



Piano di Qualifica

20 febbraio 2007

Sommario

Il documento identifica e dettaglia la strategia di verifica e validazione proposta per il collaudo del prodotto, incluse le eventuali prove con le azioni corrispondenti in ingresso e risultati attesi. Documento redatto secondo le Norme di Progetto v1.9.

Informazioni documento

Produzione	WheelSoft - wheelsoft@gmail.com
Redazione	Alberto De Bortoli - adeborto@studenti.math.unipd.it Matteo Borgato - mborgato@studenti.math.unipd.it
Approvazione	Stefano Ceschi Berrini - sceschib@studenti.math.unipd.it Alessio Rambaldi - arambald@studenti.math.unipd.it Silvio Daminato - sdaminat@studenti.math.unipd.it Matteo Borgato - mborgato@studenti.math.unipd.it Giulio Favotto - gfavotto@studenti.math.unipd.it
File	Piano_di_Qualifica_v1.9.pdf
Versione	1.9
Stato	Formale
Uso	Esterno
Distribuzione	Wheelsoft prof. Alessandro Sperduti prof. Renato Conte prof. Tullio Vardanega



Diario delle modifiche

- | | |
|-----|--|
| 1.9 | 20/02/07 - Inserimento nella sezione <i>Attività di verifica sul codice</i> delle sezioni <i>Test di unità</i> , <i>Test d'integrazione</i> e <i>Metriche del prodotto</i> (G.Favotto) |
| 1.8 | 18/02/07 - Riorganizzazione della struttura del documento con la ridenominazione della sezione <i>Gestione amministrativa della revisione</i> in <i>Gestione delle attività di verifica</i> e relativo aggiornamento, inserimento delle sezioni <i>Attività di verifica sulla documentazione</i> come ampliamento della sezione <i>Resoconto delle attività di verifica sui documenti</i> e della sezione <i>Attività di verifica sul codice</i> in cui vengono definiti i criteri di verifica da attuare suddivisi in tipologie e il resoconto delle verifiche già effettuate (G.Favotto) |
| 1.7 | 01/02/07 - Correzione imprecisioni (S. Daminato) |
| 1.6 | 28/01/07 - Si è cercato di spiegare la strategia di verifica e validazione che Wheelsoft intende utilizzare per il progetto <i>SIAGAS</i> . Creata la tabella di criticità del software e spiegata inoltre la pianificazione del collaudo. (S. Ceschi Berrini) |
| 1.5 | 23/01/07 - Modificate le sezioni di Tecniche di Analisi, Metodi di verifica e Resoconto delle attività di verifica. (S. Ceschi Berrini) |
| 1.4 | 20/01/07 - Aggiornato documento in base al nuovo organigramma (S. Ceschi Berrini) |
| 1.3 | 16/01/07 - Modifica della struttura del documento, aggiunte nuove sezioni, correzione errori del documento consegnato per la RR. (S. Ceschi Berrini) |
| 1.2 | 11/12/06 - correzioni varie e approvazione |
| 1.1 | 07/12/06 - prima stesura del documento |



Indice

1	Introduzione	5
1.1	Scopo del documento	5
1.2	Scopo del prodotto	5
1.3	Glossario	5
1.4	Riferimenti	5
1.4.1	Normativi	5
1.4.2	Informativi	6
2	Visione generale della strategia di verifica	7
2.1	Organizzazione, pianificazione strategica e temporale, responsabilità	7
2.2	Dettaglio delle revisioni	8
2.2.1	Revisione dei Requisiti	8
2.2.2	Revisione del Progetto Preliminare	8
2.2.3	Revisione del Progetto Definitivo	8
2.2.4	Revisione di Qualifica	8
2.2.5	Revisione di Accettazione	8
2.3	Risorse necessarie e disponibili	9
2.3.1	Risorse umane	9
2.3.2	Risorse tecnologiche	9
2.4	Tecniche di Analisi	9
2.5	Metodi di verifica	10
3	Gestione delle attività di verifica	12
3.1	Comunicazione e risoluzione di anomalie	12
3.2	Procedure di controllo di qualità di processo	12
3.2.1	Livelli di criticità	13
3.2.2	Metriche di valutazione del processo	13
4	Attività di verifica sulla documentazione	14
4.1	Pianificazione	14
4.2	Resoconto	14
4.2.1	Verifiche tramite analisi	14
4.2.2	Verifiche tramite prove (test)	15
4.2.3	Esito delle revisioni	15
5	Attività di verifica sul codice	16
5.1	Pianificazione	16
5.1.1	Analisi statica	16
5.1.2	Analisi dinamica	16
5.2	Metriche del prodotto	20
5.2.1	Numero di livelli	20
5.2.2	Numero di parametri	20
5.2.3	Complessità ciclomatica	20
5.2.4	Numero delle anomalie	21
5.2.5	Fan-in e Fan-out	21



6	Attività di verifica finali	22
6.1	Specifica della campagna di validazione (collaudo incluso)	22
6.2	Dettaglio dell'esito della campagna di validazione	22



1 Introduzione

1.1 Scopo del documento

Lo scopo di questo documento è pianificare il processo relativo alla garanzia di qualità. Il processo relativo alla garanzia di qualità (Quality Assurance Process) deve assicurare che i prodotti e i processi relativi al ciclo di vita adottato siano conformi ai requisiti specificati e aderiscano ai piani prestabiliti. Il Piano di Qualifica dovrà inoltre identificare e dettagliare appropriatamente la strategia di verifica e validazione proposta dal fornitore per il collaudo del prodotto, incluse le eventuali prove, con le rispettive precondizioni e postcondizioni. Deve essere verificata la congruenza e la soddisfaccibilità dei requisiti presenti nel documento di “Analisi dei Requisiti v2.0”.

1.2 Scopo del prodotto

Per lo scopo del prodotto si faccia riferimento al documento “Analisi dei Requisiti v2.0”.

1.3 Glossario

Il glossario è presente in un documento PDF chiamato “Glossario.v1.8.pdf”, allegato alla documentazione per la Revisione dei Requisiti, che definisce e spiega i termini tecnici utilizzati nei documenti ufficiali al fine di eliminare ogni ambiguità relativa al linguaggio.

1.4 Riferimenti

1.4.1 Normativi

- Capitolato “Sportello Informatico per Attivazione e Gestione di Attività di Stage”, emesso dal committente prof. Alessandro Sperduti in data 07/11/06 e reperibile all’indirizzo:

www.math.unipd.it/~tullio/IS-1/2006/Progetti/SIAGAS.html

- Analisi dei Requisiti
- Norme di Progetto
- IEEE/EIA 12207.0-1996 - Software Life Cycle Processes
- IEEE Std 610.12-1990 - Standard Glossary of Software Engineering Terminology
- IEEE Std 730TM-2002 (revision of IEEE Std 730-1998) - Standard for Software Quality Assurance Plans
- IEEE Std 1012-1998 - Standard for Software Verification and Validation



1.4.2 Informativi

- Frederick P. Brooks: No Silver Bullet: Essence and Accidents of Software Engineering
- Guide to the SWEBOK (www.swebok.org)



2 Visione generale della strategia di verifica

2.1 Organizzazione, pianificazione strategica e temporale, responsabilità

Si assume che un processo ben pianificato ed amministrato porti al raggiungimento di un alto livello di qualità del prodotto. Le attività di verifica, necessarie per giungere al collaudo del sistema, saranno definite dal responsabile, il quale le suddividerà in moduli che verranno assegnati ai singoli verificatori; la pianificazione deve includere la totalità dei moduli che costituiscono il progetto e rispettare i tempi definiti dal responsabile. L'organizzazione delle strategie di verifica sarà principalmente basata sull'Analisi dei Requisiti. In questo modo, si otterrà la tracciabilità finalizzata a collegare i requisiti richiesti con le componenti e i relativi metodi che saranno individuati in dettaglio. Come detto in precedenza, responsabili dell'attività di verifica saranno i Verificatori ed il Responsabile di progetto.

Il processo di verifica deve essere istanziato ogni qual volta il prodotto di un processo subisca sostanziali cambiamenti. Tali cambiamenti devono essere necessariamente notificati nel registro delle modifiche relativo a quel prodotto. La segnalazione e la correzione di eventuali problemi o anomalie riscontrati durante processo di verifica dovranno essere trattati dal processo di risoluzione dei problemi.

La verifica inoltre si occuperà di assicurarsi che la definizione del requisito sia fatta in maniera non ambigua e la relativa trasposizione del componente ricopra tutte le caratteristiche specificate nel requisito. Sarà importante quindi il tracciamento, nel quale si associa ad ogni requisito uno o più componenti.

L'attività di verifica dovrà essere attuata in maniera costante ed al passo con l'avanzamento del prodotto, in modo da minimizzare i rischi di fallimento e/o discrepanze più o meno gravi.

Si è deciso di adottare un modello di ciclo di vita incrementale. La tipologia del sistema da sviluppare e la struttura dei processi di revisione si prestano bene ad un approccio che prevede necessità di adattamento ai cambiamenti: evoluzione dei problemi e tecnologie utilizzabili.

I processi di revisione adottati saranno di due tipi:

- revisione formale:
 - revisione dei requisiti (**RR**)
 - revisione di accettazione (**RA**)
- revisione informale:
 - revisione del progetto preliminare (**RPP**)
 - revisione del progetto definitivo (**RPD**)
 - revisione di qualifica (**RQ**)



2.2 Dettaglio delle revisioni

2.2.1 Revisione dei Requisiti

- **Prodotti in ingresso:** Capitolato d'Appalto, Analisi dei Requisiti, Piano di Progetto
- **Funzione:** accordarsi con il committente (cliente) per una descrizione il più possibile esaustiva del prodotto software atteso
- **Stato di uscita:** prodotto descritto

2.2.2 Revisione del Progetto Preliminare

- **Prodotti in ingresso:** Specifica Tecnica, aggiornamento del Piano di Qualifica
- **Funzione:** accertamento di realizzabilità, attivare la fase di realizzazione del prodotto
- **Stato di uscita:** prodotto specificato

2.2.3 Revisione del Progetto Definitivo

- **Prodotti in ingresso:** Definizione di Prodotto, aggiornamento del Piano di Qualifica
- **Funzione:** informare il committente riguardo le caratteristiche definitive del prodotto, attivare la fase di qualifica
- **Stato di uscita:** prodotto definito

2.2.4 Revisione di Qualifica

- **Prodotti in ingresso:** aggiornamento del Piano di Qualifica, inclusione della specifica delle prove di accettazione, versione preliminare del Manuale d'Uso
- **Funzione:** approvazione della campagna di verifica; attivazione della fase di accettazione
- **Stato di uscita:** prodotto qualificato

2.2.5 Revisione di Accettazione

- **Prodotti in ingresso:** versione definitiva del Piano di Qualifica, inclusione dell'esito delle prove di accettazione, versione definitiva del Manuale d'Uso
- **Funzione:** collaudo del sistema per Accettazione del prodotto da parte del committente
- **Stato di uscita:** prodotto accettato



2.3 Risorse necessarie e disponibili

La verifica dei moduli che compongono il prodotto *SIAGAS* impegna risorse umane e tecnologiche. L'amministratore di progetto eseguirà un'attività di supervisione e coordinamento di tali risorse durante la verifica. Questa procedura è necessaria per accertarsi che le risorse siano sufficienti ed efficienti per consentire una verifica di buon livello.

2.3.1 Risorse umane

Attualmente il gruppo prevede la seguente suddivisione dei ruoli:

- **1 Responsabile**
- **1 Amministratore**
- **3 Progettisti**
- **2 Verificatore**

Si faccia riferimento al piano di progetto per vedere la suddivisione dei ruoli nel tempo.

I verificatori avranno un ruolo chiave per il processo di qualità del prodotto in quanto dovranno cercare di scoprire errori ed anomalie sia nei documenti che nel codice. Per far questo dovranno controllare sintassi e semantica dei documenti e dovranno fare dei test per poter verificare il codice.

2.3.2 Risorse tecnologiche

Il gruppo dispone e si avvarrà delle seguenti risorse tecnologiche:

- **Un Server Web Apache:** per testare il progetto complessivo;
- **L^AT_EX:** per la stesura/correzione dei documenti;
- **Strumenti di validazione forniti dal W3C:** per aderire a standard affidabili;
- **Diversi tipi di Browser:** per estendere il dominio di utilizzo;
- **Editor testuali per le pagine php ed html:** per la stesura/correzione della *parte web*;
- **MySQL:** per testare il corretto funzionamento del *SIAGASDB*.

2.4 Tecniche di Analisi

- **Analisi Statica:** viene svolta in maniera contestuale alla stesura del codice a partire dai requisiti di sistema. È necessaria al fine di individuare eventuali situazioni in cui il codice sia ridondante, la complessità non sia appropriata e vi siano problematiche riguardanti la progettazione "a tavolino". Bisogna esplicitare e analizzare le dipendenze tra ingressi e uscite che caratterizzano le unità di codice testate, e provare la correttezza del codice sorgente rispetto ai requisiti prefissati.



- **Analisi Dinamica:** verifica dinamicamente il comportamento di ogni singola unità di codice, sia indipendentemente (ogni singolo componente) sia nell'insieme (sul sistema). Deve cominciare prima della fine della fase di codifica e le sue esigenze devono essere considerate anche in fase di progettazione.

Il progettista ne definirà le strategie di applicazione:

- il Test di Unità:
 - * *funzionale*: ha lo scopo di verificare ogni unità affinché soddisfi i requisiti previsti mediante l'impiego di un insieme di dati in ingresso che siano in grado di generare risultati attesi
 - * *strutturale*: verifica la logica interna del codice, cioè sono effettuati controlli affinché ciascuna unità vada ad attivare ogni cammino di esecuzione possibile
- Test di Integrazione: è applicato alle componenti e si serve della logica di integrazione funzionale per selezionare le funzioni da integrare, per ordinare le componenti per numero di dipendenze e per eseguire l'integrazione a partire dalle componenti con precedenza. Lo scopo finale è quello di individuare difetti di progettazione o carenze in fase di verifica. L'integrazione tra componenti avviene in modo incrementale e assemblando per primi i moduli che forniscono flusso di controllo e flusso dei dati, garantendo sempre un'integrazione che sia reversibile.
- Test di Sistema: verifica del sistema per accertare che siano rispettati i requisiti iniziali.
- Test di Accettazione: collaudo del sistema effettuato in presenza del committente.

2.5 Metodi di verifica

Le revisioni formali ed informali saranno lo strumento principale per la verifica. Queste serviranno per esaminare e convalidare documentazione, requisiti, codice, diagrammi e tecniche di verifica.

Le tecniche di verifica si divideranno in Analisi statica, Analisi dinamica e Tracciamento (atto a dimostrare completezza ed economicità dell'implementazione).

L'Analisi statica, come descritto dagli standard di certificazione, viene applicata combinando i seguenti metodi:

- Flusso di controllo
- Flusso dei dati
- Flusso dell'informazione
- Esecuzione simbolica
- Verifica formale del codice
- Verifica di limite



- Uso dello stack
- Comportamento temporale
- Interferenza
- Codice oggetto

Inoltre la verifica dovrà includere attività come:

- Inspection : per rilevare la presenza di difetti tramite la lettura mirata del codice. Quest'attività è suddivisa in più fasi (pianificazione, definizione della lista di controllo, lettura del codice, correzione dei difetti). In ogni fase dovrà essere documentata l'attività svolta;
- Walkthrough : per rilevare la presenza di difetti tramite la lettura critica del codice. Quest'attività è suddivisa in più fasi (pianificazione, lettura del codice, discussione, correzione dei difetti). In ogni fase dovrà essere documentata l'attività svolta;

Wheelsoft si impegnerà a fare un'Analisi di questo tipo utilizzando i metodi necessari e più efficienti rispetto alle esigenze:

- Per la documentazione verrà fatta analisi sintattica e semantica del testo, oltre che controllo di possibili ambiguità nel linguaggio ed errori logici in schemi/modelli/diagrammi.
- Per quanto riguarda il codice, verranno combinati metodi di Analisi statica (come descritto sopra).

Inoltre, per fare test sui componenti singoli e sul sistema generale, verrà svolta Analisi dinamica utilizzando anche driver e stub.

Tutte le fasi dell'attività di verifica dovranno essere documentate (attività svolte, risultati e anomalie).



3 Gestione delle attività di verifica

3.1 Comunicazione e risoluzione di anomalie

Il Verificatore, per ogni modulo verificato, dovrà redigere un documento ad uso interno nel quale saranno specificati:

- modulo in analisi;
- stato del modulo;
- dati in ingresso;
- dati in uscita attesi;
- dati in uscita rilevati;
- esito del test ed eventuali anomalie riscontrate;

Nelle norme di progetto viene definita un modello di struttura per la relazione di un test case e l'inserimento di eventuali anomalie nel software di supporto.

Per ogni anomalia riscontrata si dovranno accertare le:

- possibili cause
- possibili soluzioni

Il Responsabile provvederà all'approvazione del documento e lo inoltrerà all'Amministratore, indicando il riferimento alla segnalazione di anomalia attivata. Verificatori e Programmatori designati raccoglieranno la segnalazione per provvedere alla sua gestione.

3.2 Procedure di controllo di qualità di processo

L'Amministratore ha a suo carico il controllo della gestione dell'attività di verifica, dell'emissione e distribuzione dei documenti durante i test e della gestione e risoluzione delle anomalie.

Ogni processo dovrà essere descritto in maniera dettagliata e non ambigua, in modo che sia semplice da controllare.

Dovranno essere indicati anche i livelli di criticità.



3.2.1 Livelli di criticità

Wheelsoft cercherà di attenersi allo standard IEEE 730-2002. La seguente tabella specifica le priorità rispetto la criticità del software:

Grado	Componente	Danni al prodotto
3	componente fondamentale rispetto alle funzionalità del sistema	prodotto inutilizzabile
2	componente importante rispetto alle funzionalità del sistema	importanti disagi causati al prodotto
1	componente influisce su funzionalità rilevanti del sistema	prodotto utilizzabile ed anomalia osservabile
0	componente influisce su funzionalità trascurabili	nessun disagio sostanziale causato al prodotto

3.2.2 Metriche di valutazione del processo

In termini di metriche relative ai processi di verifica Wheelsoft si propone di attenersi allo standard IEEE 1012:1998.

Dovranno essere analizzati vari fattori per valutare lo sviluppo e la qualità del prodotto. Questi saranno:

- Complessità del processo;
- Risorse richieste dal processo;
- Tempo richiesto dal processo;
- Eventuale gestione degli errori.

Nel caso in cui la qualità dell'output del processo sia accettabile, si cercherà di ridurre i costi senza diminuire la qualità.



4 Attività di verifica sulla documentazione

4.1 Pianificazione

Non potendo eseguire attività di dinamica sulla documentazione, la verifica sarà caratterizzata principalmente da attività di analisi statica:

- **Adesione di un documento alle norme di progetto**

- **Tracciamento**

Il tracciamento in questa fase ricopre un ruolo determinante per quanto riguarda la correttezza di un documento, in quanto permette di verificare che un'esigenza manifestata dal proponente venga formalmente descritta. Saranno pertanto necessarie le seguenti forme di tracciamento:

- **Tracciamento requisiti-fonti**

Il tracciamento requisiti-fonti viene descritto completamente nel documento di “Analisi Dei Requisiti v2.0” utilizzando dei codici di fonte e di requisito univoci in modo da poter associare ad ogni requisito una fonte documentata;

- **Tracciamento requisiti-componenti**

Il tracciamento requisiti-componenti viene descritto completamente nel documento “Specifica Tecnica v1.7” utilizzando dei codici di requisito e di componente univoci per poter associare ad ogni requisito uno o più componenti;

- **Tracciamento requisiti-metodi**

Il tracciamento requisiti-metodi viene descritto completamente nel documento “Definizione di Prodotto v1.6” utilizzando dei codici metodo univoci per poter associare ad ogni requisito uno o più metodi;

4.2 Resoconto

4.2.1 Verifiche tramite analisi

Verifiche attuate finora:

Revisione	Tecnica
Revisione Requisiti	In questa fase è stata attuata analisi statica secondo i criteri e le modalità previste, redigendo i documenti di verifica richiesti ed avviando le procedure per la correzione degli errori riscontrati
Revisione Progetto preliminare	Anche in questa fase è stata attuata analisi statica secondo i criteri e le modalità previste, redigendo i documenti di verifica richiesti ed avviando le procedure per la correzione degli errori riscontrati. Particolare attenzione è stata posta nella verifica dei diagrammi UML inseriti in cui sono stati riscontrate e corrette varie imperfezioni, dovute principalmente all'inesperienza nella redazione di tali modelli.



4.2.2 Verifiche tramite prove (test)

Al momento non sono state fatte verifiche tramite prove.

4.2.3 Esito delle revisioni

Revisioni fatte finora:

Tipo di revisione	Esito
Revisione dei Requisiti	<ul style="list-style-type: none">• Documento “Analisi dei requisiti v2.0”: A fini di tracciabilità occorre associare un identificatore alfanumerico a ciascun singolo requisito. Al momento invece i requisiti hanno identificatori numerici a intersezione non vuota. Buona qualità tecnica. Buona l’impostazione anche se l’analisi non è ancora sufficientemente approfondita. Alcuni termini sono generici (requisiti desiderabili: Interfaccia immediata? Efficienza del sistema ? ...). Manca tracciamento requisiti verso le fonti. Manca anche una specifica (almeno preliminare) della strategia di verifica dei requisiti;• Documento “Piano di Progetto”: Errori e orrori terminologici;• Documento “Piano di Qualifica”: nella versione attuale manca di profondità e di prospettiva strategica.
Revisione Progetto Preliminare	<ul style="list-style-type: none">• Documento “Specifica Tecnica v1.6”: Fig.3: manca il nome della relazione “stage-docente”, la classe “Log” è isolata;• Documento “Piano di Progetto”: Ottimo. Sarebbe utile una tabella riassuntiva dei costi preventivati a finire rivisti alla luce del consuntivo attuale;• Documento “Piano di Qualifica”: Qualche miglioramento rispetto alla versione RR. Le sezioni 2.4-5, pur impegnative, sono lacunose perchè non relazionano le intenzioni strategiche all’identificazione dell’infrastruttura di supporto necessaria per realizzarle sul piano tecnologico e metodologico.



5 Attività di verifica sul codice

5.1 Pianificazione

La verifica sul codice si avvarrà delle forme di verifica sopra descritte, nelle forme di analisi statica e dinamica.

5.1.1 Analisi statica

Per realizzare l'analisi statica, come previsto dalle norme di progetto, si farà affidamento sulla funzionalità di segnalazione errori di PHP5. E' inoltre in fase di valutazione la possibilità di far affidamento su alcuni tools di recente sviluppo per ricevere ulteriore supporto all'analisi statica su linguaggi di scripting. Verranno comunque effettuate verifiche per rilevare la complessità in termini di annidamento e di branching.

5.1.2 Analisi dinamica

L'analisi dinamica prevede lo svolgimento di test su di un insieme finito di test case. Essendo il test una tipologia di verifica di tipo *negativo* con costi piuttosto elevati, è necessario trovare un compromesso fra il numero minimo di casi di prova necessari per dimostrare il corretto funzionamento di un componente e la quantità di risorse disponibili. Per questo motivo sarà quindi necessario effettuare le rilevazioni metriche descritte nella sezione 5.2 in modo da determinare con più precisione possibile la quantità e la modalità di test da effettuare.

- **Test di unità**

Sono stati valutati vari strumenti di supporto per realizzare tali test; l'orientamento attuale prevede vengano utilizzati alcuni script java-php facenti parte dei pacchetti JWebUnit (estensione di JUnit) e SimpleTest in modo da verificare sia i componenti del livello base, il data layer, che i componenti logici PHP presenti nel logic layer ed infine i componenti HTML-CSS presenti nel presentation layer.

Statement coverage e branch coverage saranno effettuati sulla base delle metriche rilevate.

Segue la descrizione dei test per componente.

- **Data layer**

- * **Descrizione componente:** Il Data layer contiene tutti i dati relativi al sistema SIAGAS.
- * **Strategia:** Oltre al controllo dell'adesione del documento alle norme di progetto relative alla creazione del Data layer, dovranno essere verificate le condizioni di sicurezza del database, la presenza dei vincoli di unicità delle chiavi, nonché il corretto funzionamento di alcune query che testino i casi limite. Queste query sono in via di definizione.

- **pear.MDB2**

- * **Descrizione componente:** *pear.MDB2* è il package che si occupa di eseguire query sul Data layer.



- * **Strategia:** Non verranno effettuati test strutturali in quanto MDB2 è un package esterno fornito come stabile.
- **attori.Utente**
 - * **Descrizione componente:** Questa componente è una classe astratta. Contiene solo metodi, comuni alle sottoclassi, senza parametri, che ritornano sempre lo stesso output (se lo stato del Data layer rimane invariato).
 - * **Strategia:** Si dovrà utilizzare una qualsiasi sottoclasse derivata non astratta per testare i metodi tramite un *driver* che richiama i metodi definiti in *Utente*.
- **attori.UtenteAutenticato**
 - * **Descrizione componente:** Questa componente è una classe astratta. Contiene solo metodi, comuni alle sottoclassi, senza parametri, che ritornano sempre lo stesso output (se lo stato del Data layer rimane invariato).
 - * **Tecnica:** Si dovrà utilizzare una qualsiasi sottoclasse derivata non astratta per testare i metodi tramite un *driver* che richiama i metodi definiti in *Utente*.
- **attori.UtenteNonAutenticato**
 - * **Descrizione componente:** Questa componente raccoglie i metodi per un utente non autenticato. I metodi di questa componente contengono costrutti di controllo per svolgere funzioni di accesso e di registrazione attraverso query sul Data layer.
 - * **Strategia:** L'obiettivo è quello di ottenere un *branch coverage* dei costrutti di controllo realizzando un driver tramite il quale poter fornire tutti i valori significativi, accertando che i metodi si comportino come dovrebbero. Test funzionali con *valori legali* ed *illegali* verranno pertanto effettuati per ogni metodo con appositi script, mentre i test strutturali saranno effettuati già in fase di scrittura e se necessario saranno approfonditi al rilevamento di valori anomali delle metriche previste.
- **attori.Studente, attori.Amministratore, attori.Docente, attori.Proponente, attori.Super_utente**
 - * **Descrizione componenti:** Queste componenti si comportano similmente in quanto rappresentano gli utenti del sistema, pertanto le tecniche di verifica saranno comuni.
 - * **Strategia:** Test funzionali con *valori legali* ed *illegali* verranno pertanto effettuati per ogni metodo con appositi script, mentre i test strutturali saranno effettuati già in fase di scrittura e se necessario saranno approfonditi al rilevamento di valori anomali delle metriche previste.
- **attori.Sistema**
 - * **Descrizione componente:** *Sistema* è la classe principale dell'applicazione; è realizzata come un singleton con i metodi che permettono ad esempio la generazione di documenti, accesso al Data layer (attraverso MDB2), l'autenticazione etc... Si presterà



attenzione alla sicurezza nei metodi di questa classe, salvando i documenti creati in una cartella sicura e non permettendo l'esecuzione di query pericolose o l'inserimento di dati pericolosi (come codice javascript).

- * **Strategia:** È necessario assicurare che il singleton torni effettivamente un'istanza di sé stesso ad un'ipotetico chiamante. Verrà dunque realizzato uno *stub* tramite il quale ottenere l'istanza. In particolar modo si dovrà accertare che le richieste successive alla prima non creino una nuova istanza dell'oggetto in questione. Test funzionali con *valori legali* ed *illegali* verranno effettuati per ogni metodo con appositi script, mentre i test strutturali saranno effettuati già in fase di scrittura e se necessario saranno approfonditi al rilevamento di valori anomali delle metriche previste. La generazione dei documenti verrà testata manualmente. Gli automatismi verranno testati nella fase successiva.

– Presentation layer

- * **Descrizione componente:** Il presentation layer fornisce l'interfaccia grafica all'utente del sistema eseguendo le funzioni rese disponibili dal livello sottostante. E' pertanto composto da svariate componenti la cui conoscenza singola non risulta importante in quanto esse risultano omogenee anche in termini di esigenze di verifica.
- * **Strategia:** Verrà verificato che ogni pagina creata rispetti le norme di progetto esistenti, quindi le validazioni W3C in termini di CSS, XHTML e WAI-AAA dovranno essere superate positivamente. Attraverso i tool selezionati per il controllo del codice html sarà verificato non siano presenti link scorretti o errori sintattici. I test funzionali e strutturali sul codice php saranno effettuati con appositi script utilizzando *valori legali* e *valori illegali*. Ogni cammino dovrà risultare coperto.

• Test d'integrazione

La scelta architetturale relativa all'utilizzo di una struttura a tre strati promuove la scalabilità del sistema ed offre la possibilità di una facile sostituzione dei livelli interoperanti. Il passo successivo all'aver effettuato i test di unità è di, partendo dal basso (cioè dal data layer), sostituire ciascuno strato con *driver* creati ad hoc. Tale tipo di approccio è di carattere incrementale e s'avvicina alla strategia di tipo bottom-up.

Vengono descritte ora le strategie di integrazione delle componenti:

– Integrazione Data layer, pear.MDB2

- * **Descrizione contestuale:** MDB2 è un package che permette di connettersi ad un database astraendo dal tipo di database per maggiore portabilità. MDB2 usa metodi per connettersi e per eseguire query.



- * **Strategia:** Saranno necessari test funzionali per verificare i casi estremi da eseguire con appositi script; test strutturali, essendo PEAR un package esterno, non saranno eseguiti.
- * **Aspettazione:** Si dovranno riscontrare gli esiti attesi
- **Integrazione attori.***Sistema*, pear.MDB2
 - * **Descrizione contestuale:** *Sistema* è la classe che più interagisce con pear.MDB2, esegue sia query di interrogazione che di modifica. Le altre classi che eseguono query usano i metodi di *Sistema*, i quali ritornano l'informazione così come ritornata da pear.MDB2 a *Sistema*. Non ci sarà quindi necessità di eseguire test anche sull'intergrazione, per quanto riguarda l'esecuzione di query, tra le altre classi del package attori e la classe *Sistema* stessa.
 - * **Strategia:** Su una tabella di prova verranno effettuate un numero di query-test sufficiente a garantire il buon funzionamento dei metodi (almeno una query per ogni categoria, dove per categoria s'intende *select*, *insert*, *update* e *delete*).
 - * **Aspettazione:** La query richiesta viene effettuata senza errori con l'effetto desiderato.
- **Integrazione Presentation layer**, attori.*UtenteNonAutenticato*
 - * **Descrizione contestuale:** Per permettere agli utenti di autenticarsi nel sistema SIAGAS il presentation layer necessita di un'istanza di *Amministratore*, *Studiante*, *Docente* o *PropONENTE*. Per ottenere quest'istanza il presentation layer invoca un metodo su *UtenteNonAutenticato*. Il metodo avrà al suo interno dei costrutti di controllo che, in base ai parametri passati, torneranno al chiamante la corretta istanza dell'utente.
 - * **Strategia:** Dovranno essere effettuati almeno tanti test quanti sono i possibili tipi di istanze che il metodo ritorna.
 - * **Aspettazione:** Il metodo che ritorna l'istanza agisce correttamente.
- **Integrazione attori.***Sistema*, attori
 - * **Componenti integrate:** attori.*UtenteNonAutenticato*, attori.*Studiante*, attori.*Amministratore*, attori.*Proponente*, attori.*Docente*, attori.*Sistema*
 - * **Descrizione contestuale:** Per poter accedere al database e per poter eseguire altre azioni importanti, le classi del package attori usano il singleton *Sistema*. Devono quindi poter ottenere un'istanza di *Sistema* usando un metodo statico.
 - * **Strategia:** Si effettuerà una banale prova di funzionamento del metodo statico per accertarsi che la classe *Sistema* ritorni un'istanza di sé stessa.
 - * **Aspettazione:** Il metodo statico che restituisce l'istanza funziona correttamente.
- **Integrazione Presentation layer**, attori
 - * **Descrizione contestuale:** Il presentation layer, per poter eseguire interrogazioni e per azioni quali generazione di password,



generazione di notifiche ed allerte, deve usare metodi delle classi del package attori.

- * **Tecnica:** Si effettueranno test di funzionamento di questi metodi, con copertura completa dei cammini.
- * **Aspettazione:** I metodi si comportano secondo la definizione fatta.

5.2 Metriche del prodotto

Al fine di monitorare la qualità del codice e quindi del prodotto finale verranno prese in considerazione le seguenti metriche quantitative.

5.2.1 Numero di livelli

Questa metrica rileva il massimo livello d'annidamento di ciascun metodo. Il livello d'annidamento di un metodo rappresenta la misura in cui il suo codice si ramifica, ed è determinato facendo riferimento al numero di istruzioni che impongono una divisione del flusso. Il NdL si calcola determinando il massimo tra tutti i livelli presenti in un metodo. Si assume che un valore di NdL alto comporti alcune problematiche relative alla comprensione e alla complessità del codice.

5.2.2 Numero di parametri

Questa metrica si basa sul conteggio del numero di parametri di un metodo. Metodi con un valore alto di NdP potrebbero dover essere modificati al fine di ottenere un maggiore livello d'astrazione ed un aumento del grado di disaccoppiamento. L'alto numero di parametri di un metodo, dunque, potrebbe significare una mancanza di coesione nelle funzionalità fornite dal metodo stesso (con conseguenti problemi relativi la manutenibilità).

L'obiettivo è limitare il numero di parametri massimo a 6.

5.2.3 Complessità ciclomatica

La complessità ciclomatica misura il numero di cammini linearmente indipendenti che attraversano un metodo. Questa misurazione fornisce un singolo valore intero che può essere comparato con la complessità di altri programmi. Tale valore è, tra le altre cose, utilizzato per acquisire due tipi di informazioni. Innanzi tutto esprime il numero minimo di test da effettuare sul software per garantire un'adeguata *branch coverage*. In secondo luogo è utile, a partire dalla progettazione architetturale, per controllare l'affidabilità, la testabilità e l'amministrabilità del software. Ciascun metodo può essere rappresentato con un grafo ad albero. Il valore di complessità ciclomatica relativo a quel determinato metodo è espresso dalla formula che segue.

$$CC = E - N + P$$

- CC = Complessità Ciclomantica
- E = Numero di archi



- N = Numero di nodi
- P = Numero di Componenti Connesse

Si assume che un alto valore di complessità ciclomatica non denota necessariamente un grosso rischio relativo alla parte di codice presa in esame. Esso è, piuttosto, il sintomo che il codice in considerazione può non essere ben congegnato dal punto di vista logico (e che, sicuramente, sarà oggetto di un non trascurabile lavoro di prova).

Uno dei problemi che concerne l'analisi ciclomatica riguarda la stima del suo valore in relazione all'istruzione di tipo **switch**. La presenza di uno **switch** all'interno del codice, infatti, aumenta significativamente il valore di complessità ciclomatica. Verranno effettuate delle considerazioni per valutare le possibili trattazioni del problema.

L'obiettivo è portare il codice generato ad avere una complessità ciclomatica inferiore a 10.

5.2.4 Numero delle anomalie

Questa metrica si basa sul conteggio delle anomalie riscontrate. Tale metrica sarà disponibile in maniera aggiornata in ogni momento grazie al supporto fornito dall'applicazione di bugtracking indicata nelle norme. Importante sarà rilevare il momento in cui il numero delle anomalie riscontrate non risulterà più crescere ma sarà costante, in modo da non impiegare ulteriori risorse inutilmente.

5.2.5 Fan-in e Fan-out

Questa metrica dimostra l'utilità o la dipendenza di un componente rispetto ad altri. Verranno rilevati questi valori per i vari componenti del sistema. L'adozione di un software di supporto è in fase di valutazione, ma vista la relativa compattezza del sistema tale metrica potrebbe essere rilevata anche manualmente.



6 Attività di verifica finali

Questa sezione sarà definita in maniera più dettagliata nella prossima fase.

6.1 Specifica della campagna di validazione (collaudo incluso)

Ai fini della validazione si controllerà che il prodotto finale soddisfi tutti i requisiti richiesti dal proponente prof. A.Sperduti e raccolti nell'ultima versione dell'analisi dei requisiti.

Ai fini del collaudo, il prodotto verrà testato in due modalità:

- α Test (pre collaudo): Attività che verrà fatta internamente al fornitore. Wheelsoft si impegna a svolgere test utilizzando il Server Web Apache per provare il *SIAGASDB* ed i moduli php. Inoltre verranno utilizzate varie piattaforme software (diversi tipi di browser e diversi sistemi operativi) in modo da poter verificare la portabilità del sistema. Per l'accessibilità si valideranno le pagine seguendo gli standard W3C;
- β Test (collaudo) : Attività controllata e guidata dal committente che verterà ai test svolti nell' α Test.

6.2 Dettaglio dell'esito della campagna di validazione

Al momento non è stata fatta campagna di validazione.