Simulatore di Gestione della Memoria di un Elaboratore SiGeM

http://stylosoft.altervista.org
stylosoft@gmail.com

Piano di qualifica	08 Marzo 2008
Documento Esterno - Formale - v3.1	Piano_di_qualifica_3.1.pdf

Redazione:

Luca Rubin

Revisione:

Daniele Bonaldo

Approvazione:

Luca Rubin

Lista di distribuzione:

Prof. Vardanega Tullio
Prof. Palazzi Claudio
Stylosoft

Registro delle modifiche:

Versione	Data	Descrizione delle modifiche
3.1	17/03/2008	Aggiunto punto 5.2
3.0	08/03/2008	Correzione generale documento, e avanzamento versione
2.3	03/03/2008	Aggiunti paragrafi 5.2 , 5.3 , 7
2.2	14/02/2008	Ampliati punti 6.4.1 e 6.4.2
2.1	05/02/2008	Aggiunto paragrafo 6 “Modalità di verifica”
2.0	24/01/2008	Correzione grammaticali e di impaginazione
1.3	20/01/2008	Inserimento stili e indice automatizzato
1.2	14/01/2008	Inserimento dettagli sul repository SVN
1.1	12/01/2008	Ridefinita l'intera struttura del documento con maggiore attenzione sulla strategia da seguire

1.0	06/12/2007	Correzione grammaticale, aggiustamento indice
0.2	04/12/2007	Sistemazione indice; inserimento dei punti 2.3 e 2.4

Sommario

Questo documento vuole tracciare le linee guida per l'organizzazione delle attività di verifica e di accertamento della qualità del prodotto “SiGeM”.

Indice

1 Introduzione

1.1 Scopo del documento

Nel presente documento verranno presentate le strategie con cui Stylosoft intende garantire la qualità del prodotto. Nei punti a seguire, verranno illustrate le modalità con cui lo sviluppo del prodotto sarà verificato. Al seguente documento potranno essere apportate modifiche in corso d'opera per adattare al meglio l'attività di verifica del nostro prodotto.

1.2 Scopo del prodotto

Il prodotto in oggetto ha la funzione di software didattico per lo studio dei meccanismi di gestione della memoria di un elaboratore multiprogrammato.

1.3 Glossario

Nel documento “Glossario_3.0.pdf” verranno riportati i termini più specifici con il relativo significato. Si ricorda, inoltre, che ogni termine presente nel “Glossario” sarà scritto in corsivo nei vari documenti.

1.4 Riferimenti

1.4.1 Normativi

- ISO/IEC 12207
- ISO/IEC 9126:2001
- SWEBOK

1.4.2 Informativi

- [NI] Norme interne (Norme_interne_ 3.0.pdf)
- [AR] Analisi dei requisiti (Analisi_dei_requisiti_2.0.pdf)
- [ST] Specifica tecnica (Specifica_tecnica_3.0.pdf)
- [PP] Piano di progetto (Piano_di_progetto_3.0.pdf)
- [RT] Risultati dei test (RisultatiTest_1.0.pdf)
- [RAS] Risultati dell'analisi statica (Risultati_Analisi_1.0.pdf)

2 Visione generale della strategia di verifica

2.1 Organizzazione, pianificazione strategica e temporale, responsabilità

Il modello di ciclo di vita scelto per il progetto è quello sequenziale, dato che tale modello impone una certa disciplina e comporta verifiche rigorose sul completamento di ogni fase. Per maggiori dettagli sul ciclo di vita scelto vedere la sezione 2 del piano di progetto ([PP]).

Le attività di verifica verranno suddivise ed assegnate ai membri del gruppo dal responsabile di progetto, il quale si preoccuperà anche che tutti i moduli seguano le regole definite in questo documento rispettando le scadenze prefissate.

Le attività di verifica e validazione copriranno l'intero periodo di sviluppo del progetto.

Le fasi principali in cui verranno applicati i due precedenti processi (verifica e validazione) saranno:

- Fase Iniziale: dopo aver improntato i primi documenti, tra cui anche quello dell'analisi dei requisiti, verranno inizializzate le prime attività di verifica in cui i verificatori preposti, controlleranno la loro corretta stesura dal punto di vista grammaticale e dal punto di vista dei contenuti. Nel caso in cui venisse riscontrato un errore o una mancanza, sarà compito del verificatore, provvedere alla segnalazione dell'errore utilizzando gli strumenti messi a disposizione del Team. Questi strumenti, e le regole di utilizzo, sono presenti nelle norme interne ([NI]). Nel momento in cui la verifica termina senza il riscontro di errori, il documento potrà giungere al responsabile il quale dopo aver apposto il proprio nome (per approvazione) potrà consegnarlo al cliente.
- Fase di Progettazione: è possibile vedere questa fase come due sotto-fasi successive:
  - Progettazione architetturale
  - Progettazione di dettaglio

  Nella prima i verificatori devono far si che tutti i requisiti pattuiti con il cliente siano stati presi in considerazione e siano stati soddisfatti nella loro interezza. Nella seconda sotto-fase, i verificatori dovranno controllare che siano applicate le decisioni prese nelle fasi precedenti (fase iniziale e progettazione architetturale).
- Fase di codifica e di test sul codice: in questa fase i verificatori possono procedere con l'esecuzione di una vera e propria attività di analisi e test del codice. Le attività di verifica in questa fase avranno inizio non appena ci saranno unità e/o moduli di codice con un grado di indipendenza tale da poter eseguire alcune o tutte le attività sotto citate. Per adempiere a tale compito, i verificatori dovranno praticare attività di:
  - Analisi statica: attraverso l'uso di metodi di lettura (deck check) come *inspection* e

walkthrough; il metodo "inspection" verrà usato per rilevare gli errori più prevedibili di codifica con ricerche mirate (attraverso la definizione di una lista di controllo). Per il metodo "walkthrough" la verifica verrà eseguita dai verificatori i quali saranno affiancati da alcuni programmatori preposti a tale compito.

- Analisi dinamica: consiste nell'esecuzione del codice nella sua interezza o in parti autonome. Tali analisi potrà quindi fornire risultati sia sull'intero sistema sia sui singoli moduli. Alcune tecniche per adempiere a tale scopo sono WhiteBox (verifica che il programma generi i risultati voluti seguendo ogni suo possibile cammino logico), BlackBox (verifica che ogni singola funzione produca i risultati attesi provando ogni potenziale situazione in cui si possa trovare).

– Fase di collaudo del prodotto: in questa fase il prodotto è pronto per essere eseguito e testato nella sua interezza. Il collaudo avverrà essenzialmente in due fasi:
  - α-test: consiste in un collaudo interno all'azienda.
  - β-test: è successivo all'α-test e viene rilasciato anche a soggetti esterni all'azienda. Anche i soggetti esterni avranno l'opportunità di segnalare eventuali anomalie riscontrate, attraverso l'invio di e-mail all'indirizzo stylosoft@gmail.com.

Sulla base di vincoli economici e temporali verrano adottati due processi di revisione:

– Un processo di revisione esterna di tipo sanzionatorio condotta dal cliente applicato a:
  - RR (Revisione dei Requisiti);
  - RA (Revisione di Accettazione);
– Un processo di revisione esterna (di progresso) non sanzionatorio applicato a:
  - RPP (Revisione del Progetto Preliminare);
  - RPD (Revisione del Progetto Definitivo);
  - RQ (Revisione di Qualifica);

2.2 Dettaglio delle revisioni

2.2.1 Revisione dei Requisiti (RR)

– Funzione: concordare con il cliente una descrizione condivisa del prodotto
– Prodotti in ingresso:
  - Capitolato d'appalto
  - Incontri con il cliente
  - Analisi dei requisiti (AR)
  - Piano di Qualifica (PQ)
– Stato di uscita:
  - Prodotto descritto

2.2.2 Revisione del Progetto Preliminare (RPP)

- Funzione: attivazione della fase realizzativa del prodotto (accertamento di realizzabilità)
- Prodotti in ingresso:
  - Specifica Tecnica (ST)
  - Aggiornamento del Piano di Qualifica (PQ)
- Stato di uscita:
  - Prodotto specificato

2.2.3 Revisione del Progetto Definitivo (RPD)

- Funzione: informare il cliente delle caratteristiche definitive del prodotto: attivarne la fase di qualifica
- Prodotti in ingresso:
  - Definizione di Prodotto (DP)
  - Aggiornamento del Piano di Qualifica (PQ)
- Stato di uscita:
  - Prodotto definito

2.2.4 Revisione di Qualifica (RQ)

- Funzione: approvazione della campagna di verifica; attivazione della fase di accettazione
- Stato di ingresso:
  - Aggiornamento del Piano di Qualifica (PQ)
  - Include la specifica delle prove di accettazione
  - Versione preliminare del Manuale d'Uso (MU)
- Stato di uscita:
  - Prodotto qualificato

2.2.5 Revisione di Accettazione (RA)

- Funzione: accettazione del prodotto
- Stato di ingresso:
  - Versione definitiva del Piano di Qualifica (PQ)
  - Include l'esito delle prove di accettazione
  - Versione definitiva del Manuale d'Uso (MU)
- Stato di uscita:
  - Prodotto accettato

2.3 Risorse necessarie e risorse disponibili

Nel Piano di Progetto ([PP]) sono specificate le risorse disponibili nonché la loro organizzazione nel tempo. Le risorse oltre ad essere di natura tecnologica, riguardano anche l'aspetto umano con i relativi obblighi di tempo. L'amministratore di progetto dovrà eseguire un'attività di supervisione durante le attività di verifica affinché le risorse siano sufficienti ed efficienti per consentire il conseguimento degli obiettivi prefissati.

2.4 Strumenti, tecniche e metodi di verifica

Gli strumenti software adottati saranno prodotti open-source e/o freeware per non incidere ulteriormente sul costo totale del progetto. I metodi di verifica sono specificati in questo documento nei punti successivi. E' a disposizione dell'azienda un repository presso il quale saranno mantenuti tutti i file relativi al progetto. Tale repository giocherà un ruolo fondamentale durante tutto il periodo di sviluppo, permettendo una gestione più controllata ed efficiente dei file (documenti, codice o altro). Per maggiori dettagli consultare il documento [NI]. Ogni modifica apportata al repository verrà conservata nel tempo ed in qualsiasi momento sarà possibile procedere al ripristino di stati passati. Inoltre ogni modifica sarà riconducibile al suo autore, in modo tale da poter individuare eventuali responsabilità in caso di problemi.

3 Gestione amministrativa della revisione

3.1 Comunicazione e risoluzione delle anomalie

Qualora durante la fase di verifica venisse trovata un'anomalia, spetta al verificatore la notifica di tale fatto. Nel fare ciò, il verificatore, dovrà procedere con i mezzi e le regole definite nelle norme interne ([NI]). Tali notifiche dovranno specificare tra le altre cose:

- unità di analisi
- dati in ingresso
- dati in uscita attesi
- dati in uscita rilevati
- anomalia riscontrata
- possibile causa
- possibile soluzione.

Queste notifiche dovranno essere comunicate, oltre al redattore del file, anche al responsabile di progetto. Le anomalie dovranno essere estinte nel tempo più breve possibile, rispettando le priorità assegnate alla notifica (ogni notifica può infatti avere

una priorità più o meno alta a seconda della gravità dell'anomalia).

3.2 Trattamento delle discrepanze

Quando un requisito prefissato non viene soddisfatto, si parla di discrepanza. Una discrepanza può essere più o meno grave e può interessare un intero requisito o solo parte di esso. Si parla quindi di discrepanze totali o parziali. Quando una discrepanza viene individuata, si dovrà procedere secondo un iter preciso regolato dalle norme interne al Team. Per ciascuna discrepanza rilevata, si dovrà specificare il requisito non soddisfatto (e in caso di discrepanza parziale, la parte del requisito non soddisfatto), gravità della discrepanza, possibili cause e/o responsabilità ed infine le possibili soluzioni. Se i costi e i rischi per estinguere la discrepanza saranno troppo elevati, si potrà in alcuni casi decidere di tralasciarla; questo tipo di decisione dovrà poi essere comunicata al cliente e dovutamente motivata.

4 Accertamento di qualità

Per produrre software di qualità verrano seguite le procedure di controllo sopra descritte. Si cercherà di soddisfare le esigenze del cliente e di eventuali altri portatori d'interesse tramite il completo soddisfacimento dei requisiti analizzati, sia impliciti che dichiarati. Tramite incontri con il cliente si cercherà di accertare la qualità del nostro operato come azienda software. Durante tali incontri, che verranno fissati a progetto ben avviato, il committente potrà esprimere il suo giudizio a riguardo.

Le caratteristiche che il prodotto dovrà avere (grazie all'applicazione delle norme espresse in questo documento) saranno le seguenti:

- funzionalità: il prodotto dovrà essere in grado di offrire le funzioni prefissate e necessarie all'utente;
- affidabilità: dovrà avere una tolleranza ai guasti soddisfacente, cercando di prevenire l'insorgere di errori, e cercando di gestire eventuali situazioni anomale non previste;
- usabilità: dovrà essere comprensibile e di veloce apprendimento;
- efficienza: dovrà avere tempi di risposta ragionevoli, e le risorse richieste dovranno essere contenute;
- manutenibilità: dovrà essere facile da analizzare e verificare in modo tale da permettere la modifica di alcune parti del programma in modo efficace ed efficiente;
- portabilità: il prodotto finale dovrà essere in grado di adattarsi ad ambienti diversi.

5 Resoconto delle attività di verifica

5.1 Tracciamento UseCase - requisiti

UC1.1	RFO16,RFO17,RFP02
UC1.2	RFO01,RFO02,RFO03,RFO06,RFO07,RFO08,RFO09,RFO10,RFD01, RFP03
UC1.3	RFO01,RFO02,RFO03,RFO06,RFO07,RFO08,RFO09,RFO10,RFD01
UC1.5	RFO16
UC1.6	RFO24
UC1.4.1	RFO01,RFO02,RFO03,RFO06,RFO07,RFO08,RFO09,RFO10,RFO15, RFD01, RFD02
UC1.4.2	RFO04,RFO05,RFP01
UC1.4.3	RFO22
UC2.1	RFO18
UC2.1.1	RFO18
UC2.1.2	RFO18
UC2.2	RFO20
UC2.3	RFO19,RFD03
UC2.4	RFO23
UC2.5	RFO13,RFO14
UC2.5.1	RFO12
UC2.5.2	RFO11
UC3	RFO21
UC4	RFO23

Tabella 5.1: Tracciamento UseCase - requisiti

5.2 Dettaglio delle verifiche tramite analisi

L'analisi statica del codice è stata effettuata tramite l'uso degli strumenti:

- FindBugs
- RefactorIT
- PMD

Attraverso tali programmi il codice è stato verificato e corretto in modo da eliminare i più semplici e comuni errori di programmazione che non per questo dovrebbero essere sottovalutati.

I risultati prodotti sono riportati nel file [RAS].

5.3 Dettaglio delle verifiche tramite test

Per le verifiche tramite test verranno utilizzati gli strumenti definiti nelle norme di progetto [NI]. Lo strumento principale di cui il Team si servirà per l'esecuzione di test è NetBeans in combinazione con Junit. Con tali software verrano create delle nuove classi con l'unica funzione di test. Tali classi saranno nominate automaticamente con l'aggiunta di “Test” alla fine del nome della classe prima dell'estensione “.java”.

L'esecuzione di questi test potrà dare come output due soli possibili risultati: test passed o test failed. I risultati delle operazioni di test sono riportati nel documento [RT].

6 Modalità di verifica

6.1 Tracciamento

Al fine di garantire il soddisfacimento di tutti i requisiti è necessario lo svolgimento delle attività di tracciamento. Queste attività variano di fase in fase durante tutto il ciclo di vita. In seguito sono riportate le modalità con cui il tracciamento sarà effettuato in ciascuna fase:

- Analisi: vengono tracciati i requisiti classificati nel documento [AR], con gli UseCase (vedi tabella 5.1).
- Progettazione: tutti i requisiti funzionali vengono tracciati con i relativi componenti architetturali; a loro volta i componenti architetturali vengono tracciati con le singole unità logiche (vedi paragrafo 5 della Specifica Tecnica).

- Codifica: ogni unità logica viene tracciata con il modulo software che la implementa.
- Manutenzione: vengono mantenuti i vincoli definiti nelle fasi precedenti per quanto concerne il tracciamento dei requisiti iniziali. Ogni modifica/revisione, comporterà il controllo e l'eventuale aggiornamento, dei tracciamenti precedenti.

6.2 Verifica nell'analisi

L'analisi è l'attività che sta alla base di tutto il progetto. E' necessario, per tanto, che in questa fase non vengano commessi errori. Per far ciò, anche in questa fase dovranno essere effettuate procedure di verifica. In primo luogo il tracciamento consente di verificare la correttezza dei requisiti individuati, in modo da eliminare quelli inutili, modificare quelli poco chiari o non corretti, aggiungere quelli mancanti.

La verifica in questa fase si dovrà quindi concentrare nei seguenti punti:

- controllo che non vi siano requisiti mancanti, e in tal caso aggiungerli;
- verifica che ogni requisito sia quanto più chiaro e non ambiguo, consentendo alle fasi successive di procedere in modo rapido e corretto;
- verifica che non vi siano requisiti errati, che possano portare alla realizzazione di un prodotto non conforme alle aspettative del cliente e/o all'insorgere di costi non preventivati;
- verifica che la classificazione dei requisiti sia stata effettuata nel modo corretto, distinguendo i requisiti in: funzionali e non funzionali, impliciti ed espliciti;

6.3 Verifica nella progettazione

Tutta la fase di progettazione (sia architetturale, sia di dettaglio) si dovrà basare sulle decisioni prese nella fase precedente. La realizzazione dei diagrammi dovrà avvenire con gli strumenti e seguendo le regole specificate nelle Norme interne [NI]. In tale fase si dovranno realizzare più diagrammi, anche di tipologia diversa, che permettano di individuare in modo prima astratto, e poi dettagliato, le scelte architetturali e di dettaglio del prodotto. Tali scelte dovranno essere motivate e dichiarate in modo chiaro e corretto. Sarà così possibile proseguire con le fasi successive senza che vi siano ambiguità o mancanze di altro genere. Le attività di verifica in questa fase dovranno considerare i seguenti punti:

- aderenza dei diagrammi con le norme di progetto specificate nel file omonimo;
- correttezza dei nomi scelti, facendo in modo che il significato di tale componente/unità sia chiaro ed immediato
- correttezza dei collegamenti tra i vari componenti/unità, controllando le cardinalità, e i vincoli specificati
- controllo della corrispondenza con i requisiti stabiliti nella fase di analisi; tale controllo sarà possibile in parte anche con l'utilizzo del tracciamento;

Tutti questi controlli dovranno essere eseguiti dal verificatore preposto, il quale dovrà procedere con la lettura del documento Specifica tecnica [ST]. Il tracciamento dovrà essere riportato nel paragrafo preposto a tal fine nel presente documento.

6.4 Verifica nella codifica

La verifica in questa fase dovrà procedere secondo due diversi tipi di analisi (come quanto specificato al punto 2.1):

- Analisi statica
- Analisi dinamica

Nei punti successivi saranno trattate più in dettaglio queste due fasi.

6.4.1 Analisi statica

Tale tipo di analisi è caratterizzata dal fatto che il codice non viene eseguito. Per questo motivo la verifica avviene attraverso l'utilizzo di tecniche che analizzano staticamente il codice. Proprio per il fatto che il codice non viene eseguito, la verifica non può prendere in considerazione la validazione di requisiti funzionali. Inoltre la verifica tramite analisi statica non prende in considerazione il sistema nella sua interezza ma, piuttosto considera moduli indipendenti.

L'analisi statica che dovrà essere effettuata, si baserà principalmente sull'utilizzo di metodi di lettura del codice (desk-check):

- Inspection: consiste nella lettura mirata del codice. In questo tipo di attività il verificatore è nettamente distinto dal programmatore. Sarà necessaria la definizione di una lista di controllo che denoti i punti su cui si dovranno incentrare i controlli.

- Walkthrough: non agisce sulla base di presupposti. Viene realizzata leggendo il codice con occhio critico, percorrendo quelli che possono essere i probabili cammini logici di esecuzione del codice. Per tale attività viene richiesta la cooperazione di verificatori e sviluppatori, i quali dovranno operare in sintonia mantenendo separati i propri ruoli. Questa tipologia di controllo è più costosa della precedente e verrà, per questo motivo, applicata in misura inferiore rispetto ad attività di Inspection.

Oltre a queste tecniche di analisi, i verificatori saranno dotati di opportuni strumenti che permettano loro di effettuare analisi automatizzate. Questo tipo di analisi, consiste quindi nell'utilizzo di metodi formali implementati dagli strumenti specificati nelle norme interne [NI].

Le metriche che saranno oggetto di valutazione in questa fase di analisi sono:

- complessità ciclomatica: grado di complessità per ogni singola funzione; in linea di principio un buon grado di complessità ciclomatica non dovrebbe superare il fattore 10; saranno comunque accettare funzioni con complessità fino a 30. Un grado più elevato non sarà concesso in quanto la verifica di tale codice potrebbe diventare troppo complessa e inaffidabile. Si deve infatti considerare che il fattore di complessità ciclomatica corrisponde al numero esatto di casi di prova necessari per verificare ogni possibile scelta decisionale presa dalla funzione. Tale valore sarà calcolato in modo automatico con gli strumenti specificati nelle norme interne [NI], in modo tale da garantire l'affidabilità e la precisione.

- SLOC (Source Lines Of Code); classi troppo grandi rischiano di complicare la verifica del codice; in questi casi può essere utile spezzare in più classi diverse. Per il fatto che questa metrica è fortemente dipendente dallo stile di programmazione, non avrà molto peso nella misurazione del codice.

- Stile di programmazione corretto: rispetto della lunghezza massima di una riga, denominazione corretta delle variabili, ecc.

- Fan-In e Fan-Out: questa metrica sarà applicabile già in fase di progettazione. Misura le dipendenze di ciascun componente del sistema. Fan-In misura il numero di procedure che chiamano quella che stiamo analizzando (indice di uso); fan-out misura il numero di chiamate da quella analizzata (indice di accoppiamento).

- Coverage: inteso come *Statement coverage* e *Branch coverage.* Per maggiori informazioni vedere il punto 6.4.2.

In definitiva, l'analisi statica si concentrerà sui seguenti punti:

- analisi del flusso di controllo: controlla che il codice esegua nella sequenza attesa, individuando il codice non raggiungibile logicamente, e segnalando eventuali problemi di terminazione (nei cicli o nelle ricorsioni);
- analisi del flusso dei dati: controlla che ogni variabile, nel momento in cui viene utilizzata, sia stata inizializzata con un valore corretto; individua inoltre ogni altro problema relativo all'uso delle variabili;
- analisi del flusso d'informazione: tiene in considerazione le dipendenze tra le varie classi (sia in ingresso, sia in uscita).

6.4.2 Analisi dinamica

Viene attuata con l'esecuzione del codice oggetto relativo all'intero sistema, o in parti distinte. L'analisi dinamica dovrà avere inizio non appena sarà disponibile del codice testabile, evitando di rimandare tutta l'attività di test alla fine della codifica. In questo modo il processo di verifica sarà ottimizzato in modo tale da garantire in ogni istante il maggior numero di parti di codice verificate. Ogni prova potrà prendere in considerazione l'intero sistema, unità o anche moduli.

Le prove effettuate dovranno essere fatte in modo da bilanciare:

- la quantità minima di casi di prova necessari per fornire un grado di attendibilità (sulla qualità del prodotto) adeguato;
- la quantità massima di sforzo, tempo e risorse disponibili per il completamento della verifica.

Per fare in modo che i test eseguiti siano quanto più attendibili è necessario agire in modo da permettere la ripetibilità della prova; sarà perciò necessario associare ad ogni prova, oltre ai risultati conseguiti, anche le caratteristiche dell'ambiente in cui è stata eseguita.

Il punto chiave dell'analisi dinamica è tenere a mente che le prove non possono garantire l'assenza di errori, per questo motivo non bisogna sopravalutare affidabilità dei test eseguiti.

Per ogni test dovrà essere specificato:

- l'oggetto: la parte software che sarà testata;

- l'obiettivo del test: dovrà essere specificato per ogni caso di test; dovrà essere espresso in termini precisi e quantitativi;
- l'ambiente in cui è stato eseguito: per permettere la sua ripetibilità.

Gli elementi di cui si costituirà una prova sono:

- caso di prova (test case): corrisponde alla terna: ingresso, uscita, ambiente;
- batteria di prova (test suite): sequenza di casi di prova;
- procedura di prova: procedimento per eseguire, registrare, analizzare e valutare i risultati di una batteria di prove.

Ogni unità software dovrà essere soggetta a test. E' infatti noto che più della metà degli errori, è rilevata in questa fase di verifica. Il numero di prove da effettuare corrisponderà al numero di test necessari affinché ogni riga di codice sia stata eseguita almeno una volta (*Statement coverage*). Saranno poi eseguiti test che dovranno provare la correttezza di ogni cammino logico del codice (*Branch coverage*). Vengono qui definite alcune regole per l'integrazione dei moduli nelle unità:

- assemblare i moduli in modo incrementale: viene facilitata l'attività di ricerca di moduli difettosi;
- assemblare moduli produttori prima dei moduli consumatori: mantiene il flusso di controllo e il flusso dei dati corretti.
- fare in modo da permettere la reversibilità dell'integrazione.

Una volta eseguiti i test di integrazione, inizieranno quelli di sistema. Questi test verificano il programma nella sua interezza, ed hanno lo scopo di controllare che ogni requisito sia stato soddisfatto.

6.5 Verifica dei processi

Anche i processi saranno soggetti a controlli di qualità. La garanzia di qualità sui processi è ottenuta grazie all'implementazione di un ciclo PDCA per ciascun processo. Con questo meccanismo ogni processo è in grado di portare a termine la sua mansione, ed in più riesce a migliorarsi di volta in volta mediante l'applicazione di

norme correttive. Queste norme correttive vengano ricavate sulla base dei risultati prodotti da ogni processo.

Le fasi del ciclo PDCA sono:

- Plan (pianificazione): viene stabilito un piano che definisce cioè che il processo dovrà produrre, e in che modo lo dovrà fare;
- Do (esecuzione): il processo svolge le proprie mansioni;
- Check (controllo): i risultati prodotti vengono analizzati;
- Act (azione): vengono intraprese le opportune azioni correttive, sulla base del controllo effettuato al punto precedente.

7 Pianificazione ed esecuzione del collaudo

Il collaudo ha lo scopo di validare il prodotto, cioè verificare che ogni requisito funzionale sia stato soddisfatto. Nel punto seguente saranno specificate le modalità con cui StyloSoft intende verificare il soddisfacimento dei requisiti funzionali.

7.1 Specifica della campagna di validazione

Essenzialmente la validazione del prodotto SiGeM averà come specificato al punto 2.1, nella parte dedicata alla fase di collaudo del prodotto. Saranno cioè previsti collaudi interni (α-test) ed esterni (β-test). I primi saranno eseguiti unicamente all'interno dell'azienda i secondi potranno coinvolgere anche soggetti esterni.

Nei test interni si dovrà verificare anche grazie al tracciamento, che i requisiti funzionali siano tutti pienamente soddisfatti.

Per quanto riguarda i β-test, saranno disponibili all'utente varie versioni del programma presso il repository ufficiale del Team; il committente, così come ogni altro utente, potrà provare il programma e, nel caso venissero trovati errori, potrà segnalarli al nostro Team attraverso la mail stylosoft@gmail.com. Tali segnalazioni verrano utilizzate per la correzione di errori o comunque per il miglioramento del nostro prodotto.