verificatore	2013-01-06	2.0.2
Incremento liste di controllo	Lain Daniele	Verificatore	2013-01-05	2.0.1

Incremento sezione misure e metriche	Rampazzo Federico	Analista	2013-01-03	2.0.0
Approvazione documento	Cesarato Fabio	Responsabile	2012-12-18	1.2.0
Verifica documento	Lain Daniele	Verificatore	2012-12-16	1.1.1
Verifica documento	Busato Luca	Verificatore	2012-12-15	1.1.0
Strutturato resoconto delle attività di verifica	Pezzutti Marco	Verificatore	2012-12-12	1.0.7
Stesura sezione analisi statica e dinamica	Cesarato Fabio	Analista	2012-12-01	1.0.6
Stesura gestione amministrativa della revisione	Pezzutti Marco	Amministratore	2012-11-30	1.0.5
Stesura responsabilità e risorse	Cesarato Fabio	Responsabile	2012-11-30	1.0.4
Stesura organizzazione e pianificazione strategica	Cesarato Fabio	Responsabile	2012-11-29	1.0.3
Descritti strumenti, metriche e metodi	Pezzutti Marco	Amministratore	2012-11-28	1.0.2
Stesura sezioni introduzione e obiettivi di qualità	Cesarato Fabio	Responsabile	2012-11-27	1.0.1
Creazione struttura principale	Cesarato Fabio	Analista	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 v5.2.0*.
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 v5.2.0*;
- **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 v5.2.0*;
- **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

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 deciso di adottare lo standard ISO$_G$/IEC$_G$ 15504[1] denominato SPICE[2] il quale fornisce gli strumenti necessari a valutare l'idoneità di questi ultimi.
Per applicare correttamente questo modello si deve utilizzare il ciclo di Deming$_G$[3] (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.

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[4] è stato redatto con lo scopo di descrivere questi obiettivi e delineare delle metriche capaci di misurare il raggiungimento di tali obiettivi.

2.2 Procedure di controllo di qualità di processo

La qualità dei processi verrà garantita dall'applicazione del principio PDCA, descritto nella sezione A.2. Grazie a tale principio, sarà possibile garantire un *miglioramento continuo* della qualità di tutti i processi, inclusa la verifica, e come diretta conseguenza si otterrà il 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 v5.2.0*.

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 2.9.1.

2.3 Procedure di controllo di qualità di prodotto

Il controllo di qualità del prodotto verrà garantito da:

[1] Per approfondimenti consultare la sezione A.1 in appendice A
[2] Software Process Improvement Capability dEtermination
[3] Per approfondimenti consultare la sezione A.2 in appendice A
[4] Per approfondimenti consultare la sezione A.3 in appendice A

- **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 2.8;
- **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 nell'appendice C;
- **Validazione**: conferma in modo oggettivo che il sistema risponda ai requisiti.

2.4 Organizzazione

L'organizzazione della strategia di verifica si basa sull'attuazione di attività di verifica per ogni processo attuato. Per ogni processo realizzato viene verificata la qualità del processo stesso e la qualità dell'eventuale prodotto ottenuto da esso.
Ognuna delle 5 fasi del progetto descritte nel *Piano di Progetto v5.2.0* 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 *Norme di Progetto v5.2.0* sui documenti prodotti e sui processi attuati. L'attuazione di tali attività di verifica, essendo antecedenti alla **Revisione dei Requisiti**, sono descritte nell'appendice C.1.1;
- **Analisi di Dettaglio:** in tale fase si devono verificare i processi che portano all'incremento dei documenti redatti nella precedente fase ed i relativi prodotti seguendo le procedure descritte nelle *Norme di Progetto v5.2.0*. L'attuazione di tali attività di verifica, essendo antecedenti alla **Revisione di Progetto**, sono descritte nell'appendice C.1.2;
- **Progettazione Architetturale:** in tale fase si devono verificare i processi che portano all'incremento dei documenti redatti nella precedente fase ed i relativi prodotti, oltre ai prodotti e processi attuati per l'attività di progettazione architetturale seguendo le procedure descritte nelle *Norme di Progetto v5.2.0*. L'attuazione di tali attività di verifica, essendo antecedenti alla **Revisione di Progetto**, sono descritte nell'appendice C.1.2.

Essendo la redazione dei documenti un'attività predominante e costantemente presente nel progredire del progetto, il processo di verifica si scorpora in due attività per garantirne una maggiore qualità. In ogni documento viene inoltre incluso il diario delle modifiche che permette di mantenere uno storico delle attività svolte e delle relative responsabilità.

2.5 Pianificazione strategica e temporale

Avendo l'obiettivo di rispettare le scadenze fissate nel *Piano di Progetto v5.2.0*, 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.
Le metodologie da seguire per l'individuazione e la correzione degli errori sono descritte nelle *Norme di Progetto v5.2.0*.

Ogni attività di redazione dei documenti e di codifica deve essere preceduta da uno studio preliminare sulla struttura e sui contenuti degli stessi. Tale attività 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.

2.6 Responsabilità

Per garantire che il processo di verifica sia efficace e sistematico vengono attribuite delle responsabilità a degli specifici ruoli di progetto.
I ruoli che detengono le responsabilità del processo di verifica sono il *Responsabile di Progetto* ed i *Verificatori*. La suddivisione dei compiti e le modalità di attuazione degli stessi sono definiti nelle *Norme di Progetto v5.2.0*.

2.7 Risorse

Per assicurare che gli obiettivi qualitativi vengano raggiunti è necessario l'utilizzo di risorse sia umane che tecnologiche. Coloro che detengono la responsabilità maggiore per l'attività di verifica e validazione sono il *Responsabile di Progetto* e il *Verificatore*. Per una dettagliata descrizione dei ruoli e delle loro responsabilità fare riferimento alle *Norme di Progetto*.
Per risorse tecniche e tecnologiche sono da intendersi tutti gli strumenti software e hardware che il gruppo intende utilizzare per attuare le attività di verifica su processi e prodotti. Affinché il lavoro dei *Verificatori* venga agevolato si sono predisposti numerosi strumenti automatici che eseguono controlli sistematici sui prodotti generati. Tali strumenti sono descritti in modo accurato nelle *Norme di Progetto*.

2.8 Tecniche di analisi

2.8.1 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.

2.8.1.1 Walkthrough

Si svolge effettuando una lettura critica a largo spettro. È una tecnica che viene utilizzata soprattutto nelle prime attività 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.

2.8.1.2 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 in appendice delle *Norme di Progetto v5.2.0*.

2.8.2 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*.

2.8.2.1 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.

2.8.2.2 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 al posto di quelle che non sono ancora state sviluppate, in modo da non influenzare negativamente l'esito dell'analisi.

2.8.2.3 Test di sistema

Consiste nella validazione del prodotto software nel momento in cui 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.

2.8.2.4 Test di regressione

Consiste nell'eseguire nuovamente i test riguardanti le 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.
Tale operazione è aiutata dal tracciamento, che permette di individuare e ripetere facilmente i test di unità, integrazione ed eventualmente sistema che sono stati potenzialmente influenzati dalla modifica.

2.8.2.5 Test di accettazione

È il 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.

2.9 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 v5.2.0*, 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.

2.9.1 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 v5.2.0*.

2.9.1.1 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 v5.2.0*.
Parametri utilizzati:

- Range-accettazione: $[\geq -(\textit{costo preventivo } fase \times 5\%)]$[5];
- Range-ottimale: $[\geq 0]$.

2.9.1.2 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.
Parametri utilizzati:

- Range-accettazione: $[\geq -(\textit{costo preventivo } fase \times 10\%)]$[6];
- Range-ottimale: $[\geq 0]$.

2.9.2 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.

2.9.2.1 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.

[5] L'intervallo è stato calcolato basandosi sulla rigidità dei tempi di consegna e sull'inesperienza del gruppo. La percentuale deriva da convenzioni comuni e dati empirici descritti in http://office.microsoft.com/en-us/project-help/determine-the-right-threshold-for-project-cost-and-schedule-variances-HA010173335.aspx

[6] L'intervallo è stato calcolato valutando l'inesperienza del gruppo. La percentuale deriva da convenzioni comuni e dati empirici descritti http://office.microsoft.com/en-us/project-help/determine-the-right-threshold-for-project-cost-and-schedule-variances-HA010173335.aspx

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: $[50 - 100]$.

2.9.3 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.

2.9.3.1 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: $[1 - 10]$[7].

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

2.9.3.2 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: $[1 - 3]$.

2.9.3.3 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: $[3 - 8]$.

2.9.3.4 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: $[0 - 4]$.

2.9.3.5 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: $[> 0.30]$[8].

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

2.9.3.6 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:

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

2.9.3.7 Accoppiamento

- **Accoppiamento afferente**: indica il numero di classi esterne ad un package$_G$ 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$_G$.
  Alti valori di tale indice non implicano necessariamente una progettazione errata o di bassa qualità, ma possono rappresentare la criticità del package$_G$ in esame. Tale indice può quindi essere utilizzato per evidenziare la robustezza richiesta dal package$_G$.
  Contrariamente, un valore eccessivamente basso poterebbe essere segnale di un package$_G$ che fornisce poche funzionalità. Tale package$_G$ potrebbe essere scarsamente utile;
- **Accoppiamento efferente**: indica il numero di classi interne al package$_G$ che dipendono da classi esterne ad esso.
  Mantenendo un basso valore di tale indice, è possibile mantenere il package$_G$ in grado di garantire funzionalità di base indipendentemente dal resto del sistema.

2.9.3.8 Instabilità

Metrica proposta da Robert C. Martin[9] che si prefigge di misurare l'instabilità dei package$_G$ del sistema.
La stabilità di un package$_G$ indica la possibilità di effettuare modifiche a tale package$_G$ senza influenzarne altri all'interno dell'applicazione. Tale indice è strettamente legato all'accoppiamento efferente ed afferente e viene calcolato dalla seguente formula:

$$I = \frac{Ce}{Ca + Ce}$$

dove:

- **Ce:** accoppiamento efferente;
- **Ca:** accoppiamento afferente.

Parametri utilizzati:[10]

- Range-accettazione: $[0.0 - 0.8]$;
- Range-ottimale: $[0.0 - 0.3]$.

[9] http://www.objectmentor.com/omTeam/martin_r.html
[10] I range indicati sono stati ricavati da http://staff.unak.is/andy/StaticAnalysis0809/metrics/i.html

2.9.3.9 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%].

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 2.9;
- 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 v5.2.0*.

A Standard di qualità

A.1 Standard ISO/IEC 15504

Lo standard ISO_G/IEC_G 15504 descrive come ogni processo debba essere 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.

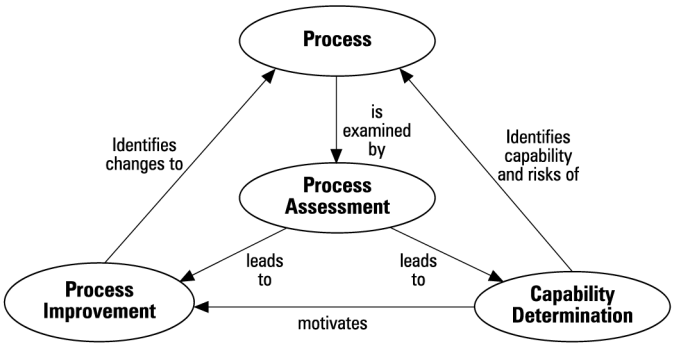

Figura 1: Modello SPY - Software Process Assessment and Improvement

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.
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%).

A.2 Ciclo di Deming

Il ciclo PDCA$_G$ 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.

A.3 Standard ISO/IEC 9126

Lo standard ISO$_G$/IEC$_G$ 9126 è stato redatto con lo scopo di descrivere gli obiettivi qualitativi di prodotto 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;

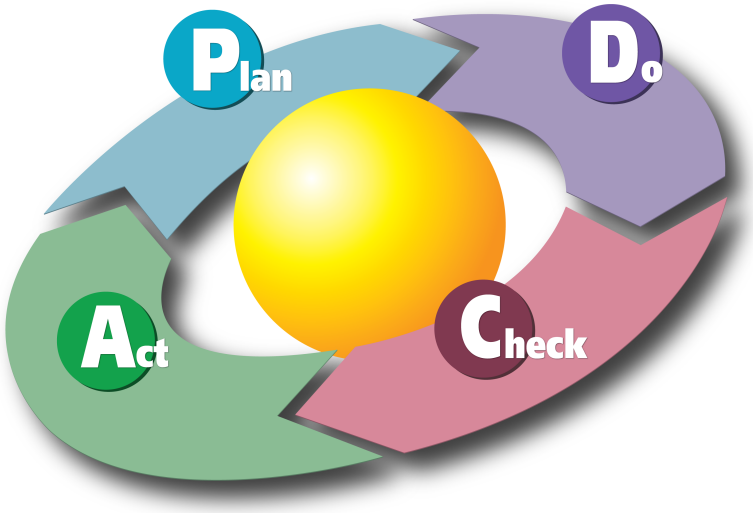

Figura 2: Ciclo PDCA

- **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:

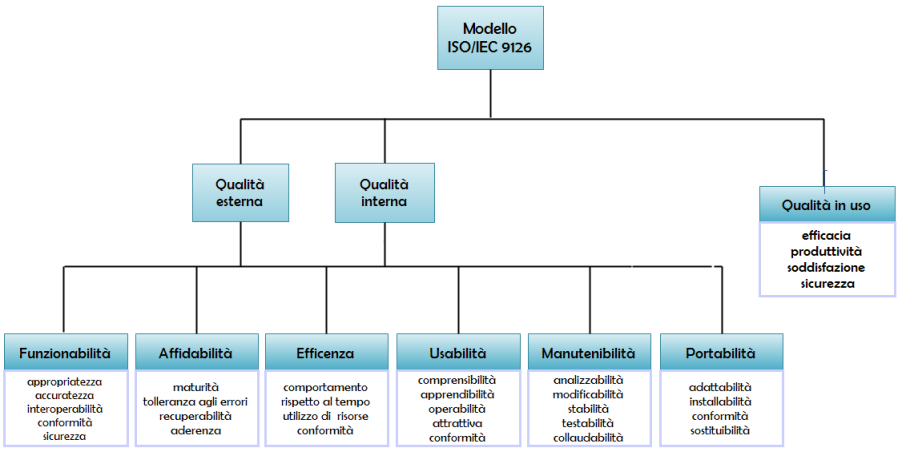

Figura 3: Caratteristiche qualitative modello ISO/IEC 9126

- **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;
  - **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.

B Pianificazione dei test

Si descrivono di seguito tutti i test di validazione, sistema ed integrazione previsti, prevedendo un aggiornamento futuro per i test di unità. Per le tempistiche di esecuzione dei test si faccia riferimento al *Piano di Progetto v5.2.0*.
Nelle tabelle sottostanti lo stato dei test **N.I.** è da intendersi come non applicato in quanto tali test saranno applicati successivamente, come descritto nel *Piano di Progetto v5.2.0*.

B.1 Test di sistema

In questa sezione vengono descritti i test di sistema che permettono di verificare il comportamento dinamico del sistema completo rispetto ai requisiti descritti nell'*Analisi dei Requisiti v4.2.0*.
I test di sistema riportati sono quelli relativi ai requisiti software individuati e pertanto meritevoli di un test.

B.1.1 Descrizione dei test di sistema

Test	Descrizione	Stato	Requisito
TS1	Viene verificato che il sistema permetta di salvare la scena$_G$ corrente nei vari formati implementati	success	R0F1
TS1.3	Verifica per corretta implementazione dei test correlati ai requisiti figli	success	R0F1.3
TS1.3.1	Viene verificato che il sistema mantenga le normali e i vertici del solido confrontando i vertici di partenza con quelli ottenuti	success	R0F1.3.1
TS1.3.2	Viene verificato che il sistema mantenga le caratteristiche delle texture presenti nell'oggetto confrontando l'istogramma della texture di partenza con quella ottenuta	success	R0F1.3.2
TS1.3.3	Viene verificato che il sistema esporti i parametri dei materiali presenti nel modello importato confrontando l'istogramma di partenza con quello ottenuto	success	R0F1.3.3
TS1.3.4	Viene verificato che il sistema esporti file che rispettino le sorgenti di luce confrontando il modello importato con quello esportato	success	R0F1.3.4
TS1.3.5	Viene verificato che il sistema esporti file che mantengano le animazioni confrontando il modello importato con quello esportato	success	R1F1.3.5

TS2	Viene verificato che il sistema esporti file che rispettino i limiti dell'iPhone 4S confrontando il modello importato con quello esportato	success	R0V2
TS2.1	Viene verificato che il sistema esporti di default file che non contengano più di 8 luci controllando il file esportato	success	R0V2.1
TS2.2	Viene verificato che il sistema esporti di default file che non contengano texture di larghezza e altezza superiore a 4096 pixel controllando il file esportato	success	R0V2.2
TS6	Viene verificato che il sistema funzioni su Ubuntu$_G$ 12.04 LTS a 32bit installando il prodotto software su tale sistema	success	R2P6
TS7.2.1	Viene verificato che il sistema notifichi gli errori nell'apertura di file importando un file corrotto o con formato non supportato	success	R2F7.2.1
TS7.2.2	Viene verificato che il sistema notifichi gli errori relativi a malfunzionamenti facendone un uso non atteso	success	R2F7.2.2
TS7.2.3	Viene verificato che il sistema notifichi gli errori relativi alla perdita di informazioni caricando un file che ecceda i limiti di importazione	success	R2F7.2.3
TS10	Viene verificato che il sistema funzioni su Ubuntu$_G$ 12.04 LTS a 64bit installando il prodotto software su tale sistema	success	R2P10

Tabella 2: Tabella di tracciamento test di sistema / requisiti

B.2 Test di integrazione

In questa sezione vengono descritti i test di integrazione, da utilizzare per i vari componenti descritti nella progettazione ad alto livello, che permettono di verificare la corretta integrazione ed il corretto flusso dei dati all'interno del sistema.
Si è deciso di utilizzare una strategia di integrazione incrementale che permette di sviluppare e verificare le componenti in parallelo.
Assemblando le componenti in modo incrementale i difetti rilevati da un test sono da attribuirsi, con maggior probabilità, all'ultima parte aggiunta e si rende ogni passo di integrazione reversibile consentendo di retrocedere verso uno stato noto e sicuro.
È stato utilizzato il metodo bottom-up$_G$ per poter integrare prima le parti con minore dipendenza funzionale e maggiore funzionalità che corrispondono ai componenti per requisiti obbligatori permettendo, in questo modo, di avere una versione funzionante delle parti obbligatorie dell'applicazione il prima possibile.
In questo modo i componenti legati a parte obbligatorie vengono testati più volte diminuendo la probabilità di presenza di errori in essi. Successivamente si risale l'albero delle dipendenze fino ad arrivare al componente di alto livello a cui saranno dedicati gli ultimi test.
Il diagramma seguente non rispetta il formalismo UML$_G$ 2.x ed è utilizzato per semplificare l'illustrazione della strategia di integrazione. Si possono notare, lungo l'albero, dei macro componenti che integrano due o più componenti di maggior dettaglio e che corrispondono a possibili rilasci di prototipi funzionanti.

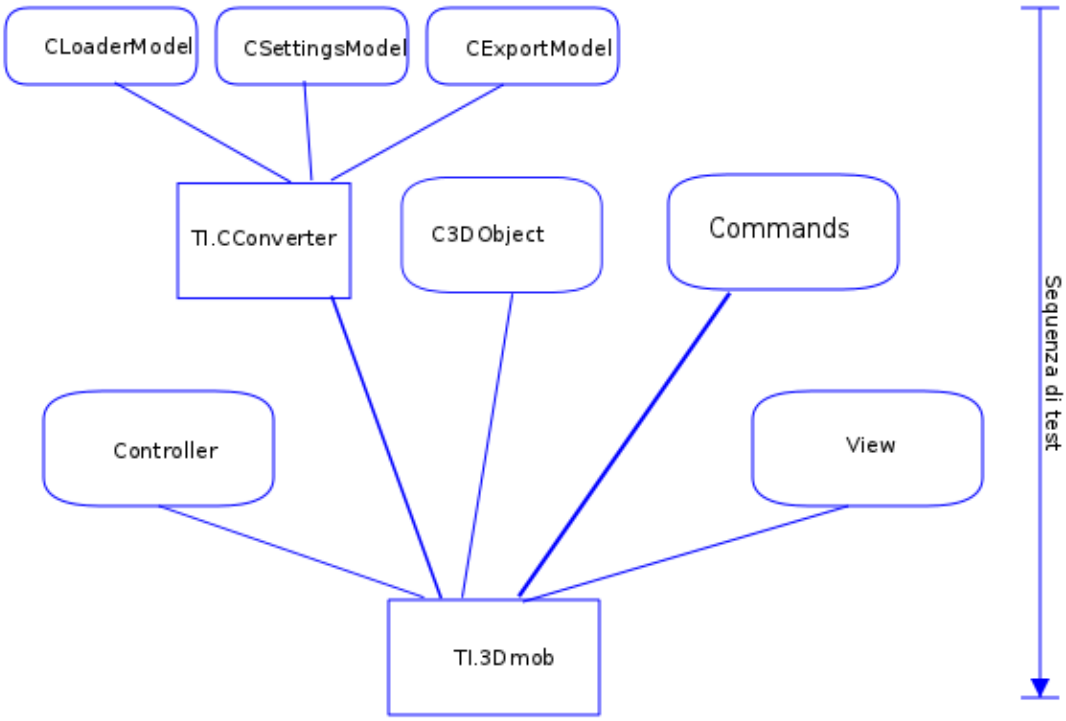

Figura 4: Diagramma informale della strategia di integrazione

B.2.1 Descrizione dei test di integrazione

Test	Descrizione	Componente	Stato
TI.DDDMob	Test di integrazione finale per Model, View e Controller	DDDMob	success
TI.CSettingsModel	**Impostazioni**: verifica il corretto funzionamento delle impostazioni dell'applicazione	CSettingsModel	success
TI.CExportModel	**Esportazione della scena**$_G$: verifica che l'operazione di esportazione nei vari formati produca i risultati attesi	CExportModel	success
TI.CLoaderModel	**Caricamento della scena**$_G$: verifica che l'operazione di caricamento della scena$_G$ avvenga correttamente	CLoaderModel	success
TI.C3DObject	**Operazioni sugli oggetti**: verifica che tutte le operazioni possibili sugli oggetti vengano eseguite correttamente	C3DObject	success
TI.Commands	**Controller dei comandi**: verifica l'interazione tra i comandi previsti	Commands	success
TI.CConverter	Test di integrazione per le funzionalità di editor di base	CConverter	success

Tabella 3: Tabella test di integrazione

B.2.2 Tracciamento componenti–test di integrazione

Componente	Test
DDDMob	TI.DDDMob
DDDMob::Model	Architettura del sistema
DDDMob::Model::Commands	TI.Commands
DDDMob::Model::CConverter	TI.CConverter
DDDMob::Model::CConverter::CSettingsModel	TI.CSettingsModel
DDDMob::Model::CConverter::CExportModel	TI.CExportModel
DDDMob::Model::CConverter::CLoaderModel	TI.CLoaderModel
DDDMob::Controller	Architettura del sistema
DDDMob::View	Architettura del sistema
DDDMob::View::CDockWidgets	Architettura del sistema
DDDMob::View::CMainWindow	Architettura del sistema
DDDMob::View::CSettingsWindow	Architettura del sistema
DDDMob::View::CErrorWindow	Architettura del sistema
DDDMob::View::CWidgetElement	Architettura del sistema
DDDMob::C3DObject	TI.C3DObject

Tabella 4: Tabella componente / test di integrazione

B.3 Test di unità

Di seguito vengono riportati i test di unità previsti per l'applicazione.

Test	Descrizione	Metodi	Stato
TU1	Si verifica che, passato il path di un file di test valido opportunamente creato ed il tipo dello stesso, la scena$_G$ 3D serializzata non risulti vuota. Si verifica anche che, dato un file di test non valido opportunamente creato, l'importazione fallisca	importFile() buildLights() buildMeshes() meshCreator() lightCreator() importLimits() getLightNumber() addObject() getAllObjects()	Success
TU2	Si verifica che, data una scena$_G$ non vuota ottenuta caricando un file valido nel sistema, dato un path ed un formato di esportazione, sia creato un file non vuoto sul disco	exportFile()	Success
TU3	Si verifica che data una scena$_G$, venga aggiunta una luce con valori di materiale predefiniti (diffusione, speculare ed emissione a bianco; lucentezza a 64 su 128; valore di emissione a 0) e alla posizione (0,0,0)	removeObject()	Success
TU4	Si verifica che, caricato un opportuno file di test contenente un oggetto noto, applicando una rotazione nota sull'oggetto i vertici si trovino nelle coordinate attese, a meno di errori numerici. Si richiede una precisione di 2 cifre decimali per il confronto dei vertici noti con i vertici ottenuti	setRotation() transformation()	Success
TU5	Si verifica che, caricato un opportuno file di test contenente un oggetto noto, applicando un ridimensionamento noto sull'oggetto i vertici si trovino nelle coordinate attese, a meno di errori numerici. Si richiede una precisione di 2 cifre decimali per il confronto dei vertici noti con i vertici ottenuti	setScale()	Success

TU6	Si verifica che, caricato un opportuno file di test contenente un oggetto noto, applicando una traslazione nota sull'oggetto i vertici si trovino nelle coordinate attese, a meno di errori numerici. Si richiede una precisione di 2 cifre decimali per il confronto dei vertici noti con i vertici ottenuti	setPosition()	Success
TU7	Si verifica che, caricato un opportuno file di test contenente un oggetto noto, modificando l'emissività dell'oggetto questa venga aggiornata correttamente. Si richiede una precisione di 2 cifre decimali per il confronto del parametro impostato con quello ottenuto	setEmission()	Success
TU8	Si verifica che, caricato un opportuno file di test contenente un oggetto noto, modificando il colore di diffusione$_G$ dell'oggetto questo venga aggiornata correttamente	setDiffuseColor()	Success
TU11	Si verifica che, caricato un opportuno file di test contenente un oggetto noto, modificando la tipologia della luce questa venga aggiornata correttamente	setLightType()	Success
TU12	Si verifica che, caricato un opportuno file di test contenente un oggetto noto, modificando la lucentezza dell'oggetto questo venga aggiornata correttamente. Si richiede una precisione di due cifre decimali per il confronto	setShininess()	Success

TU14	Si verifica che, dato il percorso del file da esportare, questo venga esportato mantenendo le caratteristiche della scena$_G$. Si richiede che tutti i valori di tutti gli oggetti contenuti all'interno della scena$_G$ vengano confrontati con gli oggetti della scena$_G$ stessa	exportFile()	Success
TU15	Si verifica che la richiesta di selezione di un device esistente imposti correttamente l'attributo interno selectedDevice, mentre la richiesta di un device inesistente emetta un errore	DeviceLimit() selectDevice()	Success
TU16	Si verifica che ad una prima chiamata il metodo crei un'istanza di Setting e ad una seconda chiamata ritorni l'istanza creata precedentemente	getSetting() Setting()	Success
TU17	Si verifica che, caricato un opportuno file di test contenente un oggetto noto, modificando il colore di emissione dell'oggetto questo venga aggiornata correttamente	setEmissionColor()	Success
TU18	Si verifica che, caricato un opportuno file di test contenente un oggetto noto, modificando il colore speculare$_G$ dell'oggetto questo venga aggiornata correttamente	setSpecularColor()	Success
TU19	Si verifica che, creato il comando ed aggiunto allo stack dei comandi, passato un nuovo elemento questo venga aggiunto alla scena$_G$, e che venga eliminato dopo l'undo del comando. Si provi l'aggiunta di una luce	CommandAddLight() CommandAddElement() undo() redo()	Success
TU20	Si verifica che, creato il comando ed aggiunto allo stack dei comandi, venga cambiato il colore di diffusione$_G$ dell'oggetto selezionato, e che venga ripristinato il vecchio colore dopo l'undo del comando	CommandColorDiffuse() redo() undo()	Success

TU21	Si verifica che, creato il comando ed aggiunto allo stack dei comandi, venga cambiato il colore di emissione dell'oggetto selezionato, e che venga ripristinato il vecchio colore dopo l'undo del comando	CommandColorEmission() undo() redo()	Success
TU22	Si verifica che, creato il comando ed aggiunto allo stack dei comandi, venga cambiato il colore speculare$_G$ dell'oggetto selezionato, e che venga ripristinato il vecchio colore dopo l'undo del comando	CommandColorSpecular() redo() undo()	Success
TU23	Si verifica che, creato il comando ed aggiunto allo stack dei comandi, venga cambiato il valore di emissione dell'oggetto selezionato, e che venga ripristinato il vecchio valore dopo l'undo del comando	CommandEmission() undo() redo() mergeWith()	Success
TU24	Si verifica che, creato il comando ed aggiunto allo stack dei comandi, venga cambiata la tipologia della luce selezionata, e che venga ripristinato il vecchio valore dopo l'undo del comando	CommandLightType() undo() redo()	Success
TU25	Si verifica che, creato il comando ed aggiunto allo stack dei comandi, venga cambiata la posizione dell'oggetto selezionato, e che venga ripristinato il vecchio valore dopo l'undo del comando	CommandPosition() redo() undo() mergeWith()	Success
TU26	Si verifica che, creato il comando ed aggiunto allo stack dei comandi, venga cambiata la rotazione dell'oggetto selezionato, e che venga ripristinato il vecchio valore dopo l'undo del comando	CommandRotation() redo() undo() mergeWith()	Success

TU27	Si verifica che, creato il comando ed aggiunto allo stack dei comandi, venga cambiata la scala dell'oggetto selezionato, e che venga ripristinato il vecchio valore dopo l'undo del comando	CommandScale() redo() undo() mergeWith()	Success
TU28	Si verifica che, creato il comando ed aggiunto allo stack dei comandi, venga cambiato il valore di lucentezza dell'oggetto selezionato, e che venga ripristinato il vecchio valore dopo l'undo del comando	CommandShininess() undo() redo() mergeWith()	Success
TU29	Si verifica che dato il nome di un oggetto contenuto nella scena$_G$ caricata venga ritornato il riferimento corretto, in caso contrario si verifica che venga restituito un riferimento nullo	selectByName()	Success

TU30	Architettura di sistema	slotRotation() slotScale() slotPosition() slotAddLight() slotColorDiffuse() slotColorSpecular() slotColorEmission() slotSelectObject() slotLightType() slotMeshShininess() slotEmission() slotSettings() slotOpen() slotSave() slotViewAbout() showError() slotViewSave() slotViewOpen() slotViewHelp() slotViewSettings() signalSettingsUpdate() signalUpdatedScene() signalError() Controller() WindowWithMenu() WindowWithDock() WindowWithStatus() exportFile() draw() CommandTransform() CommandScene() CommandEditObject() CommandEditMesh() start3DMob() CommandEditLight() id() CommandColor() id() id() id() id() update() View3D() LightWidget()	Success

TU31	Get-Set-Slot	setEmissionColor() getPosition() getRotation() getScale() setPosition() setRotation() setScale() getObject() getShininess() setShininess() getWrappedWindow() setUndoStack() getWrappedWindow() getWrappedWindow() getScene() getMesh() getLight() getSelectedObject() setSelectObject() getCommandStack() getObjectType() getAllObjects() getSceneModel() getWrappedWindow() getSceneModel() getLightType() setSpecularColor () getWrappedWindow() getWrappedWindow() setGeometry() getGeometry() getTexture() setTexture() getPath() getTexture() setPath() getPosition() getNormal() getColor() getUv() setPosition() setNormal() setColor() setUv() getTime() setTime() getPosition() setPosition() setMax()	Success

TU32	Get-Set-Slot	getMainWindow() getObject() getDiffuseColor() getEmission() getEmissionColor() getPosition() getRotation() getScale() getSpecularColor() getShininess() getController() getCommandStack() getLightType() setNumericPrecision() setSceneBackgroundColor() getObjectNames() getNumericPrecision() getSceneBackgroundColor() getSelectedDevice() setDiffuseColor() getDiffuseColor() getSpecularColor() getEmissionColor() setMin() setValue() setColor() getColor() getValue() getDouble() getColor() getBoundingBox() getKeyframes()	Success
TU33	Si verifica che, dato il percorso del file da esportare, questo venga esportato mantenendo le caratteristiche della scena$_G$. Si richiede che tutti i valori di tutti gli oggetti contenuti all'interno della scena$_G$ vengano confrontati con gli oggetti della scena$_G$ stessa, per il formato XML$_G$	exportFile() build() streamSceneObject() streamMesh() streamLight() stream$_G$ () stream$_G$ ()	Success

TU34	Si verifica che, dato il percorso del file da esportare, questo venga esportato mantenendo le caratteristiche della scena$_G$. Si richiede che tutti i valori di tutti gli oggetti contenuti all'interno della scena$_G$ vengano confrontati con gli oggetti della scena$_G$ stessa, per il formato JSON$_G$	exportFile() build() serializeSceneObject() serializeMesh() serializeLight() serialize() serialize() serialize()	Success
TU35	Si verifica che, dato il percorso del file da esportare, questo venga esportato mantenendo le caratteristiche della scena$_G$. Si richiede che tutti i valori di tutti gli oggetti contenuti all'interno della scena$_G$ vengano confrontati con gli oggetti della scena$_G$ stessa, per il formato UBJSON$_G$	exportFile() build() serializeSceneObject() serializeMesh() serializeLight() serialize() serialize() serialize()	Success
TU36	Si verifica che, data una scena$_G$ non vuota esportata su file JSON$_G$, l'importazione dello stesso istanzi una scena$_G$ uguale a quella data inizialmente	importFile () readObject() readLight() readMesh() readObjectProperties()	Success

Tabella 5: Tabella descrizione test unità

B.4 Test di validazione

In questa sezione vengono descritti i test di validazione che servono per accertarsi che il prodotto realizzato sia conforme alle attese.
Per ogni test vengono descritti i vari passi che un utente deve eseguire per testare i requisiti ad esso associati, mentre il tracciamento tra test di validazione e requisiti è riportato nel documento *Analisi dei Requisiti v4.2.0*.

B.4.1 Test TV1

L'utente vuole verificare che si possa decidere di salvare la scena$_G$ corrente nei vari formati disponibili.
All'utente è richiesto di:

- Caricare una scena$_G$ aprendo un file tra quelli consentiti (TV1.1);
- Scegliere di salvare la scena$_G$ caricata (TV1.2)

  All'utente è richiesto di:

  - Scegliere di salvare in un formato con virgola mobile a precisione singola (TV1.2.1);
  - Scegliere di salvare in un formato con virgola mobile a precisione doppia (TV1.2.2);
  - Scegliere di salvare in formato JSON$_G$ (TV1.2.3);
  - Scegliere di salvare in formato UBJSON$_G$ (TV1.2.4);
  - Scegliere di salvare in formato XML$_G$ (TV1.2.5)

    All'utente è richiesto di:

    * Validare il file XML$_G$ salvato (TV1.2.5.1).
  - Scegliere di salvare in formato leggibile (TV1.2.6);
  - Scegliere di salvare in formato minificato$_G$ (TV1.2.7).
- Verificare che un file sia stato salvato (TV1.3).

B.4.2 Test TV2

L'utente vuole verificare che impostando un limite di import e importando una scena$_G$ volutamente eccedente questi limiti, l'anteprima rispetti i limiti impostati.
All'utente è richiesto di:

- Selezionare la dimensione massima di texture da importare (TV2.1);
- Selezionare il massimo numero di luci da importare (TV2.2);
- Verificare che il programma notifichi la perdita di informazioni (TV2.3);
- Selezionare in alternativa i limiti di importazione da una lista predefinita (TV2.4).

B.4.3 Test TV3

L'utente vuole verificare che sia possibile aprire i vari tipi di file permessi.
All'utente è richiesto di:

- Caricare un file con una scena$_G$ conosciuta (TV3.1)

  All'utente è richiesto di:

  - Selezionare un file 3DS$_G$ (TV3.1.1);
  - Selezionare un file JSON$_G$ creato dell'applicazione stessa (TV3.1.2);
  - Selezionare un file nel formato Wavefront obj (TV3.1.3).

B.4.4 Test TV4

L'utente vuole verificare la possibilità di visualizzare le informazioni di sistema.
All'utente è richiesto di:

- Caricare il programma (TV4.1);
- Eseguire il comando di visualizzazione delle informazioni di sistema (TV4.2);
- Verificare che vengano visualizzate le corrette informazioni del sistema come la versione e le informazioni sulla licenza (TV4.3).

B.4.5 Test TV5

L'utente vuole verificare che sia possibile interagire con l'anteprima provando rotazione, panning, zoom e cambiando il colore di sfondo dell'anteprima.
All'utente è richiesto di:

- Caricare una scena$_G$ aprendo un file con formato compatibile (TV5.1);
- Interagire con la scena$_G$ (TV5.2)

  All'utente è richiesto di:

  - Verificare la visualizzazione di uno sfondo (TV5.2.1)

    All'utente è richiesto di:

    * Cambiare il colore dello sfondo (TV5.2.1.1).
  - Modificare la scena$_G$ (TV5.2.2)

    All'utente è richiesto di:

    * Ruotare l'anteprima della scena$_G$ (TV5.2.2.1);
    * Modificare lo zoom dell'anteprima (TV5.2.2.2);
    * Spostare la camera$_G$ (TV5.2.2.3).
- Verificare che il programma visualizzi in modo corretto le modifiche apportate alla scena$_G$ (TV5.3).

B.4.6 Test TV6

L'utente vuole verificare le funzionalità del sistema di aiuto cercando termini o argomenti e provare a visualizzare un risultato della ricerca.
All'utente è richiesto di:

- Caricare il programma (TV6.1);
- Aprire il sistema di aiuto (TV6.2);
- Cercare un termine nel sistema di aiuto (TV6.3)

  All'utente è richiesto di:

  - Verificare il risultato della ricerca (TV6.3.1).
- Navigare il sistema di aiuto per argomenti (TV6.4)

  All'utente è richiesto di:

  - Verificare la bontà dei collegamenti forniti (TV6.4.1).

B.4.7 Test TV7

L'utente vuole verificare la possibilità di annullare le modifiche fatte o ripristinare modifiche annullate.
All'utente è richiesto di:

- Caricare una scena$_G$ da file (TV7.1);
- Effettuare delle modifiche (TV7.2);
- Annullare le modifiche effettuate (TV7.3);
- Ripristinare le modifiche annullate (TV7.4);
- Verificare che i passi effettuati mantengano uno stato consistente della scena$_G$ (TV7.5).

B.4.8 Test TV8

L'utente vuole verificare che sia possibile selezionare un oggetto o una fonte di luce
All'utente è richiesto di:

- Verificare che nella lista di selezione sia possibile scegliere sia un oggetto che una luce (TV8.1).

B.4.9 Test TV9

L'utente vuole verificare che sia possibile modificare un oggetto o una luce
All'utente è richiesto di:

- Verificare la possibilità di poter selezionare un oggetto e modificarlo. (TV9.1)

  All'utente è richiesto di:

  - Caricare una scena$_G$ da file (TV9.1.1);
  - Selezionare un oggetto della scena$_G$ (TV9.1.2);

  - Interagire con l'oggetto selezionato (TV9.1.3)
    All'utente è richiesto di:
    * Ruotare l'oggetto negli assi X, Y, Z (TV9.1.3.1);
    * Traslare l'oggetto negli assi X, Y, Z (TV9.1.3.2);
    * Scalare le dimensioni dell'oggetto (TV9.1.3.3);
    * Modificare le caratteristiche del materiale dell'oggetto (TV9.1.3.4)
      All'utente è richiesto di:
      · Modificare il colore speculare$_G$ dei materiali di un oggetto nella scena$_G$ (TV9.1.3.4.1);
      · Modificare il colore di diffusione$_G$ per i materiali dell'oggetto selezionato (TV9.1.3.4.2).
    * Modificare l'opacità dell'oggetto (TV9.1.3.5)
      All'utente è richiesto di:
      · Modificare l'opacità di parte dell'oggetto (TV9.1.3.5.1).
  - Controllare che il programma visualizzi correttamente le modifiche apportate all'oggetto e quindi alla vista (TV9.1.4).

- Verificare la possibilità di poter selezionare una luce e modificarla. (TV9.2)
  All'utente è richiesto di:
  - Caricare una scena$_G$ da file (TV9.2.1);
  - Selezionare o importare una luce e interagire con essa (TV9.2.2)
    All'utente è richiesto di:
    * Selezionare l'intensità della luce (TV9.2.2.1);
    * Modificare il colore della luce (TV9.2.2.2);
    * Modificare la tipologia della fonte di luce (TV9.2.2.3);
    * Ruotare la luce negli assi X, Y, Z (TV9.2.2.4);
    * Traslare la luce negli assi X, Y, Z (TV9.2.2.5);
    * Importare una luce (TV9.2.2.6)
      All'utente è richiesto di:
      · Scegliere la tipologia di fonte luminosa tra omnidirezionale e spotlight$_G$ (TV9.2.2.6.1).
  - Controllare che il programma visualizzi correttamente le modifiche apportate alla luce e quindi alla vista (TV9.2.3).

C Resoconto delle attività di verifica

C.1 Riassunto delle attività di verifica

C.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 nelle *Norme di Progetto v5.2.0*, nella sezione normativa per i *Verificatori*.
L'*analisi statica* è stata applicata secondo i criteri e le modalità indicate nella sezione 2.8.1. Effettuando *walkthrough* sono stati riscontrati degli errori. Sono state quindi avviate le procedure per la segnalazione e la correzione, descritte nell'apposita sezione delle *Norme di Progetto v5.2.0*.
Noti gli errori, si è provveduto a:

- Correggere le imperfezioni rilevate;
- Segnalare gli errori frequenti, e riportarli nell'apposita sezione relativa all'inspection in appendice alle *Norme di Progetto v5.2.0*. 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 2.9.2.
Il *tracciamento* (requisiti - fonti, use-case - requisiti) è stato effettuato tramite l'applicativo Tracy.
L'avanzamento dei **processi** è stato controllato secondo le metodiche descritte nelle *Norme di Progetto v5.2.0* e verificati applicando la procedura descritta nell'apposita sezione delle *Norme di Progetto v5.2.0*.
Si sono calcolate le *metriche* di processo, descritte nella sezione 2.9.1.
Sono quindi stati riportati e valutati i valori di BV_G e SV_G, ed è stato riportato e valutato il grafico PDCA aggiornato al termine di tutti i processi della fase.

C.1.2 Revisione di Progettazione

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 nelle *Norme di Progetto v5.2.0*, nella sezione normativa per i *Verificatori*.
L'*analisi statica* è stata applicata secondo i criteri e le modalità indicate nella sezione 2.8.1. Effettuando *walkthrough* sono stati riscontrati degli errori. Sono state quindi avviate le procedure per la segnalazione e la correzione, descritte nell'apposita sezione delle *Norme di Progetto v5.2.0*.
Noti gli errori, si è provveduto a:

- Correggere le imperfezioni rilevate;

- Segnalare gli errori frequenti, e riportarli nell'apposita sezione relativa all'inspection in appendice alle *Norme di Progetto v5.2.0*. 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 diagrammi riportati nella *Specifica Tecnica*.
Si sono poi calcolate per i documenti le *metriche* descritte nel punto 2.9.2.
Il *tracciamento* (requisiti - componenti) è stato effettuato tramite l'applicativo Tracy.
L'avanzamento dei **processi** è stato controllato secondo le metodiche descritte nelle *Norme di Progetto v5.2.0* e verificati applicando la procedura descritta nell'apposita sezione nelle *Norme di Progetto v5.2.0*.
Si sono calcolate le *metriche* di processo, descritte nella sezione 2.9.1.
Sono quindi stati riportati e valutati i valori di BV_G e SV_G, ed è stato riportato e valutato il grafico PDCA aggiornato al termine di tutti i processi della fase.

C.1.3 Revisione di Qualifica

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

C.1.3.1 Documenti e processi

I **documenti** sono stati verificati applicando la procedura descritta nelle *Norme di Progetto v5.2.0*, nella sezione normativa per i *Verificatori*.
L'*analisi statica* è stata applicata secondo i criteri e le modalità indicate nella sezione 2.8.1. Effettuando *walkthrough* sono stati riscontrati degli errori. Sono state quindi avviate le procedure per la segnalazione e la correzione, descritte nell'apposita sezione delle *Norme di Progetto v5.2.0*.
Noti gli errori, si è provveduto a:

- Correggere le imperfezioni rilevate;
- Segnalare gli errori frequenti, e riportarli nell'apposita sezione relativa all'inspection in appendice alle *Norme di Progetto v5.2.0*. 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.
Si sono poi calcolate per i documenti le *metriche* descritte nel punto 2.9.2.

L'avanzamento dei **processi** è stato controllato secondo le metodiche descritte nelle *Norme di Progetto v5.2.0* e verificati applicando la procedura descritta nell'apposita sezione nelle *Norme di Progetto v5.2.0*.
Si sono calcolate le *metriche* di processo, descritte nella sezione 2.9.1.
Sono quindi stati riportati e valutati i valori di BV_G e SV_G, ed è stato riportato e valutato il grafico PDCA aggiornato al termine di tutti i processi della fase.

C.1.3.2 Codice

Il **codice** è stato verificato applicando la procedura descritta nelle *Norme di Progetto v5.2.0*, nella sezione *Verifica*.

L'*analisi statica* è stata applicata calcolando le metriche riportate nella sezione 2.9.3. Risultati notevoli di tali misurazioni sono riportati nella sezione C.2.4.3.

Tutti i test pianificati in sede di **Progettazione Architetturale** e **Progettazione di Dettaglio** relativi a componenti implementati sono stati eseguiti, eccezion fatta per i test di validazione che verranno eseguiti nella fase di **Validazione**.
I risultati di tali test sono riportati nell'appendice B.

C.1.4 Revisione di Accettazione

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

C.1.4.1 Documenti e processi

I **documenti** sono stati verificati applicando la procedura descritta nelle *Norme di Progetto v5.2.0*, nella sezione normativa per i *Verificatori*.
L'*analisi statica* è stata applicata secondo i criteri e le modalità indicate nella sezione 2.8.1. Effettuando *walkthrough* sono stati riscontrati degli errori. Sono state quindi avviate le procedure per la segnalazione e la correzione, descritte nell'apposita sezione delle *Norme di Progetto v5.2.0*.
Noti gli errori, si è provveduto a:

- Correggere le imperfezioni rilevate;
- Segnalare gli errori frequenti, e riportarli nell'apposita sezione relativa all'inspection in appendice alle *Norme di Progetto v5.2.0*. 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.
Si sono poi calcolate per i documenti le *metriche* descritte nel punto 2.9.2.

L'avanzamento dei **processi** è stato controllato secondo le metodiche descritte nelle *Norme di Progetto v5.2.0* e verificati applicando la procedura descritta nell'apposita sezione nelle *Norme di Progetto v5.2.0*.
Si sono calcolate le *metriche* di processo, descritte nella sezione 2.9.1.
Sono quindi stati riportati e valutati i valori di BV_G e SV_G, ed è stato riportato e valutato il grafico PDCA aggiornato al termine di tutti i processi della fase.

C.1.4.2 Codice

Il **codice** è stato verificato applicando la procedura descritta nelle *Norme di Progetto v5.2.0*, nella sezione *Verifica*.
L'*analisi statica* è stata applicata calcolando le metriche riportate nella sezione 2.9.3. Risultati notevoli di tali misurazioni sono riportati nella sezione C.2.4.3.

Tutti i test pianificati in sede di **Progettazione Architetturale** e **Progettazione di Dettaglio** relativi a componenti implementati sono stati eseguiti,sono stati eseguiti in oltre i test di validazione.
I risultati di tali test sono riportati nell'appendice B.

C.2 Dettaglio delle verifiche tramite analisi

C.2.1 Analisi

C.2.1.1 Processi

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

Attività	**SV_G**	**BV_G**
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 6: 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 v5.2.0*;
- Il limite inferiore di accettabilità del BV_G è pari a -309€. Il valore ottenuto è quindi accettabile. Dalla tabella 6 si vedono fluttuazioni del valore nelle varie fasi, ma il risultato finale è una valore molto vicino all'ottimo. 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:

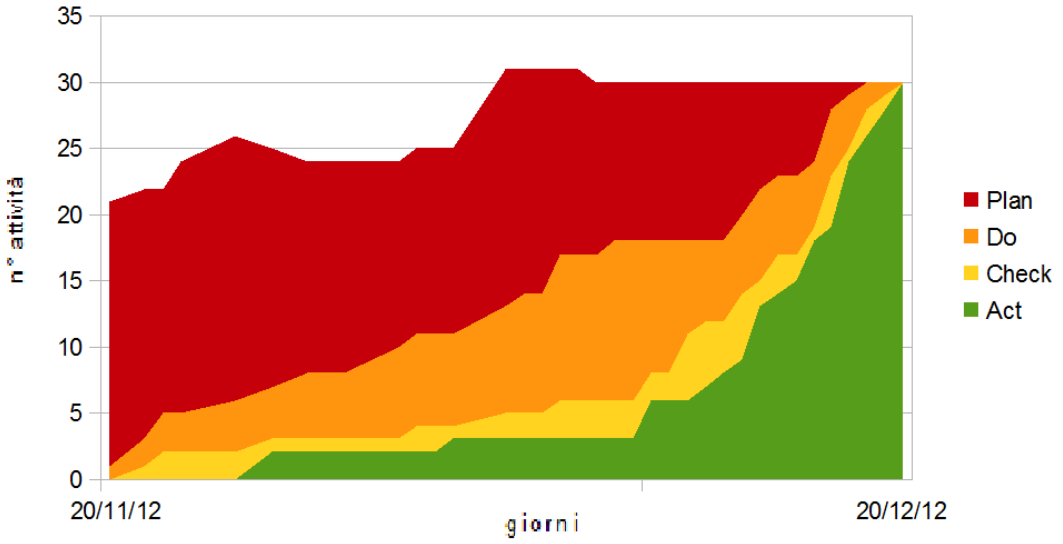

Figura 5: Grafico PDCA, fase di Analisi

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.

C.2.1.2 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 in 2.9.2.1.

Documento	Valore indice	Esito
Piano di Progetto v5.2.0	63.30	superato
Norme di Progetto v5.2.0	65.41	superato
Analisi dei Requisiti v4.2.0	66.78[11]	superato
Piano di Qualifica v5.2.0	56.19	superato
Studio di Fattibilita v3.2.0	60.26	superato
Glossario v5.2.0	52.03	superato

Tabella 7: Esiti verifica documenti, Analisi

[11] 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.

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.

C.2.2 Analisi Dettaglio

C.2.2.1 Processi

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

Attività	SV_G	BV_G
Norme di Progetto	-13€	0€
Analisi Requisiti	0€	140€
Piano di Progetto	45€	-45€
Piano di Qualifica	16€	-40€
Glossario	0€	0€

Tabella 8: Esiti verifica processi, Analisi Dettaglio

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

- $SV_G = 48€$;
- $BV_G = 55€$.

Da valori di tali indici possiamo dedurre che:

- Grazie al lavoro svolto nella fase precedente, la fase di **Analisi**, e grazie agli slack inseriti durante la pianificazione, le attività sono state svolte entro i tempi pianificati, descritti nel *Piano di Progetto v5.2.0*;
- Grazie al lavoro svolto nella precedente fase, la fase di **Analisi**, le attività hanno richiesto una quantità di risorse minore rispetto a quanto pianificato. IL BV_G è quindi risultato positivo.

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

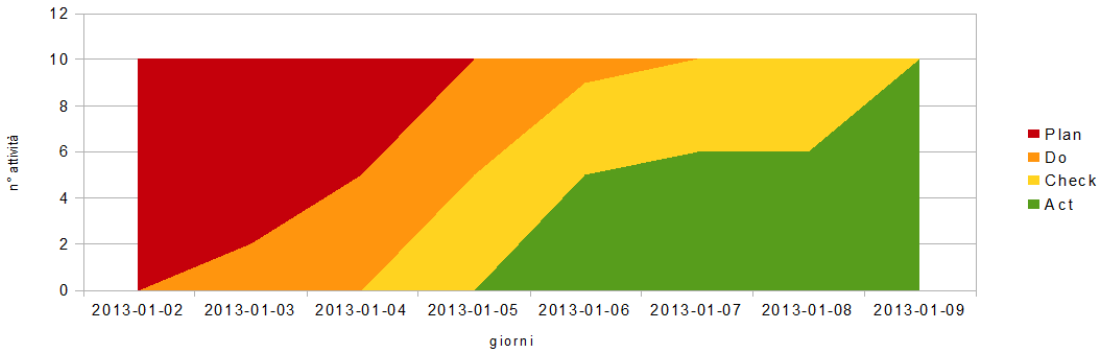

Figura 6: Grafico PDCA, fase di Analisi Dettaglio

Da tale grafico si può asserire che:

- Essendo il grafico rappresentativo di una fase di pochi giorni e poche attività, le fluttuazioni dei valori causano grande variazione grafica;
- In pochi giorni tutte le attività pianificate sono state svolte. Tale dato compare dalla basse permanenza delle attività in Do. Questo è stato possibile in quanto le attività intrinsecamente non richiedevano un carico di lavoro elevato;
- Viene evidenziata una prevalente permanenza delle attività nello stato di Check. Questo è dovuto ad un'accurata verifica effettuata sui processi e sui prodotti di questi;
- Il rallentamento a metà della fase è stato causato dal collidere degli impegni universitari con l'avanzamento del progetto.

C.2.2.2 Documenti

Vengono qui riportati i valori dell'indice Gulpease per ogni documento durante la fase di **Analisi Dettaglio**. Un documento è considerato valido soltanto se rispetta le metriche descritte su 2.9.2.1.

Documento	Valore indice	Esito
Piano di Progetto v5.2.0	62.40	superato
Norme di Progetto v5.2.0	64.31	superato
Analisi dei Requisiti v4.2.0	66.10 [12]	superato
Piano di Qualifica v5.2.0	57.15	superato
Studio di Fattibilita v3.2.0	58.43	superato
Glossario v5.2.0	51.18	superato

Tabella 9: Esiti verifica documenti, Analisi Dettaglio

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. Avendo effettuato un basso numero di modifiche ai documenti ed essendosi concentrati nella correzione e nella verifica, il valori ottenuti sono sostanzialmente invariati rispetto fase precedente.

[12] 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.

C.2.3 Progettazione Architetturale

C.2.3.1 Processi

Vengono qui riportati i valori degli indici SV_G e BV_G, descritti nella sezione 2.9.1, per le attività della fase di **Progettazione Architetturale**.

Attività	SV_G	BV_G
Norme di Progetto	10€	0€
Analisi Requisiti	-10€	0€
Piano di Progetto	-25€	40€
Piano di Qualifica	0€	-25€
Specifica Tecnica	210€	-281€
Glossario	0€	0€

Tabella 10: Esiti verifica processi, Progettazione Architetturale

Complessivamente, la fase di **Progettazione Architetturale** ha:

- SV_G = 185€;
- BV_G = -256€.

Da valori di tali indici possiamo dedurre che:

- La data di fine della **Progettazione Architetturale** si è dimostrata essere 3 giorni prima di quella pianificata nel *Piano di Progetto v5.2.0*.
  Aumentando il numero di ore di lavoro giornaliere ed utilizzando tutti gli slack inseriti durante la pianificazione il gruppo ha compresso i tempi con conseguente SV_G positivo;
- Il limite inferiore di accettabilità del BV_G è pari a -394€. Il valore ottenuto è quindi accettabile. Come già precisato, il gruppo per portare a termine gli obbiettivi entro le date imposte ha aumentato il numero di ore di lavoro giornaliere. Questo ha quindi causato un aumento del BV_G in quanto le ore complessive di lavoro sono state vicine a quelle preventivate, ma il periodo di lavoro è stato compresso.

Il grafico PDCA della fase di **Progettazione Architetturale** è:

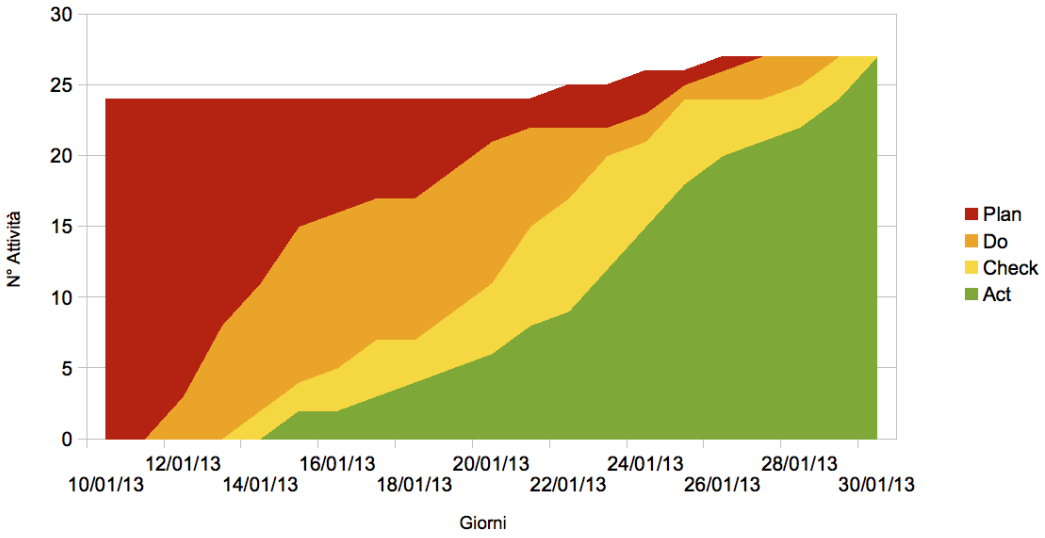

Figura 7: Grafico PDCA, fase di Progettazione Architetturale

Da tale grafico si può asserire che:

- La ripida crescita iniziale dello stato di DO è dovuta alla correzione degli errori, segnalati nella **Revisione dei Requisiti**, ai quali corrisponde un incremento dei documenti;
- Vi sono state 3 iterazioni dovute all'inesperienza del gruppo nell'eseguire i processi;
- Una delle iterazioni è dovuta all'incontro collettivo effettuato con il docente, dopo il quale è stato valutato insufficiente il *Piano di Qualifica* ed è stata iterata la stesura di parte di tale documento;
- Le altre due iterazioni sono dovute a iterazioni nella Progettazione.

C.2.3.2 Documenti

Vengono qui riportati i valori dell'indice Gulpease per ogni documento redatto durante la fase di **Progettazione Architetturale**. Un documento è considerato valido soltanto se rispetta le metriche descritte su 2.9.2.1.

Documento	Valore indice	Esito
Piano di Progetto v5.2.0	62.32	superato
Norme di Progetto v5.2.0	59.99	superato
Analisi dei Requisiti v4.2.0	67.29[13]	superato
Piano di Qualifica v5.2.0	59.31	superato
Studio di Fattibilita v3.2.0	60.58	superato
Specifica Tecnica v4.2.0	63.11	superato
Glossario v5.2.0	51.56	superato

Tabella 11: Esiti verifica documenti, Progettazione Architetturale

[13] 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.

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.
I documenti *Norme di Progetto* e *Piano di Qualifica* denotano un discostamento dei valori notevole se paragonati alla fase precedente. Questo è dovuto a modifiche pesanti nella struttura e nei contenuti dei documenti.
I diagrammi UML$_G$ inseriti nella *Specifica Tecnica* hanno superato il test di validazione di WhiteStarUML indicando la presenza di 0 elementi con errori formali.

C.2.3.3 Progettazione

Viene qui riportata una tabella riassuntiva che riporta il calcolo dei parametri di accoppiamento afferente ed efferente per i componenti individuati nella progettazione architetturale.

Componente	Afferente	Efferente
DDDMob::Model	2	2
DDDMob::Model::Commands	0	2
DDDMob::Model::CConverter	3	1
DDDMob::Model::CConverter::CSettingsModel	0	0
DDDMob::Model::CConverter::CExportModel	0	0
DDDMob::Model::CConverter::CLoaderModel	3	1
DDDMob::Controller	0	1
DDDMob::View	0	1
DDDMob::View::CDockWidgets	0	1
DDDMob::View::CMainWindow	1	1
DDDMob::View::CSettingsWindow	1	0
DDDMob::View::CErrorWindow	1	0
DDDMob::View::CWidgetElement	1	0
DDDMob::C3DObject	2	0

Tabella 12: Tabella accoppiamento componenti

Come si può vedere dalla tabella, l'accoppiamento efferente è generalmente molto basso e quindi positivo, ad eccezione del package$_G$ Controller, per il quale però questo è accettato a causa della natura intrinseca del componente. L'accoppiamento afferente mostra invece la stabilità richiesta dalle classi del Model, e questo è aiutato dalla strategia di integrazione scelta, che integra e sottopone a test per prima cosa il modello. Questo dovrebbe portare ad un componente stabile più velocemente, prevenendo il rischio di regressione dovuto ad un alto accoppiamento.

C.2.4 Progettazione di Dettaglio e Codifica

C.2.4.1 Processi

Vengono qui riportati i valori degli indici SV_G e BV_G, descritti nella sezione 2.9.1, per le attività della fase di **Progettazione di Dettaglio e Codifica**.

Attività	**SV_G**	**BV_G**
Norme di Progetto	30€	1,65€
Analisi Requisiti	0€	0€
Piano di Progetto	0€	-85,09€
Piano di Qualifica	0€	0€
Specifica Tecnica	€	0€
Definizione di Prodotto	0€	-7,37€
Codifica	0€	-271,63€
Manuale Utente	278€	-30,55€
Resoconto delle Attività di Verifica	0€	0€
Glossario	30€	0€

Tabella 13: Esiti verifica processi, Progettazione di Dettaglio e Codifica

Complessivamente, la fase di **Progettazione di Dettaglio e Codifica** ha:

- SV_G = 338€;
- BV_G = -393€.

Da valori di tali indici possiamo dedurre che:

- La data di fine della **Progettazione di Dettaglio e Codifica** si è dimostrata essere anticipata rispetto a quella pianificata nel *Piano di Progetto v5.2.0*. Aumentando il numero di ore di lavoro giornaliere ed utilizzando tutti gli slack inseriti durante la pianificazione il gruppo ha compresso i tempi con conseguente SV_G positivo;
- Il limite inferiore di accettabilità del BV_G è pari a -640€. Il valore ottenuto è quindi accettabile. Come già precisato, il gruppo per portare a termine gli obbiettivi entro le date imposte ha aumentato il numero di ore di lavoro giornaliere. Questo ha quindi causato un aumento del BV_G in quanto le ore complessive di lavoro sono state vicine a quelle preventivate, ma il periodo di lavoro è stato compresso.

Il grafico PDCA della fase di **Progettazione di Dettaglio e Codifica** è:

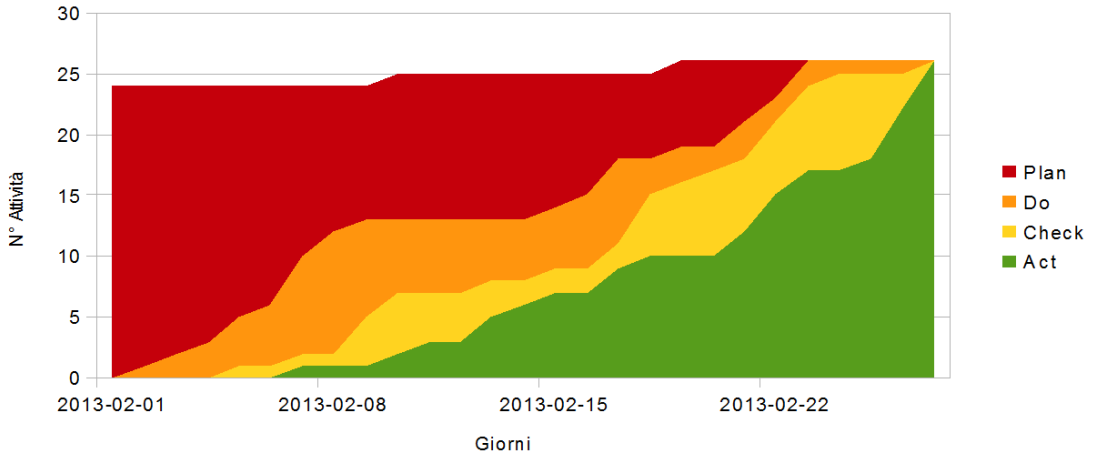

Figura 8: Grafico PDCA, fase di Progettazione di Dettaglio e Codifica

Da tale grafico si può asserire che:

- La crescita iniziale dello stato di DO è dovuta alla correzione degli errori, segnalati nella **Revisione di Progettazione**, ai quali corrisponde un incremento dei documenti;
- Vi sono state 2 iterazioni dovute all'inesperienza del gruppo nell'eseguire i processi;
- Una delle iterazioni è dovuta alla decisione del gruppo di sostituire le librerie Qt3D con le librerie OpenGL$_G$ e Assimp;
- La seconda iterazione è conseguita a piccole modifiche dell'interfaccia grafica che hanno portato all'aggiornamento del manuale utente.

C.2.4.2 Documenti

Vengono qui riportati i valori dell'indice Gulpease per ogni documento redatto durante la fase di **Progettazione di Dettaglio e Codifica**. Un documento è considerato valido soltanto se rispetta le metriche descritte su 2.9.2.1.

Documento	Valore indice	Esito
Piano di Progetto v5.2.0	62.07	superato
Norme di Progetto v5.2.0	59.81	superato
Analisi dei Requisiti v4.2.0	67.20[14]	superato
Piano di Qualifica v5.2.0	59.89	superato
Specifica Tecnica v4.2.0	63.10	superato
Definizione di Prodotto v5.2.0	65.30	superato
Glossario v5.2.0	51.41	superato

Tabella 14: Esiti verifica documenti, Progettazione di Dettaglio e Codifica

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.

C.2.4.3 Risultati delle misurazioni sul codice

Di seguito vengono riportati i risultati dei test di analisi statica e dinamica effettuati sul codice.
Per ogni test si riportano il valore medio riscontrato e i massimi, riportando la giustificazione di eventuali risultati fuori dal range di accettazione.
Ove ritenuto significativo si riportano le tabelle complete con i risultati dei test. Si omettono i risultati dei test riguardanti singoli metodi perché troppo onerosi.

Complessità ciclomatica

- **Media**: 4;
- **Massimo**: 14.

Tutti i moduli rientrano nel range di accettazione.

Livello di annidamento

- **Media**: 1.75;
- **Massimo**: 5.

Tutti i moduli rientrano nel range di accettazione.

[14] 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.

Attributi per classe

- **Media**: 3;
- **Massimo**: 9.

Tutti i moduli rientrano nel range di accettazione.

Parametri per metodo

- **Media**: 2;
- **Massimo**: 5.

Tutti i moduli rientrano nel range di accettazione.

Linee di commento per linee di codice

- **Media**: 77/100.

Tutti i moduli rientrano nel range di accettazione.

Instabilità

- **Media**: 0,38.

Componente	Instabilità
DDDMob::Model	0.5
DDDMob::Model::Commands	1
DDDMob::Model::CConverter	0.25
DDDMob::Model::CConverter::CSettingsModel	0
DDDMob::Model::CConverter::CExportModel	0
DDDMob::Model::CConverter::CLoaderModel	0.25
DDDMob::Controller	1
DDDMob::View	1
DDDMob::View::CDockWidgets	1
DDDMob::View::CMainWindow	0.5
DDDMob::View::CSettingsWindow	0
DDDMob::View::CErrorWindow	0
DDDMob::View::CWidgetElement	0
DDDMob::C3DObject	0

Tabella 15: Tabella instabilità componenti

Copertura del codice

- **Media**: 75%.

Il valore di copertura del codice è stato calcolato considerando le linee di codice effettivamente testate dai test di unità. Da tale percentuale risulta quindi esclusa la copertura dei metodi *get* e *set*, che, come si può notare dalla tabella sottostante, risultano essere molto numerosi ma non meritevoli di test di unità in quanto triviali.

Inoltre, molti metodi non testati sono virtuali puri e quindi privi di implementazione, o considerati triviali per lunghezza e operazioni eseguite. Si considera quindi sufficiente al momento, vista anche la soddisfacente copertura di linee di codice, l'esecuzione dei test di unità date le tempistiche previste dalla pianificazione.

	Numero metodi	**Percentuale sul totale**
Verificati con test di unità	77	23
Metodi triviali	85	26
Verificati per costruzione	46	14
Non testati o virtuali puri	123	37
Totale	331	100

Tabella 16: Tabella relativa al numero di metodi coperti da test di unità

C.2.5 Validazione

C.2.5.1 Processi

Vengono qui riportati i valori degli indici SV_G e BV_G, descritti nella sezione 2.9.1, per le attività della fase di **Progettazione di Dettaglio e Codifica**.

Attività	SV_G	BV_G
Norme di Progetto	-16€	-5€
Analisi Requisiti	0€	0€
Piano di Progetto	20€	2€
Piano di Qualifica	-9€	4 €
Specifica Tecnica	14€	5€
Definizione di Prodotto	13,30€	4€
Manuale Utente	-30€	8€
Glossario	8,25€	-5€

Tabella 17: Esiti verifica processi, Validazione

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

- SV_G = 5,55€;
- BV_G = 22€.

Da valori di tali indici possiamo dedurre che:

- Le attività sono state svolte in leggero anticipo rispetto a quanto pianificato quindi lo SV_G è positivo ma con un valore non elevato;
- Il limite inferiore di accettabilità del BV_G è pari a -278,90€. Il valore ottenuto è quindi accettabile. Anche il è risultato positivo in.

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

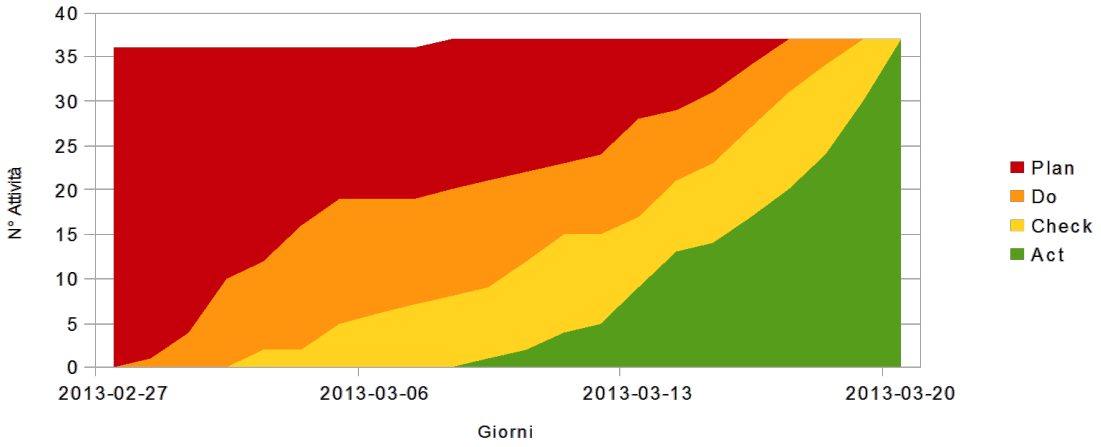

Figura 9: Grafico PDCA, fase di Validazione

Da tale grafico si può asserire che:

- La crescita iniziale dello stato di DO è dovuta alla correzione degli errori, segnalati nella **Revisione di Qualifica**, ai quali corrisponde un incremento dei documenti;

- A metà 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;
- Vi è stata una iterazione. Tale iterazione è conseguita alla ricezione di un modello 3D da parte del proponente. Tale modello includeva una tipologia di texture precedentemente non gestite. A ciò è conseguito un periodo di codifica non previsto.

C.2.5.2 Documenti

Vengono qui riportati i valori dell'indice Gulpease per ogni documento redatto durante la fase di **Validazione**. Un documento è considerato valido soltanto se rispetta le metriche descritte su 2.9.2.1.

Documento	Valore indice	Esito
Piano di Progetto v5.2.0	62.54	superato
Norme di Progetto v5.2.0	59.82	superato
Analisi dei Requisiti v4.2.0	67.20[15]	superato
Piano di Qualifica v5.2.0	60.57	superato
Specifica Tecnica v4.2.0	63.10	superato
Definizione di Prodotto v5.2.0	65.35	superato
Glossario v5.2.0	53.12	superato
Manuale Utente v5.2.0	59.26	superato

Tabella 18: Esiti verifica documenti, Validazione

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.

C.2.5.3 Risultati delle misurazioni sul codice

Di seguito vengono riportati i risultati dei test di analisi statica e dinamica effettuati sul codice.
Per ogni test si riportano il valore medio riscontrato e i massimi, riportando la giustificazione di eventuali risultati fuori dal range di accettazione.
Ove ritenuto significativo si riportano le tabelle complete con i risultati dei test. Si omettono i risultati dei test riguardanti singoli metodi perché troppo onerosi.

Complessità ciclomatica

- **Media**: 4;
- **Massimo**: 14.

Tutti i moduli rientrano nel range di accettazione.

Livello di annidamento

- **Media**: 1.75;
- **Massimo**: 5.

Tutti i moduli rientrano nel range di accettazione.

[15] 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.

Attributi per classe

- **Media**: 3;
- **Massimo**: 9.

Tutti i moduli rientrano nel range di accettazione.

Parametri per metodo

- **Media**: 2;
- **Massimo**: 5.

Tutti i moduli rientrano nel range di accettazione.

Linee di codice per linee di commento

- **Media**: 83/100.

Tutti i moduli rientrano nel range di accettazione.

Instabilità

- **Media**: 0,38.

Componente	Instabilità
DDDMob::Model	0.5
DDDMob::Model::Commands	1
DDDMob::Model::CConverter	0.25
DDDMob::Model::CConverter::CSettingsModel	0
DDDMob::Model::CConverter::CExportModel	0
DDDMob::Model::CConverter::CLoaderModel	0.25
DDDMob::Controller	1
DDDMob::View	1
DDDMob::View::CDockWidgets	1
DDDMob::View::CMainWindow	0.5
DDDMob::View::CSettingsWindow	0
DDDMob::View::CErrorWindow	0
DDDMob::View::CWidgetElement	0
DDDMob::C3DObject	0

Tabella 19: Tabella instabilità componenti

I valori si discostano di poco dai risultati precedenti la Revisione di Qualifica, grazie ad un'attività di programmazione volta soprattutto al miglioramento e alla correzione di errori. Notevole il miglioramento del rapporto tra linee di codice e linee di commento, ottenuto grazie ad un commento più estensivo di metodi, prototipi e parametri ove necessario.

Copertura del codice

- **Media**: 85%.

Il valore di copertura del codice è stato calcolato considerando le linee di codice effettivamente testate dai test di unità. La copertura è migliorata rispetto a quanto ottenuto precedentemente alla Revisione di Qualifica, grazie alla definizione di nuovi test di unità ed al miglioramento di alcuni test per assicurare la copertura di tutti i cammini di alcuni metodi complessi.
Le uniche parti di codice non triviale non testato restano così le parti di codice OpenGL$_G$, per le quali sono stati studiate le possibili tecniche per l'esecuzione di test di unità. In particolare si sono studiati:

- Authoring dell'output e differenze rispetto ad output successivi: ha lo svantaggio di non essere automatizzabile, inoltre modifiche anche solo legate all'aliasing delle mesh di output corretti farebbero fallire il test;
- Metodi di test automatico: progetti prodotti dalla comunità di sviluppatori ed ancora immaturi;
- Log con fake driver$_G$: consiste nel registrare tutte le chiamate ad OpenGL$_G$ mediante un driver$_G$ apposito. È scarsamente automatizzabile.

Si è scelto di adottare un metodo simile al fake driver$_G$, ma garantendo la correttezza a monte del codice legato al rendering$_G$. Per fare ciò, è stato specificato in modo puntuale nel documento *Definizione di Prodotto* l'ordine di chiamata alle funzioni OpenGL$_G$, in modo che l'implementazione del *Programmatore* produca un ordine delle chiamate uguale all'ordine atteso. In questo modo si prova per costruzione la correttezza dei metodi *renderInternal()* e non è stata ritenuta necessaria la verifica dell'output.

	Numero metodi	**Percentuale sul totale**
Verificati con test di unità	92	28
Metodi triviali	81	24
Verificati per costruzione	46	14
Non testati o virtuali puri	107	32
Totale	326	100

Tabella 20: Tabella relativa al numero di metodi coperti da test di unità

Bug Durante il collaudo del prodotto sono emersi un totale di 9 bug.
Tali bug sono stati repentinamente corretti, come dimostrato dal grafico::

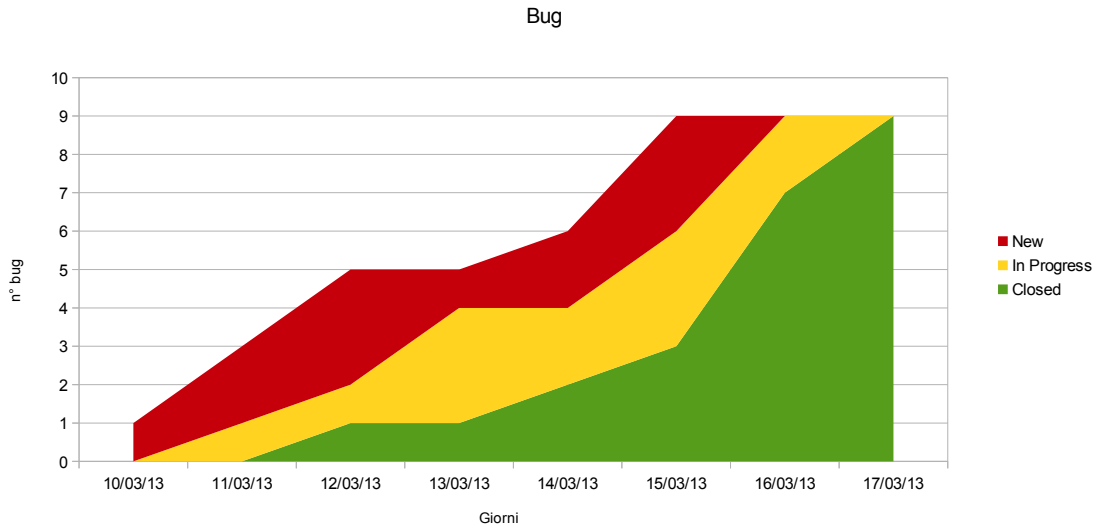

Figura 10: Grafico Bug

Un così basso numero di bug è merito di un ferreo utilizzo delle metriche stabilite e da controlli automatici che hanno portato ad un codice prevalentemente corretto e di bassa complessità. Grazie alla bassa complessità il codice è risultato essere rapidamente e facilmente correggibile.

C.3 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 revisione 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.

Si elencano di seguito le modifiche applicate in seguito alle revisioni.

C.3.1 Revisione dei Requisiti

- *Norme di Progetto*: il documento è stato integrato con i contenuti richiesti riguardanti l'attività di progettazione, sono state definite norme per la gestione dei cambiamenti e norme che garantiscano l'assenza di conflitto di interesse nella rotazione dei ruoli.
  In tale documento sono state inoltre inserite tutte le sezioni prima impropriamente presenti nei documenti *Piano di Progetto* e *Piano di Qualifica*.
  Da tale revisione è conseguita una profonda modifica del documento;

- *Analisi dei Requisiti*: sono state apportate le modifiche suggerite. Alcuni casi d'uso sono stati maggiormente specificati e dove richiesto sono stati inseriti i diagrammi;

- *Piano di Progetto*: l'utilizzo della terminologia è stato corretto in base a quanto segnalato. Il nome della fase **Verifica e Validazione** è stato corretto in **Validazione** così da non creare ambiguità. L'analisi dei rischi è stata riassunta in forma tabellare. Le attività pianificate per la redazione di *Specifica Tecnica* sono state maggiormente descritte e rinominate con l'intento di renderle più chiare;

- *Piano di Qualifica*: il documento ha subito una profonda revisione basata sulle correzioni indicate e su quanto specificato in seguito ad un ricevimento con il committente.
  È stata rivista la struttura del documento dando maggior rilievo alle procedure di strategia generale; è stata creata un'appendice dove sono state spostate le sezioni che descrivono gli standard di qualità utilizzati come riferimento e dove vengono riportati i resoconti delle diverse revisioni con il committente. Sono state trasferite nelle *Norme di Progetto* le sezioni attinenti agli strumenti e alle procedure di verifica;

- *Glossario*: sono state apportate delle modifiche alla struttura del documento secondo le correzioni. Sono stati eliminati gli indici precedentemente anteposti al contenuto.

C.3.2 Revisione di Progettazione

- *Norme di Progetto*: per tale documento è stato suggerito l'uso di diagrammi di attività per descrivere le procedure a supporto dei processi adottate dal gruppo. Tali procedure sono dettagliatamente descritte nella sezione *Ambiente di lavoro* alla quale la sezione di procedure fa riferimento e nella quale si fa ampio uso di

tali diagrammi. Si è ritenuto quindi di non dover aggiungere ulteriori diagrammi in quanto quelli necessari sono già definiti nella sopracitata sezione;

- *Analisi dei Requisiti*: il documento ha raggiunto la sua versione definitiva e non sono stati riscontrate correzioni da effettuare;
- *Specifica Tecnica*: il documento ha subito un'accurata revisione seguendo i consigli forniti dal committente e correggendo gli errori individuati. Come richiesto, i capitoli relativi ai componenti e classi sono stati collassati in unico capitolo.
  Sono stati corretti i diagrammi delle classi che contenevano errori e sono state ampliate e dettagliate le relative descrizioni testuali.
  Tutti i diagrammi di attività sono stati resi conformi allo standard UML_G prescelto e sono stati aggiornati suddividendo quei diagrammi che risultavano essere di dimensioni troppo grandi, e quindi di difficile comprensione, in sotto-diagrammi più piccoli e di più facile manutenzione;
- *Piano di Progetto*: il documento ha subito le modifiche richieste in sede di **Revisione di Progettazione**;
- *Piano di Qualifica*: il documento ha subito una modifica strutturale in quanto la sezione relativa ai test è stata spostata in appendice data la sua natura incrementale;
- *Glossario*: al documento non sono state apportate modifiche sostanziali se non l'aggiunta di nuovi termini.

C.3.3 Revisione di Qualifica

- *Norme di Progetto*: al documento sono state apportate le modifiche richieste;
- *Analisi dei Requisiti*: il documento ha raggiunto la sua versione definitiva e non sono stati riscontrate correzioni da effettuare;
- *Specifica Tecnica*: il documento ha raggiunto la sua versione definitiva e non sono stati riscontrate correzioni da effettuare;
- *Definizione di Prodotto*: il documento ha subito le modifiche richieste in sede di **Revisione di Qualifica**;
- *Piano di Progetto*: il documento ha subito le modifiche richieste, relative alla riorganizzazione di parte della struttura;
- *Piano di Qualifica*: al documento non sono state apportate modifiche sostanziali se non l'aggiunta dei resoconti delle nuove attività di verifica;
- *Manuali Utenti*:il documento ha subito le modifiche richieste in sede di **Revisione di Qualifica**;
- *Glossario*: al documento non sono state apportate modifiche sostanziali se non l'aggiunta di nuovi termini.