Egoless Group
egolessgroup@googlegroups.com

Piano di qualifica per C04/PQS

Versione 4.0

Stato del documento:
Formale ed
Esterno

Sommario :

Il documento delinea la strategia di qualifica adottata dall'organizzazione per lo sviluppo del prodotto del capitolato **C04**: "**Sistema *software* per l'informatizzazione della gestione di qualità a norma ISO 9000:2000 nelle scuole**".

Redazione

Nominativo	Ruolo	Data
Eric Miotto	Verificatore	26 Novembre 2006
Roberto Pordon	Verificatore	30 Novembre 2006
Lorenzo Daniele	Verificatore	25 Gennaio 2007
Alberto Meneghello	Verificatore	25 Gennaio 2007
Lucia Meneghello	Verificatore	19 Febbraio 2007

Lista di Distribuzione

Nominativo	Ruolo
Tullio Vardanega	Committente
Renato Conte	Committente
Lucia Meneghello	Verificatore
Margherita Collicelli	Responsabile
Eric Miotto	Amministratore/Programmatore
Stefano Gazzola	Programmatore
Roberto Pordon	Verificatore
Lorenzo Daniele	Programmatore
Alberto Meneghello	Programmatore

Approvato da:

Versione	Nominativo	Data
4.0	Margherita Collicelli	16 marzo 2007

Registro delle Modifiche:

Versione	Autore	Data
4.0	Margherita Collicelli	16 marzo 2007
Approvazione documento per la Revisione di qualifica.		
3.2	Lucia Meneghello	16 marzo 2007
Verificato documento.		
3.13	Roberto Pordon	16 marzo 2007
Corretta sezione sulle misure.		
3.12	Lucia Meneghello	16 marzo 2007
Aggiunti riferimenti ai file delle matrici di tracciamento.		
3.11	Lucia Meneghello	15 marzo 2007
Integrata sezione 5.4 sulla verifica dei processi.		
3.1	Lucia Meneghello	14 marzo 2007
Apportate modifiche alla sezione 5.4 sulla verifica dei processi.		
3.01	Lorenzo Daniele	06 marzo 2007
Integrazione del capitolo relativo ai dettagli dell'esito delle revisioni (5.5.3).		
3.0	Lorenzo Daniele	23 febbraio 2007
Approvazione documento per la Revisione di Progetto Definitivo.		
2.2	Eric Miotto	23 febbraio 2007
Correzione e verifica del documento.		
2.14	Roberto Pordon	22 febbraio 2007
Inseriti strumenti di verifica, specificate metriche.		
2.13	Lucia Meneghello Roberto Pordon	21 febbraio 2007
Corretti errori, aggiornata sezione sulla verifica del codice.		
2.12	Lucia Meneghello	19 febbraio 2007
Aggiornata lista di distribuzione. Portate modifiche alla sezione Analisi dinamica.		
2.11	Roberto Pordon	17 febbraio 2007
Corretti errori ortografici, rimossi riferimenti alle versioni dei documenti.		
2.1	Lorenzo Daniele	10 febbraio 2007
Stesura sottocapitolo 5.5.2.		
2.0	Roberto Pordon	2 febbraio 2007
Approvazione documento per la Revisione di Progetto Preliminare.		

1.10	Lorenzo Daniele	2 febbraio 2007

Rivista la struttura del capitolo 2. Stesura degli attuali sottocapitoli da 2.5.1 a 2.5.5 compresi. Leggere correzioni al capitolo 4.

1.09	Alberto Meneghello	1 febbraio 2007

Risistemata la struttura dei capitoli.

1.08	Lorenzo Daniele Alberto MeneghelloS	31 gennaio 2007

Integrati capitoli 2.6 e 4. Spostamento della tabella di tracciamento componenti architetturali – requisiti nel documento di Specifica tecnica. Stesura capitolo 3.4 e 2.7.

1.07	Stefano Gazzola	31 gennaio 2007

Modificata intestazione tabella di tracciamento componenti architetturali – requisiti

1.06	Lorenzo Daniele Alberto Meneghello	30 gennaio 2007

Integrati capitoli 2.6 e 4. Spostato contenuto di 3.4 in 2.6.

1.05	Lucia Meneghello	30 gennaio 2007

Corretti errori ortografici. Aggiornata Lista di distribuzione.

1.04	Alberto Meneghello Lorenzo Daniele	25 gennaio 2007

Aggiunti termini al glossario. Integrati capitoli 3.4, 3.1.

1.03	Lorenzo Daniele	25 gennaio 2007

Aggiornamenti ai capitolo 2.6, 2.7, 3.4. Prima stesura sottocapitolo 2.6.2. Adeguamento al modello per i documenti.

1.02	Alberto Meneghello, Lorenzo Daniele	25 gennaio 2007

Stesura sottocapitolo 4.1.

1.01	Eric Miotto	18 dicembre 2006

Corrette alcune discrepanze.

1.0	Lucia Meneghello	11 dicembre 2006

Approvazione documento per la Revisione dei Requisiti.

0.9	Eric Miotto	11 dicembre 2006

Rivista sezione sulla gestione amministrativa della revisione, in particolare riguardo le definizioni di anomalia e discrepanza.

0.8	Roberto Pordon	09 dicembre 2006

Corretti alcuni errori ortografici.

0.7	Eric Miotto	09 dicembre 2006

Individuazione ed enfatizzazione termini da inserire nel glossario; riscrittura di alcune parti; resa più snella la sezione 5.1 sulla strategia del collaudo.

0.61	Eric Miotto	08 dicembre 2006

Adeguamento del documento, al modello e alle norme; aggiornamento lista di distribuzione.

0.6	Roberto Pordon	07 dicembre 2006

Revisione generale dell'organizzazione del documento. Aggiunte e correzioni in ogni sezione.

0.5	Roberto Pordon	03 dicembre 2006

Integrazione paragrafo 2.5 e 3.1

0.4	Eric Miotto	02 dicembre 2006

Applicazione del nuovo modello di documento, precisazione pianificazione strategia della verifica, corretti alcuni errori.

0.3	Roberto Pordon	30 novembre 2006

Corretti alcuni errori.

0.2	Eric Miotto	29 novembre 2006

Aggiornata introduzione, stesura parziale della strategia di verifica e della gestione amministrativa della revisione, prima stesura lista di distribuzione.

0.1	Eric Miotto	26 novembre 2006

Prima bozza del documento.

Indice

1 Introduzione

1.1 Scopo del documento

Il documento si prefigge di illustrare come il gruppo intende assicurare che:

- il prodotto sia di qualità (e quindi soddisfi le esigenze del cliente);
- il prodotto non presenti errori.

Verranno dunque descritti in dettaglio la conduzione del processo di verifica e del collaudo, delineando le strategie, le attuazioni e i risultati.

1.2 Scopo del prodotto

Per questo argomento si rimanda all'analisi dei requisiti [AR].

1.3 Glossario

Per il glossario si rimanda al glossario [G].

1.4 Riferimenti

- [12207] ISO/IEC Standard 12207:1995
- [AR] Analisi dei requisiti, Egoless Group
- [PP] Piano di progetto, Egoless Group
- [DP] Definizione del prodotto, Egoless Group
- [NP] Norme di progetto, Egoless Group
- [MU] Manuale utente, Egoless Group
- [ST] Specifica Tecnica, Egoless Group
- [STs] Specifica dei test, Egoless Group
- [G] Glossario, Egoless Group
- [SB] Guide to the Software Engineering Body of Knowledge (SWEBOK), versione 2004, IEEE
- [RR] Verbale sulla Revisione dei requisiti, Egoless Group, 19 Dicembre 2006
- [RPP] Verbale sulla Revisione di Progetto Preliminare, Egoless Group, 06 Febbraio 2007
- [RPD]Verbale sulla Revisione di Progetto Definitivo, Egoless Group, 27 Febbraio 2007

- [UML] Specifica di UML, versione 2.0, OMG
  http://www.uml.org/
- [JC] Java Coding Style Guide, Achut Reddy, Sun Microsystems
  http://developers.sun.com/sunstudio/products/archive/whitepapers/java-style.pdf
- Matrici di tracciamento
  - Trace_Requisiti_CasiUso.pdf
  - Trace_Requisiti_Componenti.pdf
  - Trace_CasiUso_Componenti.pdf
  - Trace_Moduli_Componenti.pdf

2 Visione generale della strategia di verifica

2.1 Filosofia

Lo standard ISO/IEC 12207:1995 basa l'organizzazione interna di ogni processo sul *ciclo PDCA* (Plan, Do, Check, Act). In quest'ottica la strategia di *verifica* e *validazione* (detta nell'insieme *qualifica*) assume un ruolo determinante in quanto permette che i *processi* producano i risultati attesi nei tempi previsti.

I processi di verifica e validazione accompagnano quindi l'intera durata del progetto ed applicano a tutte le fasi dello sviluppo.

2.2 Responsabilità

Verranno ora delineate le responsabilità dei vari ruoli nel processo di verifica.

Verificatore
Il verificatore è la figura chiave per quanto riguarda i processi di verifica e validazione, delle quali è responsabile e attore. La sua attività è condotta secondo le specifiche del presente documento e in conformità alle norme di progetto.

Il verificatore è il principale responsabile della strategia di qualifica e della stesura del presente documento.

Responsabile
Nell'ambito della verifica il responsabile si occupa di assegnare le persone ad ogni problema segnalato dal verificatore. Controlla la strategia proposta dal verificatore e ne propone aggiustamenti.

Amministratore
L'amministratore è tenuto a fissare le norme per la conduzione del progetto e mette a disposizione gli strumenti e l'infrastruttura necessaria allo svolgimento delle attività di verifica.

2.3 Risorse necessarie, risorse disponibili

La “risorsa” essenziale per effettuare verifica e validazione è la disponibilità di verificatori. Il responsabile e l'amministratore forniscono rispettivamente direzione e supporto a questo processo.

Per i dettagli dell'assegnazione delle risorse si rimanda a [PP].

2.4 Oggetto della verifica

Oggetto della verifica sono:

- **le fasi dello sviluppo del software**: analisi dei requisiti, progettazione, codifica;
- **i documenti che esse producono**, divisi nelle categorie:
  - *amministrazione*: piano di progetto (e conto economico preventivo), norme di progetto (incluse norme di codifica), glossario;
  - *sviluppo*: studio di fattibilità, analisi dei requisiti, analisi dei rischi, specifica tecnica, definizione di prodotto, documentazione del codice;
  - *manuali utente*;
- **e il prodotto software**.

2.5 Pianificazione strategica e temporale

L'attività di verifica è pressoché continua durante tutto lo sviluppo. Il verificatore, seguendo lo svolgimento delle varie fasi del progetto esegue tutti i controlli necessari a garantire il rispetto delle norme e la conformità dei prodotti alle richieste e alle necessità fissate nelle fasi precedenti. Da ciò si intuisce facilmente che i momenti critici per la verifica sono alla conclusione delle attività. È in questi momenti infatti che si fissano i criteri e le linee-guida per procedere nel lavoro. Il mancato rilevamento di errori o non conformità può portare a pesanti conseguenze nel seguito. Nei casi più gravi potrebbe persino rivelarsi necessario rivedere scelte e strategie fissate in fasi precedenti con effetti potenzialmente catastrofici in termini economici e di rispetto delle scadenze.

Per l'esatta pianificazione delle verifiche nel tempo, sulla base della seguente strategia, si rimanda al piano di progetto [PP].

2.5.1 Tracciamento

Il tracciamento, verificando necessità e sufficienza dei prodotti, applicherà al termine di ogni fase (per la pianificazione temporale delle varie fasi di sviluppo riferirsi a [PP]).

Verranno prodotti documenti formali che attestino il risultato della verifica.

Nel caso vengano riscontrati errori potrebbe essere necessario ripercorrere parte della fase appena conclusa.

Per un resoconto dettagliato sull'esito del tracciamento riferirsi al capitolo 5.1 del presente documento.

2.5.2 Verifica dei requisiti

La verifica dei requisiti avverrà sia in corso d'opera sia al termine di ogni fase di analisi (per la pianificazione temporale delle varie fasi di sviluppo riferirsi a [PP]).

Non saranno prodotti documenti formali riguardo l'esito della verifica, tranne per quanto riguarda il tracciamento dei requisiti.

Nel caso vengano riscontrati errori, a seconda della gravità, potrebbe essere necessario ripercorrere parzialmente o totalmente la fase di analisi.

Per un resoconto dettagliato sull'esito della verifica riferirsi al capitolo 5 del presente documento.

2.5.3 Verifica della progettazione

La verifica della progettazione avverrà sia in corso d'opera sia al termine della fase di progettazione garantendo che non siano introdotti errori nelle attività seguenti (per la pianificazione temporale delle varie fasi di sviluppo riferirsi a [PP]).

Non saranno prodotti documenti formali riguardo l'esito della verifica, tranne per quanto riguarda il tracciamento dei componenti.

Nel caso vengano riscontrati errori, a seconda della gravità, potrebbe essere necessario ripercorrere parzialmente o totalmente la fase di progettazione.

Per un resoconto dettagliato sull'esito della verifica riferirsi al capitolo 5 del presente documento.

2.5.4 Verifica della codifica

La verifica della codifica avverrà in corso d'opera ed al termine della fase di programmazione. In questa fase verranno effettuate sia analisi statiche che dinamiche. Inoltre ulteriori verifiche tramite analisi dinamiche sul codice oggetto saranno istanziate nella fase di verifica e nella fase di validazione (per la pianificazione temporale delle varie fasi di sviluppo riferirsi a [PP]).

Rappresentando il codice prodotto il fine ultimo dello sviluppo, alla verifica di questo viene attribuita un'alta criticità. Da ciò consegue che ogni attività di verifica sulla codifica (oltre all'attività di tracciamento) sarà formalmente documentata tramite appositi verbali.

Nel caso vengano riscontrati errori a seconda della gravità, potrebbe essere necessario ripercorrere parzialmente o totalmente la fase di progettazione.

Per un resoconto dettagliato sull'esito della verifica e della validazione riferirsi rispettivamente ai capitoli 5 e 7 del presente documento.

2.5.5 Verifica dei processi

L'attività di valutazione di un processo avverrà al termine del processo stesso, mentre in corso d'opera la qualità dei processi verrà garantita tramite apposite procedure di

controllo (riferirsi alla sezione 3.4).

Nel caso l'attività di verifica porti alla luce deviazioni dagli obiettivi qualitativi, verranno intraprese azioni correttive al fine di migliorare una futura istanziazione della stessa tipologia di processo (ad esempio in un altra iterazione).

Per un resoconto dettagliato sull'esito delle verifiche sui processi riferirsi al capitolo 5.4 del presente documento.

3 Gestione amministrativa della revisione

3.1 Definizioni

Anomalia
Con il termine *anomalia* si indica una differenza inaccettabile rispetto ad un'attesa. Ad esempio se una norma di progetto prevede l'inserimento del logo del gruppo in ogni documento, la mancanza del logo in un documento è un'anomalia.

Un'anomalia deve essere obbligatoriamente risolta facendo riferimento a norme o requisiti già definiti.

Discrepanza
Con il termine *discrepanza* si denota una differenza tra l'attesa e il prodotto ottenuto, che deriva da una mancata definizione dei dettagli. Una discrepanza contrasta con dei requisiti non definiti esplicitamente perché ritenuti in qualche maniera impliciti. Ad esempio: un amministratore definisce una norma che prevede che le date devono avere il mese scritto per esteso e non indica (perché lo ritiene implicito) che la lettera iniziale del mese deve essere minuscola. Se un autore scrive il mese con l'iniziale maiuscola in un documento siamo di fronte ad una discrepanza.

Una discrepanza può essere tollerata.

3.2 Comunicazione e risoluzione di anomalie

3.2.1 Verifica informale

Quando un membro del gruppo visiona un prodotto e riscontra uno o più errori, è tenuto a comunicare all'autore del prodotto gli errori rilevati, senza attendere l'intervento del verificatore.

L'attuazione di questo tipo di verifica non comporta la produzione di atti formali (documenti).

3.2.2 Verifica formale

Il verificatore, nello svolgere il suo compito, segue la seguente procedura:

- riscontra un'anomalia;
- segnala l'anomalia secondo le regole previste in [NP];
- il personale competente corregge l'anomalia;
- il verificatore constata la correzione dell'anomalia e la verbalizza secondo le regole previste in [NP].

3.3 Trattamento delle discrepanze

Il rilevamento di discrepanze dovrà essere trattato mediante confronto tra chi rileva la discrepanza e chi ne è responsabile. Riprendendo l'esempio della data, il verificatore può essere colui che rileva la discrepanza (perché vede che in alcuni documenti il mese è scritto con l'iniziale maiuscola mentre in altri no), mentre l'amministratore ne è responsabile.

Eventualmente anche il responsabile del progetto può essere coinvolto nel confronto al fine di ridurre al minimo l'impatto del problema.

Due sono i modi di trattare le discrepanze:

- tollerare la discrepanza;
- aggiornare i documenti conformemente alle conclusioni raggiunte e procedere alla rimozione della discrepanza.

Una discrepanza può essere parzialmente rimossa. Continuando l'esempio, le date possono non essere corrette per i documenti più vecchi che non vengono più aggiornati.

Si rimanda a [NP] per le regole da seguire nel trattamento delle discrepanze.

3.4 Procedure di controllo di qualità di processo

Mentre le verifiche di processo operano al fine di assegnare valutazioni quantitative alle metriche di qualità considerate, che acquisiranno valore oggettivo solo al termine del processo stesso (riferirsi al capitolo 4.5, Verifica dei processi), le procedure di controllo agiscono in corso d'opera al fine di garantire il rispetto degli standard, delle norme qualitative ([NP] e questo documento), delle tempistiche e della pianificazione ([PP]).

Figure chiave di questa attività risultano essere il responsabile, l'amministratore di progetto e i verificatori. Il primo assicurandosi che i piani e le scadenze vengano rispettate; il secondo fornendo e garantendo un utilizzo corretto delle risorse; gli ultimi assicurando il rispetto delle scelte progettuali comuni (quali standard e norme).

In conclusione tali procedure garantiscono il corretto svolgersi dei processi nel rispetto dei canoni qualitativi, dall'istanziazione fino alla conclusione, ma non si occupano in alcun modo di assegnare un giudizio comparativo, se non in via del tutto informale.

4 Tecniche e metodi di verifica

I metodi di verifica si distingueranno, a seconda dell'oggetto da verificare, in:

- **verifica delle fasi dello sviluppo del software**: avrà luogo sia in corso d'opera che al termine delle stesse. Come già spiegato al punto 2.5 la verifica dovrà essere più completa e rigorosa possibile e dovrà garantire che il passaggio da una fase all'altra del ciclo di vita non porti ambiguità, difformità rispetto ai requisiti, problemi di realizzazione, errori di qualunque tipo. Nel dettaglio le verifiche si concentreranno su:
  - *analisi dei requisiti*: verifica interna di fattibilità e completezza mediante raffronto con lo studio di fattibilità e il capitolato d'appalto. Successiva verifica esterna mediante colloquio/incontro con il committente.
  - *progettazione*: verifica di soddisfacimento dei requisiti tramite confronto con il documento di analisi dei requisiti ed eventualmente con l'analista. Verifica di compatibilità con il linguaggio di programmazione e gli strumenti disponibili per la fase realizzativa.
  - *codifica*: controllo dell'aderenza alle norme di codifica e di conformità con il progetto. Analisi statiche del codice (flusso di controllo, flusso dei dati ecc..), test di unità e di integrazione.
- **verifica dei documenti**: lo scopo è constatare in primis la chiarezza, la completezza e la consistenza dei documenti rispetto a ciò cui si riferiscono. Inoltre è da verificare la conformità degli stessi alle norme di redazione nonché l'aspetto prettamente linguistico.
- **verifica del prodotto software**: si rimanda alla sezione sul collaudo.

4.1 Tracciamento

Viene considerato rilevante al fine di garantire che i prodotti di ogni processo siano necessari e sufficienti a soddisfare i requisiti. Si propone, quindi, come attività base di verifica dei requisiti.

Data la natura del tracciamento, durante tale attività verranno considerati solamente i requisiti funzionali.

Tale metodologia assume forme diverse a seconda della fase del *ciclo di vita* considerata:

- Analisi dei requisiti:
  - Tracciamento dei requisiti (funzionali) individuati nelle relative fonti

autorevoli (tabella dei requisiti, [AR cap. 3.3, 3.4]).

- Associazione di ogni *caso d'uso* prodotto con il requisito (funzionale) d'origine (matrice di tracciamento [AR cap. 5.2]).

- Progettazione architetturale:
  - Mappatura di ogni *componente architetturale* nel caso d'uso corrispondente. (matrice di tracciamento [ST cap. 5])
- Progettazione di dettaglio:
  - Mappatura di ogni *unità logica* nella relativa componente architetturale
- Codifica:
  - Corrispondenza tra *moduli software e* corrispettiva *unità logica* implementata.
- Manutenzione
  - Registrazione delle modifiche (corrispondenza della modifica col relativo oggetto modificato) .
  - Mantenimento della consistenza del tracciamento agli altri livelli.

La verifica del tracciamento opererà appurando la necessità e sufficienza dei prodotti di ogni fase (come definiti nei punti sopra) facendoli risalire fino ai requisiti di sistema iniziali. La necessità assicura che ogni prodotto soddisfi un requisito, cioè sia risposta a qualche bisogno; essa identifica il tracciamento all'indietro. La sufficienza (anche detta completezza), invece, accerta che la totalità dei prodotti soddisfi globalmente tutti i requisiti, cioè che le cose previste siano state fatte; essa identifica il tracciamento in avanti. Perciò la verifica si baserà su un raffronto incrociato delle matrici di tracciamento sviluppate per assicurare questa corrispondenza prodotti-requisiti.

L'attività di tracciamento verrà documentata in maniera formale. Per un resoconto dettagliato sull'esito della verifica riferirsi alla sezione 5.1 del presente documento.

4.2 Verifica dei requisiti

Inquadrare e capire il problema è prerequisito indispensabile per sviluppare un buon prodotto. Data l'estrema importanza, è fondamentale effettuare una scrupolosa verifica della correttezza formale dei requisiti individuati.

Una prima verifica dei requisiti avviene tramite tracciamento all'indietro, verso fonti autorevoli (per l'esito dello stesso riferirsi al capitolo 5.1 di questo documento). Con ciò viene garantito che ogni requisito sia conseguenza diretta di un bisogno del cliente

(necessità).

Parimenti, il tracciamento in avanti assicura che tutti i bisogni del cliente siano stati presi in considerazione (sufficienza).

Data la natura del tracciamento, durante tale attività verranno considerati solamente i requisiti funzionali.

La verifica deve inoltre considerare:

- la chiarezza formale con cui il requisito viene espresso; si deve garantire la non ambiguità dello stesso;
- l'indipendenza di ogni requisito; occorre evitare inconsistenze e sovrapposizioni;
- una precisa ed uniforme classificazione dei requisiti (funzionali o non funzionali, a diversi gradi di importanza);
- l'identificazione univoca di ogni requisito.

La verifica opererà tramite lettura dei documenti Analisi dei Requisiti [AR], del capitolato [PQS] e degli eventuali documenti o file che costituiscono fonte autorevole. L'esito verrà registrato in maniera formale solo per quanto riguarda l'attività di tracciamento (sezione 5.1).

4.3 Verifica della progettazione

Innanzitutto verrà verificata l'aderenza sintattica dei diagrammi alle norme di progetto (riferirsi ad [NP]) e allo standard UML 2.0 ([UML]) .

Per quanto concerne l'aspetto semantico verrà posta attenzione sugli aspetti individuati come critici quali:

- correttezza della cardinalità delle associazioni;
- appropriatezza della scelta dei nomi in generale;
- esaustività dei metodi, ovvero che ogni classe contenga i metodi necessari e sufficienti per manipolarla; in particolare che esistano i metodi di inserimento, modifica e cancellazione per ogni oggetto sensibile;
- necessità e completezza di ogni singola componente (tramite tracciamento);
- coerenza e adeguatezza delle scelte architetturali.

La verifica stessa opererà tramite lettura dei documenti [ST] e [DP] e degli eventuali file, documenti o diagrammi allegati. L'esito verrà registrato in maniera formale solo per quanto riguarda l'attività di tracciamento (5.1).

4.3.1 Misure

Per la valutazione della progettazione saranno misurati gli Afferent Couplings (anche detti Structural Fan-in) e Efferent Couplings (Structural Fan-out).

Il valore Afferent Couplings di un componente e' il numero di altri componenti che lo usano. Questo valore indica in generale il livello di utilità e riuso.

Il valore Efferent Couplings di un componente e' il numero di componenti che esso chiama. Questo valore indica la sua dipendenza (accoppiamento) dal resto del codice.

Una buona progettazione dovrà mantenere basso il valore di Efferent Couplings per facilitare riuso, manutenibilità, portabilità, integrazione, test.

Per praticità queste misure saranno effettuate sul codice.

Le metriche corrispondenti sono definite in [NP].

4.4 Verifica della codifica

Verranno considerati globalmente i seguenti attributi riguardanti il codice (norme e linee guida esposte nei documenti di riferimento definiranno tecniche, metodi e criteri per la valutazione degli stessi, vedere [NP] e [JC]):

Legenda:

- Attributo: identifica il criterio di valutazione;
- Significato: breve descrizione dell'attributo di un frammento di codice;
- Tipologia di analisi: individua la tipologia d'analisi più adatta, tra statica e dinamica, a valutare l'attributo. Ciò non esclude che entrambe possano essere utilizzate;
- Documento di riferimento: linee guida che definiscono i criteri di valutazione dell'attributo.

Attributo	Posseduto dal codice se:	Tipologia di analisi	Documento di riferimento
Funzionalità	Soddisfa i requisiti per esso prefissati	Analisi statica	Analisi dei requisiti ([AR])
Affidabilità	Ha buona tolleranza alle eccezioni	Analisi dinamica	Analisi dei requisiti ([AR])
Usabilità	E' facilmente comprensibile da terzi	Analisi dinamica	Manuale utente ([MU])

Attributo	Posseduto dal codice se:	Tipologia di analisi	Documento di riferimento
Efficienza	L'esecuzione ha un basso costo in termini di risorse	Analisi dinamica	Norme di progetto ([NP])
Manutenibilità	Può essere facilmente modificato per adeguarsi ad un cambiamento dei requisiti	Analisi statica	Norme di progetto ([NP])
Portabilità	Può essere immerso in un ambiente esterno senza sforzi eccessivi	Analisi statica	Norme di progetto ([NP])

Ovviamente le definizioni si applicano anche al prodotto nel suo complesso.

Tutte le attività di verifica sul codice verranno formalmente documentate tramite appositi verbali. Per un resoconto dettagliato sull'esito della verifica riferirsi alla sezione 5. Per un resoconto sull'esito della validazione riferirsi alla sezione 7.

Per quanto riguarda le norme e le linee guida di codifica riferirsi a [NP].

4.4.1 Analisi statica

Poiché si interessa del codice sorgente, la complessità di tale tipologia di analisi può risultare troppo elevata per potersi rivolgere al sistema nel suo complesso. Perciò le tecniche statiche verranno applicate solo su singole componenti in ogni istante.

In fase di verifica verranno considerati entrambi i metodi di analisi statica:

- metodi manuali: a basso costo in termini di risorse, si basano sulla lettura del codice;
- metodi formali: si basano sulla verifica di soddisfacibilità di alcune proprietà logico-algebriche di particolari sezioni del codice.

Per un resoconto dettagliato sull'esito della verifica riferirsi alla sezione 5.2.

Metodi formali
Data la loro natura formale e conseguente complessità elevata, verranno utilizzati *tools* appositamente creati a tale scopo.

Metodi manuali
Verranno seguite due strategie ampiamente consolidate nell'ingegneria del software:

1. Walkthrough;
2. Inspection.

Entrambe coinvolgono sia verificatori che programmatori ma vi sono differenze sostanziali che si esplicano nella filosofia di fondo, nel modo in cui le due figure cooperano e nella durata.

La prima effettua senza alcun presupposto una verifica esaustiva su tutto il codice, simulandone l'esecuzione, e prevede un'ampia interazione tra le parti coinvolte così da sciogliere in corso d'opera qualsiasi dubbio relativo all'oggetto in esame; per questo risulta essere temporalmente più onerosa rispetto alla Inspection le quali, presupponendo un insieme definito di errori più frequenti (errori di programmazione, violazioni alle norme comuni...), tendono ad una lettura mirata del programma separando in modo rigido chi verifica e chi programma.

Riassumendo in forma tabellare

Walktrough	Inspection
Verifica tutto il codice	Verifica mirata del codice
Nessun presupposto	Presuppone gli errori da cercare basandosi su norme e linee guida
Ampia interazione verificatori – programmatori	Rigida separazione verificatori – programmatori
Durata maggiore	Durata minore

La seguente tabella esplica le attività con cui le due tecniche si svolgono ed è conseguenza diretta delle differenti idee di fondo viste sopra.

Walktrough	Inspection
Pianificazione	Pianificazione
Lettura del codice	Definizione della lista di controllo
Discussione	Lettura del codice
Correzione dei difetti	Correzione dei difetti

Misure
Sulle classi saranno effettuate le seguenti misure:

- **NCSS** (Non Commenting Source Statements): saranno misurati il numero di

istruzioni di metodi, classi e file.

- **complessità ciclomatica:** fornisce una misura della complessità del flusso di controllo. E' inoltre importante per stabilire a priori il numero di test che devono essere svolti su una componenti.

Le metriche corrispondenti sono fissate in [NP].

4.4.2 Analisi dinamica

Consiste nell'esecuzione controllata del codice oggetto prodotto, al fine di assicurarsi che soddisfi le aspettative delineate nei requisiti, affinate in fase di progettazione e specificate nella specifica tecnica [ST].

Nella progettazione del prodotto in esame si terrà conto delle esigenze derivanti dalla necessità di effettuare test, al fine di garantire un più completo possibile scandagliamento di ogni unità.

L'analisi dinamica o test sarà iniziata pressoché in parallelo alla fase di codifica al fine di evitare definizione di codice poggiante su dell'altro precedentemente steso, ma non verificato.

Nel definire la strategia di prova bilanceremo il rapporto tra la minima quantità di casi prova sufficienti a fornire un elevato livello di certezza sulla qualità del prodotto e la massima quantità di tempo e risorse disponibili al completamento della verifica. Un aiuto per determinare il numero di test necessari viene dall'impiego della complessità ciclomatica.

Ogni test sarà specificato come fosse esso stesso un sistema e i criteri seguiti nel fare ciò si focalizzano su:

- **Oggetto del test**: unità, intese come minima entità software identificabile, e relative integrazioni sulla base delle relazioni funzionali.
- **Obiettivo del test**: ne verrà specificato uno per ogni caso di test identificato, entrando il più possibile nel dettaglio. Dunque ogni oggetto avrà le sue prove, ognuna delle quali avrà uno specifico obiettivo.
- **Ripetibilità**: una prova con determinati input, applicata al modulo per cui è stata definita, deve produrre sempre i medesimi risultati e tali risultati valgono esclusivamente per tale prova, cioè non possono essere generalizzati.

E' importante tenere conto del fatto che i test si limitano ad individuare difetti qualora ve ne siano (producono un risultato negativo), e non sono quindi in grado di provarne l'assenza.

Tuttavia, i test dall'applicazione dei quali non emergono difetti, permettono di avere

garanzia di non regressione. Tale caratteristica ci permetterà di verificare la correttezza delle stime di tempo e costi previste nella specifica di ogni prova.

Entrando nel dettaglio degli elementi di una prova, nella specifica di ognuna saranno inseriti:

- **Caso di prova** (test case): i dati in ingresso, la relativa uscita prevista e l'ambiente in cui si svilupperà la prova relativamente ad un determinato oggetto.
- **Batteria di prove** (test suite): si riferisce ad una sequenza di casi di prova i quali dovranno essere in numero tale da garantire che la maggior quantità possibile di codice sia stata analizzata.
- **Procedura**: i passi effettuati (in modo automatico o meno) per eseguire, registrare e valutare i risultati di una batteria di prove.

In fede a quanto appena detto verranno definiti i test di unità.

Per quanto riguarda il dominio di ingresso, per sua natura infinito, è necessario definirne uno limitato ma caratterizzato da valori che permettano di individuare un numero soddisfacente di casistiche di risultati al fine di provare che l'unità è corretta nella sua completezza.

In particolare, tali ingressi, con i relativi esiti attesi, verranno utilizzati per effettuare test funzionali i quali hanno lo scopo di attuare lo *statement coverage*, cioè eseguire almeno una volta tutte le linee di comando di ciascun modulo.

Tale test tuttavia non entra nel merito della logica del codice dell'unità dunque si effettueranno anche test di tipo strutturale atti a eseguire *branch coverage*, cioè analizzare i vari cammini interni al modulo: i dati in ingresso precedentemente definiti devono dunque essere tali da attivare ogni singolo percorso.

Per dettagli e specifiche di test si rimanda al documento [STs].

Riferirsi al capitolo 6 per quanto riguarda la pianificazione di α-test e β-test.

4.5 Verifica dei processi

Il processo di verifica è il punto ideale per effettuare misurazioni sui processi e capire in che modo questi ultimi possono essere migliorati. Tale attività si propone di dare una valutazione comparativa ai processi istanziati e conclusi. Per questo verrà effettuata solo al termine di ogni processo. Invece, in corso d'opera, la qualità dei processi verrà sostenuta tramite apposite procedure di controllo (riferirsi alla sezione 3.4).

La qualità di processo viene assicurata seguendo le fasi del ciclo PDCA:

- Plan: vengono pianificate la tempistica di ogni processo ([PP]) e le risorse (umane e non) per una corretta istanziazione;
- Do: il processo viene concretamente istanziato secondo il piano (vedere anche sezione 3.4: Procedure di controllo di qualità di processo);
- Check: attraverso l'utilizzo di metriche quantitative, quindi oggettive, vengono caratterizzate l'efficacia e l'efficienza del processo, rispettivamente per verificare il grado con cui i prodotti dello stesso rispondono ai requisiti e per valutare quanto le risorse siano state effettivamente necessarie;
- Act: vengono applicate azioni correttive volte a migliorare il processo in termini degli oggetti della verifica

Le attività' di verifica della qualità dei processi trovano, quindi, forma nella fase di misurazione (check).

Si valuteranno i seguenti attributi di processo:

1. comprensibilità: esprime il grado in cui i membri del gruppo hanno compreso la finalità del processo;
2. rapidità: definisce se il particolare processo ha rispettato le tempistiche previste;
3. accettabilità: caratterizza la capacita' del processo di immergersi nella realtà preesistente;
4. visibilità: indica il grado di chiarezza e tangibilità dei prodotti del processo;
5. supportabilità: indica se la definizione del processo è tale da poter essere sostenuta o da altri processi o da strumenti;
6. affidabilità: è la misura con cui un processo riesce ad evitare l'insorgere di imprevisti (errore, guasto ...);
7. robustezza: definisce il grado in cui un imprevisto influisce sulla corretta prosecuzione del processo;
8. manutenibilità: indica la capacità del processo di adattarsi ai cambiamenti.

Per un resoconto dettagliato sull'esito della verifica riferirsi alla sezione 5.4.

4.6 Strumenti di supporto

A supporto dell'attività di verifica verranno impiegati i seguenti strumenti:

4.6.1 Issue tracking system

Si tratta di un ausilio software alla memorizzazione, al tracciamento ed alla risoluzione dei problemi riscontrati in fase di verifica.

I problemi sono gestiti tramite liste accessibili dal gruppo di progetto in lettura e scrittura. In esse i problemi vengono classificati secondo un grado d'importanza e ad essi sono associati attributi aggiuntivi quali data di immissione, creatore della segnalazione, una descrizione dettagliata, attività risolutive intraprese e grado di risoluzione. Inoltre è possibile assegnare la risoluzione di un problema ad un particolare membro del gruppo di progetto tramite liste specifiche (possibilmente gestite dal responsabile di progetto); eventualmente si potrà richiedere la riassegnazione della responsabilità ad altri, se adeguatamente motivata.

La segnalazione di un problema percorre le seguenti fasi:

1. un verificatore segnala l'esistenza di un problema al responsabile;
2. il responsabile verifica che il problema sia reale e non sia soltanto frutto di un'errata valutazione soggettiva;
3. il responsabile crea l'istanza del problema all'interno del sistema aggiungendovi tutti i dati rilevanti quali priorità e responsabilità di risoluzione;
4. il gruppo nel suo complesso viene avvisato dell'esistenza di un nuovo problema;
5. il problema viene risolto o eventualmente riassegnato, e all'interno del sistema viene aggiornato lo stato di avanzamento della risoluzione;
6. il sistema marca il problema come risolto.

4.6.2 Programmi per i test

I test si inquadrano nell'ottica dell'analisi dinamica eseguita sul codice (paragrafo 4.4).

Per la definizione degli strumenti utilizzati fare riferimento a [STs].

4.6.3 Programmi per l'analisi statica del codice

A supporto dell'analisi statica del codice si utilizzerà il programma FindBugs (findbugs.sourceforge.net) per la ricerca di "*bug patterns*" cioè di parti di codice che sono potenzialmente fonte di errori.

Per la valutazione della complessità ciclomatica si utilizzeranno i dati forniti dal programma Cobertura (cobertura.sourceforge.net).

4.6.4 Tools per il tracciamento

Per il tracciamento si utilizzerà il programma OSRMT (www.osrmt.com).

5 Resoconto delle attività di verifica

5.1 Dettaglio delle verifiche di tracciamento

Per la filosofia alla base della verifica di tracciamento riferirsi a 4.1 (Tracciamento – Tecniche e metodi).

Viene eseguito il tracciamento dei prodotti di ogni fase del ciclo di vita (riferirsi a [PP]) rispetto ai prodotti della fase precedente.

5.1.1 Analisi preliminare: tracciamento requisiti – fonti

Riferirsi alla tabella dei requisiti ([AR], sezione 3.1).

La tabella garantisce la necessità e sufficienza di tutti i requisiti relativamente alle fonti.

5.1.2 Analisi dei requisiti: tracciamento casi d'uso – requisiti

Riferirsi alla matrice di tracciamento ([AR], sezione 5.2) o alla matrice allegata.

La matrice garantisce la necessità e sufficienza di tutti i casi d'uso relativamente ai requisiti (riferirsi a [AR] per la lista dei requisiti e per i diagrammi dei casi d'uso).

5.1.3 Progettazione architetturale: tracciamento componenti – requisiti

Riferirsi alla matrice di tracciamento ([ST] capitolo 5) o alla matrice allegata.

La matrice garantisce la necessità e sufficienza di tutti i componenti architetturali relativamente ai requisiti (riferirsi a [AR] per la lista dei requisiti ed a [ST] per i diagrammi dei componenti architetturali).

5.1.4 Progettazione di dettaglio: tracciamento componenti di dettaglio – componenti architetturali

Fare riferimento alla matrice allegata.

5.1.5 Codifica: tracciamento moduli – componenti di dettaglio

Fare riferimento alla matrice allegata.

5.2 Dettaglio delle verifiche tramite analisi

5.2.1 Esito dell'analisi FindBugs su WSDidattica

La verifica del codice con FindBugs sul componente WSDidattica ha prodotto i seguenti

risultati:

Tipo di bug (# occorrenze)	Soluzione adottata
Read of unwritten field (34)	Tutti i campi segnalati sono in realtà istanziati al momento del deployment, quindi questo non è un bug.
Unwritten field (9)	Vedi sopra.
Method uses toArray() with zero-length array argument (11)	L'array passato per argomento serve solo a restituire un array del tipo voluto (vedi documentazione di java.util.List<E>) .
Consider returning a zero length array rather than null (11)	Questa correzione è fatta in automatico da NetBeans.

5.2.2 Esito dell'analisi FindBugs su APPDidattica

La verifica del codice con FindBugs sul componente APPDidattica ha prodotto i seguenti risultati:

Tipo di bug (# occorrenze)	Soluzione adottata
Comparison of String parameter/objects using == (3)	Corretto dal programmatore.
Confusing method names (6)	Manteniamo i nomi per uniformità.
Method invokes exit (1)	La chiamata ad exit è relativa alla normale terminazione del programma.
Non-transient non-serializable instance field in serializable class (23)	Riguarda i campi dati dei Form. Ignoriamo il problema.
Certain swing methods need to be invoked in Swing thread (20)	I metodi sono chiamati da classi Main di prova quindi ignoriamo il problema.
Usage of GetResource may be unsafe if class is exdended (7)	Per il momento ignoriamo il problema. Sarà corretto in seguito.
Null pointer dereference (1)	Riguarda una parte del programma in fase di sviluppo. Sarà corretto successivamente.
Suspicious equals() comparison (1)	Corretto dal programmatore.
Method returning array may	Corretto dal programmatore.

expose internal representation (1)	
Storing reference to mutable object (1)	Corretto dal programmatore.
Method invokes inefficient Boolean constructor (2)	Generato automaticamente da NetBeans. Ignorato.
Method allocates a boxed primitive just to call toString() (1)	Corretto dal programmatore.
Method invokes inefficient Integer constructor (4)	Generato automaticamente da NetBeans. Ignorato.
Method invokes inefficient String constructor (3)	Corretto dal programmatore.
Inner class could be made static (10)	Per il momento ignoriamo il problema. Sarà corretto in seguito.
Private method is never called (4)	Uno sarà usato in seguito. Gli altri 3 sono stati cancellati.
Class is not derived from an Exception (9)	Falso positivo di FindBugs.
Dead store to local variable (1)	Falso positivo di FindBugs.
Load of a known null variable (1)	Riguarda una parte del programma in fase di sviluppo. Sarà corretto successivamente.
java.lang.Exception is caught when Exception is not thrown (4)	Falso positivo di FindBugs.

5.3 Dettaglio delle verifiche tramite prove (test)

Riferirsi al documento [STs].

5.4 Resoconto sullo svolgimento dei processi

5.4.1 Primari

Fornitura:
Per quanto riguarda la gestione dei rapporti con il cliente (prof. Zambello) c'è stato qualche serio problema di coordinamento.

In seguito alla fase iniziale antecedente la presentazione dell'offerta, in cui il dialogo è stato scarno ma comunque utile per la definizione dei requisiti, il contatto è

praticamente cessato.

Ciò è accaduto per la difficoltà di reperire il cliente, poiché ogni tentativo di contatto attraverso mail si concludeva in una sua non risposta.

E' evidente che tale processo non ha funzionato tuttavia al momento sembra aver intrapreso la strada del miglioramento, grazie alla presa di coscienza da parte del cliente della necessità di creare una comunicare diretta con i fornitori (Egoless Group e Swell Systems).

Sviluppo
In questa fase rientra tutto il periodo di creazione del prodotto secondo le pianificazioni di tempi e costi fatte inizialmente.

Nel nostro caso si era stabilito, secondo il modello evolutivo, di svolgere due iterazioni in modo da dedicare la prima al soddisfacimento dei requisiti obbligatori del modulo gestione didattica e nella seconda di integrarli.

Il processo di sviluppo, rispetto tali pianificazioni, ha avuto una serie di adeguamenti in corso d'opera: ciò si è reso necessario, inizialmente per sopperire ai ritardi creati dalla definizione dei termini di collaborazione e successivamente per mediare le difficoltà di coordinamento, in termini di scadenze comuni, con il secondo gruppo di lavoro.

Allo stato attuale ci siamo visti costretti ad eliminare la seconda iterazione e a perseguire esclusivamente il soddisfacimento dei requisiti obbligatori del modulo gestione didattica.

Tuttavia, l'esperienza acquisita nel corso del progetto ci ha permesso, nell'ambito dei nuovi confini di sviluppo stabiliti, di definire rigorose scadenze interne e di coordinarci in maniera ottimale per rispettarle.

A tale proposito c'è da dire che, nel processo di sviluppo, sicuramente si è sempre realizzata una corretta definizione e assegnazione di ruoli e compiti; attraverso lo strumento dei tickets, messo a disposizione da Trac, si è provveduto ad assegnare ad ogni componente le proprie competenze in relazione al suo ruolo e le relative scadenze.

In questo modo il coordinamento interno si è mostrato essere corretto.

5.4.2 Di supporto

Documentazione:
La gestione a la stesura dei documenti si è rivelata pressoché ottimale in quanto fin

dall'inizio sono state definite delle norme rigorose e dettagliate nel documento preposto a tale scopo [NP].

Ad ogni aggiornamento di tale documento, l'amministratore ha provveduto ad avvisare tempestivamente gli altri collaboratori delle modifiche apportate, così da permettere loro di prenderne visione ed adeguarsi nell'immediato.

In questo modo i documenti venivano stesi fin da principio secondo lo standard interno, evitando di dover dedicare del tempo, ovviamente sprecato, all'adeguamento di eventuali discrepanze.

Verifica
In riferimento alla verifica sui documenti, il processo è stato intrapreso nei tempi e nei termini corretti; infatti, in procinto delle scadenze per le revisioni formali e non formali, i documenti venivano controllati ed eventualmente adeguati secondo le discrepanze o le imprecisioni rilevate dal verificatore.

Per quanto riguarda la verifica tramite tracciamento, malgrado le difficoltà a reperire un tool adeguato che ci permettesse di attuarlo in modo automatico, siamo comunque stati in grado di mettere in piedi delle tecniche che, pur essendo elementari (tabella disegnata a mano all'interno dei documenti contenente componenti e relativa correlazione identificata da una x sulla cella corretta), si sono rivelate ugualmente corrette ai fini della verifica.

Relativamente alla codifica invece, la verifica è stata intrapresa in ritardo. Tale affermazione va intesa nel senso che lo studio dei tool necessari e la loro configurazione secondo le nostre esigenze ha richiesto grossi sforzi in termini temporali ed intellettuali che si sono sommati ai ritardi accumulati per le motivazioni di cui al paragrafo relativo al processo Sviluppo.

Validazione
Al momento non ancora effettuata.

5.4.3 Organizzativi

Infrastruttura
L'infrastruttura di cui abbiamo avvertito l'esigenza e la necessità fin dall'inizio si componeva, oltre che dei pc personali di ogni componente, del server attraverso il quale condividere e versionare documenti e materiale relativo al progetto.

Alcuni problemi nel reperimento delle componenti necessarie al server, hanno dilazionato la sua messa in funzione.

Ciò ha sicuramente reso meno agevoli delle procedure (versionamento, organizzazione dei documenti, ..) di per se potenzialmente poco onerose.

Il fatto di aver avuto a che fare con operazioni poco snelle ha causato spreco di tempo che, sebbene nel singolo non sembrasse rilevante, nel complesso lo era.

Una volta reso operativo il server, invece, la dinamica delle procedure è migliorata e tutt'ora mostra i sui vantaggi in termini di tempo ed energie.

Relativamente ai pc personali, ogni qualvolta l'amministratore terminava lo studio di componenti software e le riteneva compatibili con le nostre necessità, provvedeva a comunicarlo agli altri componenti i quali tempestivamente provvedevano a reperile e installarle. Dunque da quest'ottica il processo ha funzionato in maniera corretta.

5.5 Dettaglio dell'esito delle revisioni

5.5.1 Revisione dei requisiti

La revisione dei requisiti è avvenuta il 19 Dicembre 2006.

Presenti alla revisione:

Tullio Vardanega	Committente
Renato Conte	Committente
Alberto Meneghello	Fornitore
Eric Miotto	Fornitore
Lorenzo Daniele	Fornitore
Lucia Meneghello	Fornitore
Margherita Collicelli	Fornitore
Roberto Pordon	Fornitore
Stefano Gazzola	Fornitore

Per l'esito riferirsi a
http://localwww.math.unipd.it/%7Econte/Valutazioni/RR/RR_EgolessGroup.pdf

o al documento allegato [RR].

5.5.2 Revisione di progetto preliminare

La revisione di Progetto Preliminare è avvenuta il 06 Febbraio 2007.

Presenti alla revisione:

Tullio Vardanega	Committente

Renato Conte	Committente
Alberto Meneghello	Fornitore
Eric Miotto	Fornitore
Lorenzo Daniele	Fornitore
Lucia Meneghello	Fornitore
Margherita Collicelli	Fornitore
Roberto Pordon	Fornitore
Stefano Gazzola	Fornitore

Per l'esito riferirsi a
http://localwww.math.unipd.it/%7Econte/Valutazioni/RPP/rpp_egoless.pdf

o al documento allegato [RPP].

5.5.3 Revisione di progetto definitivo

La revisione di Progetto Definitivo è avvenuta il 27 Febbraio 2007.

Presenti alla revisione:

Tullio Vardanega	Committente
Renato Conte	Committente
Alberto Meneghello	Fornitore
Eric Miotto	Fornitore
Lorenzo Daniele	Fornitore
Lucia Meneghello	Fornitore
Margherita Collicelli	Fornitore
Roberto Pordon	Fornitore
Stefano Gazzola	Fornitore

Per l'esito riferirsi a
http://localwww.math.unipd.it/%7Econte/Valutazioni/RPD/Valutazioni-RPD.html

o al documento allegato [RPD].

5.5.4 Revisione di qualifica

La sezione è vuota poiché la revisione non è ancora stata effettuata.

5.5.5 Revisione di accettazione

La sezione è vuota poiché la revisione non è ancora stata effettuata.

6 Pianificazione della validazione

6.1 Specifica della campagna di validazione (collaudo incluso)

6.1.1 Premessa

Nell'ambito dello sviluppo di software (ma non solo) la validazione del prodotto finito è considerata attività di particolare importanza perché essenzialmente consiste nell'accertarsi che il prodotto faccia ciò che è richiesto (*the product is the right system*).

In realtà, la validazione non dovrebbe presentare particolari sorprese. Se il processo di verifica ha funzionato correttamente durante tutto lo sviluppo, il controllo finale diventa solamente la constatazione che il prodotto soddisfa tutti i requisiti.

Per rendere possibile ciò è necessario porre particolare attenzione alla progettazione che deve essere studiata in modo da rendere più semplice e rapida possibile la ricerca e correzione di errori (*Construction for Verification* [SB] cap. 4 1.3)

6.1.2 Conduzione del collaudo

Il collaudo interno del prodotto software (α-test) consisterà in:

- **verifica finale del tracciamento**: controllo che nel passaggio tra le varie fasi di sviluppo non siano comparsi errori (riferirsi anche a 4.1 e 5.1).
- **prove di esecuzione**: per dare evidenza del fatto che il prodotto funzioni correttamente. Nell'ottica della strategia di verifica questo tipo di test non deve assolutamente configurarsi come tecnica di ricerca e soluzione di errori ma deve essere solo una prova che visualizzi, che renda “visibile” l'effettività del prodotto (riferirsi anche a 4.4 e 5.3).

Le modalità del collaudo esterno (β-test) saranno invece concordate con il committente.

7 Resoconto della campagna di validazione

Questa sezione è attualmente vuota dato che la validazione non è ancora stata eseguita.

7.1 Resoconto dei collaudi interni (α-test)

Questa sezione è attualmente vuota dato che il collaudo interno non è ancora stata eseguita.

7.2 Resoconto dei collaudi esterni (β-test)

Questa sezione è attualmente vuota dato che il collaudo esterno non è ancora stata eseguita.