Progetto SIGEOL

Piano di Qualifica

v3.0.0

Redazione: Carlo Scortegagna, Barbiero Mattia

19 maggio 2009

quixoft.sol@gmail.com

Verifica:	Scarpa Davide
Approvazione:	Beggiato Andrea
Stato:	Formale
Uso:	Esterno
Distribuzione:	QuiXoft
	Rossi Francesca
	Vardanega Tullio
	Conte Renato

Sommario

Piano di Qualifica per il progetto "SIGEOL", necessario per regolamentare le operazioni di verifica, validazione e controllo qualità durante il periodo di sviluppo.

Indice

1 Introduzione

1.1 Scopo del documento

Lo scopo di questo documento è di definire e pianificare le procedure che il team QuiXoft adotterà per garantire la qualità del prodotto denominato "SIGEOL". Requisito fondamentale per garantire un adeguato livello qualitativo è il rispetto delle specifiche definite in fase di analisi del progetto.

Particolare attenzione verrà posta anche nella definizione delle attività di verifica e validazione del materiale prodotto e, col procedere dello sviluppo, nella descrizione degli ambienti di prova e di collaudo.

1.2 Scopo del prodotto

Il prodotto "SIGEOL" si prefigge di automatizzare la generazione, la gestione, l'ottimizzazione e la consultazione degli orari di lezione.

Per ulteriori informazioni riguardanti scopi e funzioni del prodotto si prega di fare riferimento al documento Analisi dei Requisiti.

1.3 Glossario

Le definizioni dei termini specialistici usati nella stesura di questo e di tutti gli altri documenti possono essere trovate nel documento Glossario al fine di eliminare ogni ambiguità e di facilitare la comprensione dei temi trattati. Ogni termine la cui definizione è disponibile all'interno del glossario verrà marcato con una sottolineatura.

1.4 Riferimenti

- Capitolato d'appalto reperibile all'indirizzo:
  http://www.math.unipd.it/ tullio/IS-1/2008/Progetti/SIGEOL.html
- documento Analisi dei Requisiti alla sua ultima versione
- Norme di Progetto
- standard ISO/IEC 9126:2001

2 Visione generale

2.1 Organizzazione

Le attività finalizzate a garantire la qualità del prodotto "SIGEOL" inizieranno in concomitanza con l'inizio della fase di analisi del progetto, per garantire immediatamente l'assenza di errori o incongruenze.

Il controllo della qualità continuerà quindi per tutta la durata del progetto, sino alla consegna ed all'accettazione del prodotto finito. Durante tale

periodo di tempo sarà compito dei membri del team QuiXoft aggiornare il presente documento con i risultati aggiornati dei test e delle verifiche a cui il progetto verrà sottoposto. Al momento della consegna del prodotto, sarà compito del committente valutarne le conclusioni ed esaminare se i risultati rispetteranno appieno i requisiti stilati durante l'iniziale fase di analisi.

Nelle fasi successive all'accettazione del prodotto la gestione qualità non sarà più compito del team QuiXoft: per ulteriori informazioni sulla fase di manutenzione si prega di consultare il documento Piano di Progetto.

Qualsiasi risultato prodotto dal team QuiXoft dovrà essere sottoposto a verifica, sia esso un documento ufficiale, un file di codice sorgente o un qualsiasi altro risultato del lavoro del team.

Vista la natura del team di sviluppo, in cui tutti i componenti occuperanno a rotazione tutti i ruoli necessari, particolare attenzione sarà posta nel garantire l'assenza di conflitti di interesse: nessun Verificatore dovrà quindi trovarsi nella posizione di valutare un documento o un file sorgente prodotto da lui stesso in una precedente fase del progetto.

2.2 Obiettivi

Il lavoro del team, dalla fase di consegna del capitolato all'accettazione del prodotto finale, dovrà essere svolto nell'ottica di ottenere un risultato che soddisfi le esigenze del committente. Sarà responsabilità di tutti i componenti ottenere un prodotto finale che soddisfi pienamente tutti i requisiti esposti.

L'obiettivo di ogni componente del team QuiXoft è di completare le proprie mansioni nel miglior modo possibile, di rispettare le Norme di Progetto, di restare fedele a quanto prestabilito nel Piano di Progetto per quanto riguarda le ore massime di impegno personale ed il tetto massimo di spesa per portare a compimento l'intero progetto.

Ogni attività del team deve essere svolta puntando all'obiettivo finale di realizzare un prodotto di qualità, finalizzato al soddisfacimento:

- **del committente,** realizzando un prodotto che soddisfi pienamente i requisiti, che sia funzionale, usabile e pratico, che sia affidabile, efficiente e sicuro;
- **del team stesso,** realizzando un progetto qualitativamente alto che dia la prospettiva di sviluppi futuri, che possa essere riusato facilmente, che sia facilmente manutenibile, che sia portabile.

2.3 Caratteristiche del prodotto

Il risultato dal lavoro del team QuiXoft dovrà il più possibile aderire ai parametri di qualità descritti nello standard ISO/IEC 9126:2001. Le princi-

pali caratteristiche che il prodotto dovrà garantire per essere considerato di qualità sia dal committente sia dal team stesso saranno:

- **Funzionalità:**
  - Utilità: fare esattamente quello per cui è stato progettato
  - Accuratezza: funzionare rispettando i requisiti
  - Interoperabilità: interagire facilmente con l'esterno
  - Conformità: conformarsi a norme e standard
  - Sicurezza: impedire accessi non autorizzati
- **Affidabilità:**
  - Maturità: garantire una bassa frequenza di fallimenti dovuti a errori
  - Tolleranza ai guasti: mantenere le prestazioni prefissate in caso di errori
  - Ripristinabilità: ristabilire i dati perduti in caso di errori
- **Usabilità:**
  - Comprensibilità: poco sforzo necessario per comprenderne l'uso
  - Apprendimento: facilitare la comprensione dell'utente
  - Operabilità: agevolare l'utente nel controllo del software
- **Efficienza:**
  - Rispetto dei tempi di risposta e di esecuzione
  - Rispetto delle risorse utilizzate
- **Manutenibilità:**
  - Analizzabilità: facilitare la diagnostica degli errori
  - Modificabilità: agevolare la manutenzione futura
  - Stabilità: rischi controllati in caso di modifica
  - Verificabilità: validazione semplificata dopo una manutenzione
- **Portabilità:**
  - Adattabilità: adattamento ad ambienti diversi senza modifiche aggiuntive
  - Installabilità: permette l'installazione in uno specifico ambiente
  - Conformità: aderire a standard, norme e convenzioni di portabilità
  - Sostituibilità: sostituire elementi dell'ambiente esterno

Nel caso di sviluppo di prodotti che possiedono un interfaccia via pagina web, sarà altresì considerata caratteristica importante la validazione del codice tramite gli strumenti messi a disposizione dal consorzio W3C. Per maggiori informazioni si può visitare il sito *http://www.w3.org*.

2.4 Responsabilità

Ogni componente del team QuiXoft è tenuto a svolgere il lavoro a lui assegnato con precisione, puntando a renderlo il più possibile esente da errori. Primo requisito per tenere alto il livello di qualità dei lavori svolti è quindi di

leggere con attenzione le Norme di Progetto, e di applicarne alla lettera il contenuto.

Nonostante ciò, è ovviamente sempre presente una percentuale di incertezza sulla perfezione del lavoro svolto, ed è proprio compito del presente documento illustrare le procedure da seguire in caso di errori, imprecisioni o incongruenze, al fine di ottenere il maggior livello qualitativo possibile.

Le figure chiave per la gestione della qualità all'interno del team sono:

- **Verificatore:** è colui che effettivamente controlla ed esegue i test sul materiale prodotto dai membri del team. Ogni documento, file o prodotto passa necessariamente attraverso le sue verifiche, e senza il suo responso positivo nulla può essere dichiarato ufficiale. Se durante una fase del progetto vi fosse la necessità di avere più di un verificatore, sarà compito del Responsabile eleggere un rappresentante che coordini le loro attività interne;
- **Responsabile:** suo compito è di convalidare le modifiche proposte dai Verificatori e di controllarne l'operato;
- tutti gli altri ruoli previsti all'interno del team (Analista, Progettista, Programmatore, Amministratore) sono tenuti a sottoporre il proprio lavoro al Verificatore per le fasi di controllo e test. E' loro responsabilità anche apportare le modifiche che il Verificatore avrà segnalato essere necessarie e che il Responsabile avrà convalidato.

2.5 Pianificazione

Come detto in precedenza, il controllo qualità parte già dall'inizio del progetto: anche le attività di analisi dei requisiti e di studio di fattibilità necessitano della verifica da parte di qualcuno estraneo alla redazione, per evitare errori o incongruenze. Tali attività di controllo del lavoro svolto continuano fino alla consegna del prodotto finale.

Durante lo sviluppo del progetto saranno svolte delle revisioni per saggiarne lo stato di salute. In queste occasioni sarà possibile valutare il livello di qualità del lavoro svolto.

I processi di revisione previsti saranno di due tipi:

1. Revisioni formali:
   - revisione dei requisiti (RR)
   - revisione di accettazione (RA)
2. Revisione informali:
   - revisione del progetto preliminare (RPP)
   - revisione di qualifica (RQ)

Le revisioni formali verranno svolte in presenza del committente: uno scarso livello qualitativo del prodotto in questo caso comprometterebbe l'assegnazione (nel caso della RR) o l'approvazione (nel caso della RA) del prodotto, vanificando gli sforzi del team. Le revisioni formali, infatti, hanno effetto sanzionatorio.

Le revisioni informali invece verranno svolte in presenza del docente di riferimento, e avranno scopo puramente informativo: nel caso di qualità inadeguata, quest'ultima verrà solamente notificata e il team potrà recuperare gli eventuali errori fatti.

Per la pianificazione oraria delle attività di verifica durante le varie fasi del progetto e per le date delle revisioni si prega di consultare il documento Piano di Progetto.

2.6 Risorse

La gestione qualità dell'intero progetto impegna risorse umane come tecnologiche, ed è compito dell'Amministratore coordinarne l'utilizzo.

Una stima dei tempi che i Verificatori dedicheranno al controllo della qualità è consultabile sul Piano di Progetto. Nonostante ciò, non è da escludere che tali ipotesi di impegno siano inadeguate. Il livello di carico infatti varierà in base all'onerosità e alla tipologia del processo in esame, ed è arduo preventivare con precisione i tempi necessari per assicurare un corretto e preciso svolgimento delle attività di verifica.

Sarà compito dell'Amministratore gestire l'uso delle risorse umane e tecnologiche per garantire che siano sufficienti ed efficienti per consentire una verifica di buon livello.

2.7 Tecniche e medoti di verifica

La qualità dei processi è critica per la qualità del prodotto, quindi il team QuiXoft si impegna a garantire e migliorare la qualità dei propri processi.

Per far ciò il prodotto che si intende sviluppare, nonchè ogni processo istanziato, saranno sottoposti a costante attività di verifica in ogni fase di lavoro descritta nel Piano di progetto. Di seguito sono riportate le metodologie per attuare tale attività.

2.7.1 Verifica del codice prodotto

L'intero sviluppo del sistema "SIGEOL" sarà guidato dalla metodologia *Test-Driven Development* (d'ora in poi TDD).

La strategia di sviluppo TDD enfatizza in modo formidabile i test automatizzati per verificare la correttezza del codice. Invece di costruire una struttura di test alla fine dello sviluppo dell'applicazione, il TDD prevede che la scrittura del test sia la fase iniziale del processo. Quindi l'implementazione di una nuova funzionalità viene preceduta dalla definizione dei test

che ne stabiliscono la correttezza formale e semantica. Questo approccio garantisce che il codice prodotto risulti naturalmente modulare ed i moduli possano essere testati in isolamento gli uni dagli altri.

Il membri del team QuiXoft dovranno quindi, nella fase denominata *Progettazione*, focalizzare la loro attenzione sulla scelta di un linguaggio di programmazione, un framework ed un insieme di plugin che facilitino ed incoraggino l'uso di una tale strategia. Sarà poi compito degli addetti alla fase denominata *Realizzazione* aderire il più possibile a questa metodologia.

Le diverse tipologie di test sono le seguenti:

- **Test di unità:** ha lo scopo di verificare ogni unità software affinchè soddisfi i requisiti previsti mediante l'impiego di un insieme di dati in ingresso;
- **Test di integrazione:** ha lo scopo di verificare la cooperazione di più unità software;
- **Test di sistema:** ha lo scopo di verificare il soddisfacimento dei requisiti di tutto il sistema, in relazione anche all'ambiente d'installazione.

Sarà compito del Verificatore accertarsi della corretta applicazione di tale metodologia ed inoltre sarà incaricato di effettuare un'analisi del codice per evitare un'eccessiva ridondanza od una complessità elevata sia in termini di manutenibilità che riguardo all'efficenza. A tal scopo si cercheranno di utilizzare strumenti di benchmark che facilitino la misurazione delle performance del sistema.

2.7.2 Verifica della documentazione

L'incaricato alla verifica di ogni documento dovrà accertarsi che gli argomenti trattati siano comformi a quanto presente nel *Sommario* e nella sezione *Scopo del documento* e che il contenuto di ogni sezione rispetti il titolo assegnato alla stessa.

Inoltre dovrà segnalare parti del documento che risultino poco chiare od ambigue, eliminare porzioni di testo ritenute superflue e privare il documento di ogni errore ortografico, grammaticale o dattilografico.

2.7.3 Ambienti di prova e di collaudo

Con il procedere del progetto, come anche specificato nell' ANALISI DEI REQUISITI, si renderà necessario organizzare degli ambienti di prova e di collaudo per il software prodotto, per rilevare errori o inconsistenze nei dati. Nei prossimi rilasci del presente documento ne verranno specificate le caratteristiche e le funzionalità.

3 Automazione delle attività di verifica

La procedura di verifica può risultare spesso tediosa e ripetitiva se svolta in modo completamente manuale, nonchè l'assenza di un'adeguata infrastruttura a supporto di tale attività può essere fonte di errori ed imprecisioni.

Per questo motivo il team QuiXoft utilizza una serie di applicativi che facilitino ed automatizzino il più possibile l'attività di verifica, rendendo disponibili i risultati quando questi saranno necessari.

3.1 Tickets e consultazione ore di lavoro

Quando un compito è assegnato ad un membro tramite un ticket, il Responsabile stimerà le ore di lavoro necessarie per lo svolgimento dell'incarico ed annoterà tale numero nell'apposito spazio presente nel ticket.

Il componente che ha accettato il ticket, una volta completata la propria mansione, registretà le ore effettive di lavoro che potranno essere superiori alle aspettattive del Responsabile. Tramite gli strumenti messi a disposizione dal sito *http://www.assembla.com*, è immediata la consultazione del carico di lavoro e quindi il Responsabile, od un suo delegato, potrà apportare, se lo ritiene necessiario, le dovute modifiche al Piano di Progetto.

3.2 Validazione delle pagine web

L'adesione agli standard proposti dal consorzio W3C nella creazione di pagine web è un elemento fondamentale per la corretta visualizzazione delle stesse attraverso diversi tipi di browser. Per questo motivo il team QuiXoft utilizzerà gli strumenti messi a disposizione da tale consorzio per validare le pagine create.

Questo compito risulta spesso ripetitivo e, se svolto manualmente, produce un'eccessivo consumo di ore di lavoro. Per questo motivo verrà utilizzato un plugin che validerà ogni pagina creata in modo automatico e renderà disponibili immediatamente i risultati di tale validazione per poter eseguire le adeguate correzioni. Per maggiori informazioni consultare il sito http://code.google.com/p/htmltest/

3.3 TDD e Mock objects

Come precedentemente dichiarato, l'intero sviluppo del progetto sarà guidato dalla metodologia Test-Driven Development (confronta sezione 2.7.1). Ogni componente del team, nell'arco di tutta la durata del progetto, dovrà ricoprire ogni ruolo (per maggiori dettagli consultare il documento Piano di Progetto), quindi ogni membro ha discrete compotenze riguardo la stesura del codice. Per questo motivo il Verificatore di turno (o i Verificatori se ne sono previsti più di uno) avrà il compito di sviluppare i test anticipatamente alla stesura del codice. In questo modo il Programmatore

avrà il solo compito di implementare le funzionalità che sono richieste dai test.

Per facilitare la scrittura di test verrà utilizzato un plugin che permetta la creazione di Mock object, letteralmente "oggetti di pezza", in altre parole un oggetto che simula il comportamento di uno reale, permettendo di isolare una componente del sistema dalle altre per poterla testare al meglio. Un mock object non ha solamente metodi per simulare un oggetto reale, ma ha un'implementazione dei metodi propri di tale oggetto nella quale viene verificato il comportamento di chi usa l'oggetto, per esempio verificando che i metodi siano richiamati in un certo ordine o con certi parametri.

La scelta dei plugin è ricaduta su Mocha alla versione 0.9.5 in quanto utilizzato anche dal team di sviluppo di Ruby on Rails per effettuare la verfica del codice prodotto. Per maggiori informazioni consultare il sito *http://mocha.rubyforge.org/*.

Per la parte dell'applicazione sviluppata in Java (in particolare per la componente MiddleMan) verrà invece utilizzato il framework MockRunner (http://mockrunner.sourceforge.net/) il quale permette di simulare un'ambiente J2EE ed effettuare unit test.

3.4 Metriche sulla copertura dei test sul codice

Durante l'implementazione da parte del Programmatore delle funzionalità richieste dai test, è possibile che venga aggiunto del codice che non verrà mai testato. A questo scopo sarà utilizzato uno strumento per il linguaggio di programmazione Ruby che, in modo completamente automatico, esplori il codice sorgente segnalando quelle porzioni che non vengono verificate da nessun test.

Lo strumento renderà disponibili le porzioni di codice non testato, nonchè la percentuale di codice coperta da test. Sarà quindi auspicabile che tale percentuale raggiunga il 100% prima della consegna del prodotto.

Per maggiori informazioni è consigliata la consultazione del sito *http://rubyforge.org/projects/rcov/*.

4 Gestione amministrativa

4.1 Comunicazioni

Come spiegato nel documento Norme di Progetto, ad ogni prodotto del lavoro del team è assegnato un ticket, consultabile sul sito di appoggio *http://www.assembla.com*. Per prodotto si intende ovviamente qualsiasi risultato del lavoro dei componenti del team QuiXoft (documentazione, codice sorgente, moduli o programmi completi, ecc...).

Al momento del completamento di un ticket accettato dovrà esservi associato un altro ticket che ne richieda la verifica. Il nuovo ticket verrà

assegnato all'unico Verificatore nel caso ce ne sia solo uno, oppure al rappresentante eletto nel caso ce ne siano più d'uno. Sarà compito di quest'ultimo completare la verifica richiesta o inoltrare il ticket a qualche suo sottoposto.

Con questa procedura si assicura che ogni compito svolto verrà poi controllato dal Verificatore designato.

4.2 Risoluzione anomalie

Nel caso il Verificatore noti qualche anomalia nelle fasi di controllo e verifica, dovrà inserire i risultati all'interno del relativo ticket e chiuderlo segnandolo come completato.

Sarà cura del Responsabile poi controllare le modifiche proposte dalla fase di verifica, e nel caso di approvazione creare un nuovo ticket all'autore del prodotto testato per fargli apportare i dovuti cambiamenti.

La procedura appena descritta dovrà venir svolta finchè il Verificatore non notificherà più errori nei suoi test.

4.3 Trattamento delle discrepanze

La procedura per il trattamento delle discepanze è formalmente equivalente a quella appena elencata per la risoluzione delle anomalie.

L'unica differenza risiede nel fatto che il Verificatore che la nota dovrà inserire, nel resoconto del ticket che conclude la sua fase di verifica, anche il prodotto da cui la discrepanza è nata.

Se delle discrepanze dovessero presentarsi frequentemente tra i documenti prodotti e le Norme di Progetto, il Verificatore sarà tenuto a notificarlo all'Amministratore, il quale, in collaborazione con il Responsabile, procederà ai dovuti provvedimenti.

Non è infatti tollerabile che del tempo venga sprecato per correggere e notificare errori che potrebbero essere facilmente evitati leggendo e seguendo con più attenzione le Norme di Progetto.

5 Resoconto delle attività di verifica

Ogni risultato ottenuto dalle attività di verifica, validazione e qualifica dovrà essere attentamente elencato in questa sezione, in modo da assicurare che tutti i problemi e le relative soluzioni siano tracciate per garantire la massima qualità possibile del prodotto finale.

5.1 Dettaglio delle verifiche tramite analisi

5.1.1 Documentazione consegnata

Tutta la documentazione prodotta per essere consegnata alle revisioni è accuratamente controllata dal Verificatore di turno.

L'analisi statica effettuata su questi documenti si basa principalmente sull'accertarsi che le regole elencate nelle Norme di Progetto siano perfettamente rispettate, che non siano presenti errori grammaticali, che il testo sia scorrevole e di facile comprensione per il lettore e, infine, che non siano presenti errori di impaginazione o altri problemi che complichino la consultazione della documentazione.

La seguente lista contiene un sunto dei risultati di tali verifiche, e verrà aggiornata al proseguire dello sviluppo del progetto. Per la lista completa degli errori rilevati e delle relative correzioni è sempre possibile consultare l'elenco dei ticket aperti e conclusi con successo dai vari Verificatori del team QuiXoft durante i vari periodi.

- documentazione per la **Revisione dei Requisiti:** numerosi gli errori grammaticali e sintattici rilevati nella documentazione, corretti celermente dai rispettivi redattori. All'inizio del periodo grande sforzo è stato impiegato dai Verificatori nella segnalazione di gravi problemi di impaginazione della documentazione: ciò è dovuto alla poca esperienza nell'uso di LaTeX e dei relativi template. Col procedere dei giorni tali problemi sono stati completamente risolti, ed ora i template sono ritenuti totalmente affidabili ed esenti da errori. Discorso similare riguardo le slide per la presentazione: dopo i primi giorni di incertezza dovuti all'instabilità del template, si è risolto il tutto ed anche questo template è da ritenersi affidabile.

- documentazione per la **Revisione di Progetto Preliminare:** particolare attenzione è stata posta nel controllare con pari attenzione tutta la documentazione: in fase di Revisione dei Requisiti, infatti, sono stati trovati dal Committente banali errori grammaticali nel Piano di Progetto, che potevano benissimo essere evitati dedicando maggiore attenzione nella sua verifica. Anche in questa seconda fase di sviluppo sono stati segnalati errori ed incertezze grammaticali. Si è comunque rilevata una diminuzione rispetto al periodo precedente, indice di una maturazione ed una maggiore attenzione dei membri del team QuiXoft.

- documentazione per la **Revisione di Qualifica:** le operazioni di verifica della documentazione sono state svolte come nei periodi precedenti, segnalando e portando alla correzione alcuni errori e varie incertezza sintattiche che rendevano difficoltosa la lettura. Particolare attenzione è stata posta invece nell'evitare errori in fase di consegna della documentazione: nella consegna precedente, infatti, era stata consegnata una versione dell'Analisi dei Requisiti compilata con le immagini sbagliate. Successivamente, grazie alla disponibilità del Docente di riferimento, il problema è stato risolto con una nuova consegna del suddetto documento, ma tali errori andrebbero assolutamente evitati.

5.2 Dettaglio delle verifiche tramite test

Per l'esecuzione delle attività di verifica tramite test verranno utilizzati i moduli e le classi rese disponibili dal framework Rails, e più precisamente ogni test di unità dovrà estendere la classe ActiveSupport::TestCase ed ogni test funzionale dovrà estendere la classe ActionController::TestCase.

5.2.1 Unit Tests

Nello sviluppo di un'applicazione tramite il framework Rails, i test di unità (unit test) sono specifici per la verifica degli elementi appartenenti al componente Model. Data la particolare importanza di questa componente in quanto gestisce la persistenza dei dati, il team QuiXoft ha scelto di effettuare gli unit tests attraverso l'uso di fixtures, ovvero con istanze reali di un determiato model opportunamente salvate in file con estensione `yml` all'interno della directory `test/fixtures`. Prima dell'esecuzione della verifica il database di test sarà inizializzato con i dati contenuti nelle fixtures che vengono utilizzate in quel particolare test.

Ogni classe che implementa un insieme di test per un particolare model dovrà essere denominata `NomeModelTest` ed essere salvata su di un file chiamato `nome_model_test.rb` all'interno della directory `test/unit`.

Un esempio di unit test per il model `address` e delle relative fixtures utilizzate è dato dalla seguente porzione di codice:

nel file `address_test.rb`

```
class AddressTest < ActiveSupport::TestCase
  fixtures :addresses, :buildings

  def setup
    @a=Address.new
  end

  def test_correct_address
    @a.telephone="049-9050231"
    @a.street="Via Belzoni 6"
    @a.city="Villafranca veronese"
    assert @a.save
  end

  def test_destroy_address_in_building_address_id_must_nil
    addresses(:address_2).delete
    !assert(buildings(:building_1).address_id)
  end
end
```

nel file `adresses.yml`

```
address_2:
  city: Vicenza
  telephone: 049-9075393
  street: via rossini 9
```

nel file `buildings.yml`

```
building_1:
  name: Torre Archimede
  address: address_2
```

Di seguito sono riportate le tabelle che riassumono l'intera campagna di test delle classi appartenenti al componente model.

Test su AcademicOrganization

Id caso di prova	Descrizione	Obiettivo	Esito
1.1	Istanza di un oggetto con attributi nulli.	Il sistema deve riconoscere l'oggetto come non valido.	✓
2.1	Il contenuto dell'attributo number dell'oggetto d'istanza è 0.	number può contenere solo valori interi compresi tra 1 e 4. Non essendo valido l'oggetto non dovrà essere salvato nel database.	✓
2.2	number contiene il valore 7.	Lo stesso del caso di prova 2.1.	✓
2.3	number contiene il valore 3.	number contiene un valore corretto e quindi non devono essere riscontrati errori di validazione su di esso.	✓
3.1	Il contenuto dell'attributo name dell'oggetto d'istanza è 12a34.	name può contenere solo caratteri alfabetici. Non essendo valido l'oggetto non dovrà essere salvato nel database.	✓
3.2	name contiene un valore valido.	Non devono essere riscontrati errori di validazione su name.	✓
4.1	Agli attributi name e number dell'oggetto da salvare nel db vengono assegnati gli stessi valori di una tupla già presente.	Non devono esistere più tuple con la stessa coppia di valori contenuti in name e number. Non essendo valido l'oggetto non dovrà essere salvato nel database.	✓

Test su Address

Id caso di prova	Descrizione	Obiettivo	Esito
5.1	Istanza di un oggetto con attributi nulli.	Il sistema deve riconoscere l'oggetto come non valido.	✓
6.1	Il contenuto dell'attributo telephone è 12345-123456.	Il prefisso ha più di quattro cifre e quindi non è valido. L'oggetto per questo motivo non deve essere salvato nel db.	✓
6.2	telephone contiene il valore 1234-12345.	Il prefisso è corretto ma il numero contiene meno di sei cifre. L'oggetto per questo motivo non deve essere salvato nel db.	✓
6.3	telephone contiene il valore 1a23-12345.	Contiene un carattere alfabetico. L'oggetto per questo motivo non deve essere salvato nel db.	✓
6.4	telephone rispetta l'espressione regolare.	Non devono essere riscontrati errori di validazione su telephone.	✓
7.1	Cancellazione di un indirizzo associato ad uno user.	Eliminato l'indirizzo, lo user associato deve avere il contenuto della chiave esterna(address_id) a nil.	✓
8.1	Cancellazione di un indirizzo associato ad un palazzo.	Eliminato l'indirizzo, il palazzo associato deve avere il contenuto della chiave esterna(address_id) a nil.	✓
9.1	Creazione e salvataggio di un indirizzo valido.	L'oggetto deve essere salvato nel database.	✓

Test su BooleanConstraint

Id caso di prova	Descrizione	Obiettivo	Esito
10.1	Istanza di un oggetto con attributi nulli.	Il sistema deve riconoscere l'oggetto come non valido.	✓
11.1	L'attributo bool dell'oggetto d'istanza contiene il valore 3.	Il valore 3 deve essere convertito a false.	✓
11.2	bool assume il valore 1	Il valore 1 deve essere convertito a true.	✓
11.3	bool assume il valore 0.	Il valore 0 deve essere convertito a false.	✓
12.1	L'attributo isHard dell'oggetto d'istanza contiene il valore 11.	isHard deve contenere solo valori interi compresi tra 0 e 10. L'oggetto per questo motivo non deve essere salvato nel db.	✓
12.2	isHard contiene il valore negativo -1.	Lo stesso del caso di prova 12.1	✓
12.3	isHard contiene il valore 1.1.	Lo stesso del caso di prova 12.1	✓
12.4	isHard contiene il valore 0.	Non devono essere riscontrati errori di validazione su isHard.	✓

Test su Building

Id caso di prova	Descrizione	Obiettivo	Esito
13.1	Istanza di un oggetto con attributi nulli.	Il sistema deve riconoscere l'oggetto come non valido.	✓
14.1	L'attributo name dell'oggetto da salvare assume un valore uguale a quello di un oggetto già salvato.	Non devono esistere due palazzi con lo stesso nome. Quindi l'istanza del caso di prova non essendo valida non deve essere salvata nel db.	✓
15.1	Eliminazione di un oggetto di tipo Building.	Non deve più esistere nel database il palazzo, l'indirizzo associato e tutte le classi appartenenti.	✓

Test su Capability

Id caso di prova	Descrizione	Obiettivo	Esito
16.1	Istanza di un oggetto con attributi nulli.	Il sistema deve riconoscere l'oggetto come non valido.	✓
17.1	L'attributo name dell'oggetto da salvare assume un valore uguale a quello di un oggetto già salvato.	Una capability non può avere lo stesso nome di un'altra. Quindi l'istanza del caso di prova non essendo valida non deve essere salvata nel db.	✓

Test su Classroom

Id caso di prova	Descrizione	Obiettivo	Esito
18.1	Istanza di un oggetto con attributi nulli.	Il sistema deve riconoscere l'oggetto come non valido.	✓
19.1	L'attributo name dell'oggetto da salvare assume un valore uguale a quello di un oggetto già salvato. Inoltre entrambi possiedono lo stesso valore di building_id(chiave esterna per building)	In un stesso palazzo non possono esistere classi con lo stesso nome. L'istanza del caso di prova non essendo valida non deve essere salvata nel db.	✓
20.1	Eliminazione di un oggetto di tipo Classroom.	Non devono più esistere nel database la classe e tutti i vincoli associati.	✓

Test su Curriculum

Id caso di prova	Descrizione	Obiettivo	Esito
21.1	Istanza di un oggetto con attributi nulli.	Il sistema deve riconoscere l'oggetto come non valido.	✓
22.1	L'attributo name dell'oggetto da salvare assume un valore uguale a quello di un oggetto già salvato. Inoltre entrambi possiedono lo stesso valore di graduate_course_id(chiave esterna per graduate_course)	In un stesso corso di laurea non possono esistere due curricula con lo stesso nome. L'istanza del caso di prova non essendo valida non deve essere salvata nel db.	✓
23.1	Un insegnamento(oggetto di tipo Teacher) è associato a due curricula. Cancellazione di uno dei due.	Dopo l'operazione di eliminazione, l'insegnamento deve essere ancora presente nel db.	✓
23.2	Cancellazione dell'altro curriculum.	Dopo l'operazione di eliminazione l'insegnamento non deve essere più presente nel db.	✓
24.1	Eliminazione di un oggetto di tipo Curriculum associato ad un insegnamento.	Non devono più esistere nel database il curriculum e l'insegnamento associato dato che non apparteneva ad altri curricula.	✓

Test su DidacticOffice

Id caso di prova	Descrizione	Obiettivo	Esito
25.1	Eliminazione di un oggetto di tipo DidacticOffice(Segreteria didattica).	Non devono più esistere nel database la segreteria e lo user associato.	✓

Test su ExpiryDate

Id caso di prova	Descrizione	Obiettivo	Esito
26.1	Istanza di un oggetto con attributi nulli.	Il sistema deve riconoscere l'oggetto come non valido.	✓
27.1	L'attributo date dell'oggetto d'istanza contiene una data precedente alla data di sistema.	Per essere valida la data deve essere maggiore o uguale alla data di sistema. L'oggetto per questo motivo non deve essere salvato nel db.	✓
28.1	L'attributo period dell'oggetto d'istanza contiene il valore decimale 1.1	period deve contenere un intero compreso tra uno e quattro. L'oggetto per questo motivo non deve essere salvato nel db.	✓
28.2	period contiene il valore negativo -1.	Lo stesso del caso di prova 28.1.	✓
28.3	period contiene il valore 11.	Lo stesso del caso di prova 28.1.	✓
28.4	periodo contiene il valore 6.	Non devono essere riscontrati errori di validazione su period.	✓

Test su GraduateCourse

Id caso di prova	Descrizione	Obiettivo	Esito
29.1	Istanza di un oggetto con attributi nulli.	Il sistema deve riconoscere l'oggetto come non valido.	✓
30.1	L'attributo duration dell'oggetto d'istanza contiene un valore negativo pari a -1.	Per essere valido duration deve contenere un intero compreso tra uno e sei. L'oggetto per questo motivo non deve essere salvato nel db.	✓
30.2	duration contiene il valore decimale 1.1.	Lo stesso del caso di prova 30.1.	✓
30.3	duration contiene il valore 7.	Lo stesso del caso di prova 30.1.	✓
30.4	duration contiene il valore 6.	Non devono essere riscontrati errori di validazione su duration.	✓
31.1	Eliminazione di un corso di laurea(GraduateCourse) associato a due curricula. Entrambi sono associati ad uno stesso insegnamento(Teaching).	Non devono più esistere nel database il corso di laurea, i due curricula e l'insegnamento associato.	✓

Test su Period

Id caso di prova	Descrizione	Obiettivo	Esito
32.1	Istanza di un oggetto con attributi nulli.	Il sistema deve riconoscere l'oggetto come non valido.	✓
33.1	L'attributo year dell'oggetto d'istanza contiene un valore negativo pari a -1.	Per essere valido year deve contenere un intero compreso tra uno e sei. L'oggetto per questo motivo non deve essere salvato nel db.	✓
33.2	year contiene il valore decimale 1.2.	Lo stesso del caso di prova 33.1.	✓
33.3	year contiene il valore 7.	Lo stesso del caso di prova 33.1.	✓
33.4	year contiene il valore 6.	Non devono essere riscontrati errori di validazione su year.	✓
34.1	L'attributo subperiod dell'oggetto d'istanza contiene un valore negativo pari a -1.	Per essere valido subperiod deve contenere un intero compreso tra uno e quattro. L'oggetto per questo motivo non deve essere salvato nel db.	✓
34.2	subperiod contiene il valore decimale 1.2.	Lo stesso del caso di prova 34.1.	✓
34.3	subperiod contiene il valore 5.	Lo stesso del caso di prova 34.1.	✓
34.4	subperiod contiene il valore 4.	Non devono essere riscontrati errori di validazione su subperiod.	✓
35.1	A subperiod ed year vengono assegnati gli stessi valori di un oggetto già salvato.	Non possono esistere due periodi con gli stessi valori di subperiod ed year. L'istanza del caso di prova non essendo valida non deve essere salvata nel db.	✓

Test su QuantityConstraint

Id caso di prova	Descrizione	Obiettivo	Esito
36.1	Istanza di un oggetto con attributi nulli.	Il sistema deve riconoscere l'oggetto come non valido.	✓
37.1	L'attributo isHard dell'oggetto d'istanza contiene il valore 11.	isHard deve contenere solo valori interi compresi tra 0 e 10. L'oggetto per questo motivo non deve essere salvato nel db.	✓
37.2	isHard contiene il valore -1.	Lo stesso del caso di prova 12.1	✓
37.3	isHard contiene il valore 1.1.	Lo stesso del caso di prova 12.1	✓
37.4	isHard contiene il valore 0.	Non devono essere riscontrati errori di validazione su isHard.	✓
38.1	L'attributo quantity dell'oggetto d'istanza contiene il valore 1001.	quantity deve contenere solo valori interi compresi tra 1 e 1000. L'oggetto per questo motivo non deve essere salvato nel db.	✓
38.2	quantity contiene il valore -1.	Lo stesso del caso di prova 38.1	✓
38.3	quantity contiene il valore 1.1.	Lo stesso del caso di prova 38.1	✓
38.4	quantity contiene il valore 1.	Non devono essere riscontrati errori di validazione su quantity.	✓

Test su TeacherMailer

Id caso di prova	Descrizione	Obiettivo	Esito
39.1	Creazione di una mail contenente le istruzioni per l'attivazione di uno User tramite il metodo activate_teacher(sender, receiver). Sender e Receiver sono due oggetti di tipo User opportunamente creati che corrispondono al mittente e al destinatario.	L'indirizzo del destinatario della mail deve essere uguale al contenuto dell'attributo mail di receiver.	✓
39.2	Stessa descrizione per il caso di prova 39.1.	L'indirizzo del mittente della mail deve essere uguale al contenuto dell'attributo mail di sender.	✓
39.3	Stessa descrizione per il caso di prova 39.1.	Il subject della mail deve essere il seguente: Creazione account SIGEOL.	✓
39.4	Stessa descrizione per il caso di prova 39.1.	L'url necessario all'attivazione dell'account, presente nel corpo della mail, deve essere originato correttamente.	✓

Test su Teacher

Id caso di prova	Descrizione	Obiettivo	Esito
40.1	Aggiornamento di un oggetto di tipo Teacher lasciando degli attributi nulli.	Il sistema deve riconoscere l'oggetto come non valido e quindi non aggiornare la corrispondente tupla nel database.	✓
41.1	Eliminazione di un oggetto Teacher.	Non devono più esistere nel database il docente, lo user associato e tutti i vincoli e le preferenze.	✓

Test su Teaching

Id caso di prova	Descrizione	Obiettivo	Esito
42.1	Istanza di un oggetto con attributi nulli.	Il sistema deve riconoscere l'oggetto come non valido.	✓
43.1	cfu, labhours, classhours e studentsNumber contengono il valore 12.5.	Tutti e quattro devono contenere un valore numerico intero positivo. L'oggetto non è valido e non deve essere salvato nel db.	✓
44.1	cfu, labhours, classhours e studentsNumber contengono il valore -12.	Lo stesso del caso di prova 43.1.	✓
45.1	cfu assume il valore 21, labhours e classhours il valore 51 e studentsNumber il valore 1001.	Tutti e quattro gli attributi superano di un'unità le soglie imposte e quindi deve essere riscontrato un errore di validazione su ognuno di essi.	✓
46.1	Creazione di un insegnamento associato ad un periodo non compatibile con l'organizzazione accademica del corso di laurea dell'insegnamento.	L'oggetto d'istanza non essendo valido non deve essere salvato nel database.	✓

Test su TemporalConstraint

Id caso di prova	Descrizione	Obiettivo	Esito
47.1	Istanza di un oggetto con attributi nulli.	Il sistema deve riconoscere l'oggetto come non valido.	✓
48.1	L'attributo day dell'oggetto d'istanza contiene il valore 6.	day deve contenere solo valori interi compresi tra 1 e 5. L'oggetto per questo motivo non deve essere salvato nel db.	✓
48.2	day contiene il valore 0.	Lo stesso del caso di prova 48.1.	✓
48.3	day contiene il valore 1.5.	Lo stesso del caso di prova 48.1.	✓
48.4	day contiene il valore 1.	Non deve essere riscontrato nessun errore di validazione su day.	✓
49.1	L'attributo isHard dell'oggetto d'istanza contiene il valore 11.	isHard deve contenere solo valori interi compresi tra 0 e 10. L'oggetto per questo motivo non deve essere salvato nel db.	✓
49.2	isHard contiene il valore -1.	Lo stesso del caso di prova 49.1	✓
49.3	isHard contiene il valore 1.1.	Lo stesso del caso di prova 49.1	✓
49.4	isHard contiene il valore 0.	Non devono essere riscontrati errori di validazione su isHard.	✓
50.1	Il valore contenuto in startHour assume un valore più grande di quello in endHour	Ovviamente l'ora d'inizio(startHour) dell'indisponibilità non può essere più grande dell'ora fine(endHour). Quindi non essendo valido l'oggetto non deve essere salvato nel db.	✓

Test su TimetableEntry

Id caso di prova	Descrizione	Obiettivo	Esito
51.1	Istanza di un oggetto con attributi nulli.	Il sistema deve riconoscere l'oggetto come non valido.	✓
52.1	L'attributo day dell'oggetto d'istanza contiene il valore 6.	day deve contenere solo valori interi compresi tra 1 e 5. L'oggetto per questo motivo non deve essere salvato nel db.	✓
52.2	day contiene il valore 0.	Lo stesso del caso di prova 52.1.	✓
52.3	day contiene il valore 1.5.	Lo stesso del caso di prova 52.1.	✓
52.4	day contiene il valore 1.	Non deve essere riscontrato nessun errore di validazione su day.	✓
53.1	Il valore contenuto in startTime assume un valore più grande di quello in endTime	Ovviamente l'ora d'inizio(startTime) di un riga dello schema d'orario non può essere più grande dell'ora fine(endHour). Quindi non essendo valido l'oggetto non deve essere salvato nel db.	✓
54.1	Creazione di un oggetto TimetableEntry con gli stessi valori di day, StartTime, EndTime, classroom_id e timetable_id di un oggetto già salvato.	Per evitare sovrapposizioni, in uno stesso schema d'orario identificato da timetable_id non possono esistere righe con ugual giorno, ora di inizio, ora di fine e classe occupata. L'istanza appena creata non essendo valida non deve essere salvata nel db.	✓

Test su Timetable

Id caso di prova	Descrizione	Obiettivo	Esito
55.1	Istanza di un oggetto con attributi nulli.	Il sistema deve riconoscere l'oggetto come non valido.	✓
56.1	L'attributo year dell'oggetto d'istanza contiene il valore 20000-20.	il contenuto di year non è valido. L'oggetto per questo motivo non deve essere salvato nel db.	✓
56.2	year contiene 2007-08.	year, per essere valido deve assumere nei primi quattro caratteri(2007) un valore successivo all'anno di sistema-1. L'anno di sistema nel caso di prova corrisponde a 2009. Non essendo valido, l'oggetto non deve essere salvato nel db	✓
56.3	year contiene 2008-10.	year per essere valido doveva avere nelle ultime due cifre i caratteri 09. Non essendo valido, l'oggetto non deve essere salvato nel db	✓
56.4	year contiene 2008-09.	Non deve essere riscontrato nessun errore di validazione su year.	✓
57.1	Creazione di un oggetto Timetable con gli stessi valori di year, graduate_course_id(chiave esterna per il corso di laurea) e period_id(chiave esterna per il periodo) di un oggetto già salvato.	Non possono esistere per uno stesso corso di laurea, periodo e anno più schemi d'orario.	✓

Test su User

Id caso di prova	Descrizione	Obiettivo	Esito
58.1	Istanza di un oggetto con attributi nulli.	Il sistema deve riconoscere l'oggetto come non valido.	✓
59.1	L'attributo password dell'oggetto d'istanza è vuoto.	password deve essere di almeno sei caratteri di tipo alfanumerico più il carattere'.'. L'oggetto per questo motivo non deve essere salvato nel db.	✓
59.2	password contiene la stringa prova.	prova contiene meno di sei caratteri. Non essendo valido, l'oggetto non deve essere salvato nel db	✓
59.3	password contiene pro.va.	Non deve essere riscontrato nessun errore di validazione su password.	✓
60.1	L'attributo mail dell'oggetto d'istanza contiene la stringa prova?id=1@math.unipd.it	mail contiene al suo interno un carattere non valido:?. Non essendo valido, l'oggetto non deve essere salvato nel db.	✓
60.2	L'attributo mail contiene la stringa agrossel@math.unipd.it.	Non deve essere riscontrato nessun errore di validazione su mail.	✓
61.1	Creazione di uno user con lo stesso valore dell'attributo mail di un altro oggetto già salvato.	Non possono esistere due user con la stessa mail; quindi l'oggetto appena istanziato non è valido e non deve essere salvato.	✓
62.1	Creazione di uno user con password uguale ad alessandro.	L'oggetto istanziato viene correttamente salvato nel database. Il contenuto di password deve essere uguale alla stringa alessandro crittografata con algoritmo SHA1.	✓
63.1	Creazione di uno user senza specificarlo.	Un oggetto di tipo user deve appartenere o ad una segreteria o ad un insegnante; se non è associato a nulla l'istanza non è valida e non deve essere salvata nel db.	✓

Id caso di prova	Descrizione	Obiettivo	Esito
64.1	Al metodo authenticate vengono passati uno user ed una password non corretti.	Il metodo deve ritornare un valore booleano pari a false.	✓
64.2	Ora vengono passati uno user ed una password corretti.	Il metodo deve ritornare un valore booleano pari a true.	✓
65.1	Eliminazione di un oggetto User.	Non devono più esistere nel database lo user, l'indirizzo e il docente o la segreteria associata.	✓
66.1	Creazione di uno user associato a tutte le possibili capabilities(privilegi).	I metodi che iniziano con manage_ di un oggetto User ritornano true se l'istanza possiede quel determinato privilegio. In questo caso di prova ogni metodo deve ritornare true.	✓
67.1	Creazione di uno user associato ad una segreteria didattica.	Lo user appartiene ad una segreteria quindi il metodo own_by_didactic_office? deve ritornare true e own_by_teacher? false.	✓

Test su Belong

Id caso di prova	Descrizione	Obiettivo	Esito
69.1	Creazione di un oggetto di tipo Belong che associa un curriculum ad un insegnamento già associati.	Esistendo già l'associazione, l'istanza non deve essere salvata.	✓

5.2.2 Functional Tests

Nello sviluppo di un'applicazione tramite il framework Rails, i test funzionali (functional tests) sono specifici per la verifica degli elementi appartenenti alla componente Controller. Dato che gli unit tests sono stati effettuati tramite istanze reali, il team QuiXoft ha scelto di utilizzare la strategia dei Mock objects per la verifica delle azioni presenti nei controller.

Ogni classe che implementa un insieme di test per un particolare controller dovrà essere denominata `NomeControllerTest` ed essere salvata su di un file chiamato `nome_controller_controller_test.rb` all'interno della directory `test/functional`.

Un esempio di functional test per il controller `sessions` è dato dalla seguente porzione di codice:
nel file `sessions_controller_test.rb`

```
class SessionsControllerTest < ActionController::TestCase

  test "Guest usa New" do
    get :new
    assert_template "new"
    assert_response :success
  end

  test "Immissione di email e password validi" do
    user = stub(:id => :an_id, :mail => "a_mail",
                :password => "a_password")
    User.stubs(:authenticate).returns(user)
    user.stubs(:active?).returns(true)
    post :create, :mail => user.mail, :password => user.password
    assert_equal session[:user_id], :an_id
    assert_redirected_to timetables_url
  end
end
```

5.2.3 Copertura dei test sul codice

Come descritto precedentemente sarà utilizzato uno strumento per il linguaggio di programmazione Ruby per misurare la copertura dei test sul codice. Il team QuiXoft utilizzerà la metrica C0, ovvero sarà controllata la copertura di ogni istruzione. L'attuale situazione di coprtura dei test sul codice è riportato nel file `coverage.pdf` allegato al presente documento.

Come si può notare la percentuale per ora coperta è appena inferiore al 50%, il che è comprensibile dato lo sviluppo tramite TDD il quale testa solamente le i rami `true` delle istruzioni condizionali. Per la consegna del prodotto la percentuale dovrà essere del 100%

5.3 Dettaglio dell'esito delle revisioni

Per ciascuna revisione affrontata sarà qui riportato un resoconto dei risultati e dei problemi emersi. Tali dati dovranno essere usati come base per migliorare la qualità del progetto in visione della revisione successiva, con l'obiettivo finale di offrire un prodotto che soddisfi tutti i requisiti e che calzi con le esigenze del committente.

5.3.1 Revisione dei Requisiti

In data 7 Gennaio 2009 il team QuiXoft ha sostenuto la prima delle 4 revisioni che dovrà affrontare durante lo svolgimento del progetto: la Revisione dei Requisiti. L'offerta è stata accettata dal Committente, il quale ha ritenuto valide le idee e le proposte del team per lo sviluppo del progetto proposto.

Sono però sorti alcuni problemi e perplessità, che dovranno essere sistemati entro la prossima revisione. Segue un elenco dei documenti consegnati e dei relativi problemi riscontrati:

- Norme di Progetto: auspicabile per l'immediato futuro una maggiore attenzione alle infrastrutture di supporto alla gestione di progetto e, soprattutto, all'automazione delle attività di qualifica.
- Analisi dei Requisiti: è necessario far precedere i diagrammi dei casi d'uso all'elencazione dei requisiti. Sono assenti l'elencazione con classificazione dei requisiti individuati e il tracciamento requisiti-casi d'uso. E' consigliata l'eliminazione della Segreteria Generale e di tutti i riferimenti ad essa, per non complicare ulteriormente il progetto. Va rivista la procedura di creazione degli account di Presidente del CCS e della segreteria didattica tramite inviti: consigliata la creazione di account ad hoc in fase di installazione del sistema. La procedura degli inviti può comunque considerarsi valida per gli account dei docenti.
- Studio di Fattibilità: Non appare alcuna minima considerazione fatta sugli altri capitolati.
- Piano di Progetto: Prestare attenzione all'eccessivo e fastidioso livello di ripetizione di testo tratto da documentazione esterna: meglio riferire che copiare. E' presente qualche sintomo di insufficiente sforzo o qualità di verifica. Non pare poi particolarmente previdente assumere che l'Amministratore debba essere avvisato della rilevata insufficienza di risorse di verifica: provvedere a trovare una soluzione alternativa.
- Piano di Qualifica: insufficiente l'evidenza di dotazione di infrastrutture di supporto alla realizzazione del Piano di Qualifica e di procedure e norme che ne accompagnino l'esecuzione.
- Glossario: può essere allegerito, eliminando i termini scontati più comuni.

In fase di revisione sono inoltre stati segnalati i seguenti problemi, non legati ad un singolo documento ma comuni a tutta la consegna:

- Si suggerisce l'uso di indici ipertestuali nei documenti.

- Inserire nel nome dei file anche il numero di revisione del documento: risulta cosi distinguibile una versione dall'altra dello stesso documento.
- Si auspica un maggior controllo di quanto spedito via e-mail prima della revisione: erano infatti presenti all'interno dell'archivio decine di file di sistema, numerose cartelle inutili e altri file vuoti, di disturbo per il ricevente. Per la prossima revisione si dovrà prestare attenzione ad inviare solo i file essenziali in formato PDF, strutturati in cartelle utili alla lettura e comprensione del progetto.

5.3.2 Revisione di Progetto Preliminare

In data 29 Gennaio 2009 il team QuiXoft ha sostenuto la seconda delle revisioni previste: la Revisione di Progetto Preliminare. Il Docente di riferimento presente in sede di revisione ha ritenuto valide le scelte progettuali adottate dal team QuiXoft.

Sono tuttavia state rilevate le seguenti problematiche nella documentazione consegnata, che dovranno essere sistemate entro la successiva revisione:

- Norme di Progetto: per non appesantire troppo il documento, i termini in glossario possono essere indicati anche solo la prima volta che vengono incontrati nel documento. Tra le norme non ne è presente alcuna che riguarda la nomenclatura dei documenti. Contrariamente a quanto deciso in precedenza, è sconsigliato aggiornare i software usati: meglio mantenere per la durata del progetto la stessa versione di strumento, onde evitare invalidazioni del lavoro fatto. E' necessario dotarsi di un sistema di Code Convention al più presto. E' consigliata la scelta di un singolo DBMS da usare per lo sviluppo e per il test del progetto SIGEOL. Nonostante il framework Ruby on Rails consigli agli sviluppatori l'uso di svariati DBMS per testare la bontà e la portabilità del progetto, tale variabilità potrebbe influire negativamente sul funzionamento del prodotto finale e su tutte le fasi di test.
- Analisi dei Requisiti: porre maggiore attenzione alla consegna del documento, in quanto è stata consegnata una versione sbagliata compilata con le immagini errate. Fare attenzione anche alla versione del documento in caso di problemi simili. Sarebbe meglio numerare ed etichettare le figure. Non risulta chiaro, nel caso d'uso generale, se il SIGEOL DB è interno o esterno al sistema. Tabella di tracciamento Casi d'uso - requisiti: poiché vi sono dei requisiti comuni a più casi d'uso, è necessario inserire anche la tabella per il tracciamento inverso.
- Specifica Tecnica: il sommario del documento è un po' troppo generico. Il design pattern MVC presentato è errato perché la parte di View deve avere riferimento al Model o al View Helper. Valutare se

anche Middle Man e Algorithm potrebbero far parte del Model. Dare una didascalia e un numero alle figure, in modo da poterle riferire nel testo e rendere più facile la lettura. Conviene spostare il tracciamento requisiti-componente alla fine del documento, in modo da non spezzare il filo conduttore che "Top-Down" che governa la descrizione dell'architettura. In questa tabella ci sono requisiti che sono implementati da tutte le componenti: forse è necessario adottare una grana più fine sui requisiti, altrimenti sarà veramente difficile la verifica della loro soddisfazione da parte del prodotto. Il documento è ritenuto incompleto: è necessario integrarlo con le parti mancanti e scendere un po' più in profondità nelle componenti.

- Piano di Qualifica: C'è ancora ampio spazio per consistenti miglioramenti, sia in termini di copertura delle fasi di progetto che nella formalizzazione dei flussi di tali attività (workflow).

- Piano di Progetto: qualche miglioramento nella direzione auspicata, ma complessivamente ancora incompleto e di insufficiente profondità. Con l'avanzare del progetto, il consuntivo prevale per valore informativo sul preventivo nelle fase già svolte: per dimostrare di avere pieno controllo della situazione rispetto alle variazioni avvenute, converrà rafforzare l'analisi delle implicazioni del consuntivo, con un opportuno preventivo a finire, che sostituirà ove necessario il preventivo iniziale. Per quanto concerne l'analisi dei rischi, converrà identificare indicatori utili a rilevare livelli di rischio significativi; ai livelli di allarme converrà anche associare opportune strategie di mitigazione. L'aumento del costo di progetto a fronte del vincolo contrattuale assunto in sede di RR, pur legato all'ingresso di un nuovo componente del gruppo, va giustificato formalmente in termini di valore aggiunto in termini di qualità.

Diario delle modifiche

Data	Versione	Modifica
06-03-2009	3.0.0	Approvazione del responsabile
06-03-2009	2.3.1	Verifica e correzione della versione 2.3.0
06-03-2009	2.3.0	Aggiunto il dettaglio delle verifiche tramite test
18-02-2009	2.2.0	Aggiunti i plugin utilizzati per l'automazione delle attività di verifica
10-02-2009	2.1.0	Aggiunto il resoconto della Revisione di Progetto Preliminare
24-01-2009	2.0.0	Approvazione del Responsabile
21-01-2009	1.2.1	Correzioni generali di forma e sintassi
21-01-2009	1.2.0	Aggiunta la sezione Resoconto delle attività di verifica
20-01-2009	1.1.0	Aggiunta la sezione Automazione delle attività di verifica
09-12-2008	1.0.0	Approvazione del Responsabile e passaggio di stato in "formale"
08-12-2008	0.5.1	Correzioni varie
07-12-2008	0.5.0	Completamento stesura della sottosezione Tecniche e metodi di verifica
06-12-2008	0.4.1	Varie correzioni grammaticali
05-12-2008	0.4.0	Aggiunta delle sezione Gestione Amministrativa
05-12-2008	0.3.0	Stesura della sottosezione Tecniche e metodi di verifica
04-12-2008	0.2.0	Aggiunta delle sezione Visione Generale
03-12-2008	0.1.0	Prima stesura dell'indice del documento