2 marzo 2015

Piano di Qualifica

Informazioni sul documento

Nome Documento	Piano di Qualifica
Versione	1.0
Stato	*Formale*
Uso	*Interno*
Data Creazione	2 marzo 2015
Data Ultima Modifica	2 marzo 2015
Redazione	
Approvazione	
Verifica	
Lista distribuzione	

Prof. Tullio Vardanega
Prof. Riccardo Cardin
Proponente Zucchetti S.p.a.

Registro delle modifiche

Versione	Autore	Data	Descrizione
1.0			Prima stesura del documento.

Tabella 1: Versionamento del documento

info@gmail.com

Storico

pre-RR

Versione 1.0	Nominativo
Redazione	
Verifica	
Approvazione	

Tabella 2: Storico ruoli pre-RR

info@gmail.com

Indice

Sommario

Il presente documento contiene le norme e le convenzioni che il gruppo intende adottare durante l'intero ciclo di vita del prodotto software Premi.

info@gmail.com

1 Introduzione

1.1 Scopo del documento

Il Piano di Qualifica ha lo scopo di descrivere le strategie che il gruppo di lavoro ha deciso di adottare per perseguire obiettivi qualitativi da applicare al proprio prodotto. Per ottenere tali obiettivi è necessario un processo di verifica continua sulle attività svolte; questo consentirà di rilevare e correggere anomalie e incongruenze in modo tempestivo e senza spreco di risorse.

1.2 Scopo del Prodotto

Lo scopo del progetto è la realizzazione un software per la creazione ed esecuzione di presentazioni multimediali favorendo l'uso di tecniche di storytelling e visualizzazione non lineare dei contenuti.

1.3 Glossario

Al fine di evitare ogni ambiguità di linguaggio e massimizzare la comprensione dei documenti, i termini tecnici, di dominio, gli acronimi e le parole che necessitano di essere chiarite, sono riportate nel documento (((Glossario v?.?.?))). Ogni occorrenza di vocaboli presenti nel Glossario è marcata da una "G" maiuscola in pedice.

1.4 Riferimenti

1.4.1 Normativi

- Norme di progetto: (((Norme di Progetto v?.?.?)));
- Capitolato d'appalto C4: Premi: Software di presentazione "better than Prezi" http://www.math.unipd.it/~tullio/IS-1/2014/Progetto/C4.pdf.

1.4.2 Informativi

- Piano di progetto: (((Piano di progetto v?.?.?)));
- Slide dell'insegnamento Ingegneria del Software modulo A: http://www.math.unipd.it/~tullio/IS-1/2014/ ;
- SWEBOK – Version 3 (2004): capitolo 11 – Software Quality http://www.computer.org/portal/web/swebok/html/ch11;
- Ingegneria del software - Ian Sommerville - 9a Edizione (2010):
  - Capitolo 24 - Gestione della qualità;
  - Capitolo 26 - Miglioramento dei processi.
- Standard ISO /IEC TR 15504: Software process assessment http://en.wikipedia.org/wiki/ISO/IEC_15504;

- Standard ISO /IEC 9126: Product quality
  http://en.wikipedia.org/wiki/ISO/IEC_9126;

1.5 Strumenti,tecniche e metodi

1.5.1 Strumenti

Di seguito verranno elencati gli strumenti software che sono o saranno utilizzati dal uppo per effettuare le operazioni di verifica e validazione:

- **Aspell**: correttore ortografico per documenti redatti in LaTeX;
- **Aptana**: scelto dal gruppo per la stesura del codice ha integrato al suo interno varie funzioni di debugging ed esecuzione del codice;
- **W3C validator**: Sito che controlla la validità dei markup nei documenti scritti in HTML;
- **Jenkins** : sistema di per l'integrazione continua del codice e dei file latex;

  Verranno utilizzati anche tutti gli strumenti già integrati all'interno del browser per lo sviluppo delle pagine web.

1.5.2 Tecniche

- **Analisi statica**: consiste nell'analizzare il codice tramite tools e letture senza tuttavia eseguirlo. Data la natura di questo tipo di analisi, è possibile applicarla anche per il controllo di tutti i documenti testuali prodotti. Si esegue applicando i due seguenti metodi:
  - **Inspection**: l'obiettivo di questa tecnica di analisi è l'individuazione di difetti attraverso la lettura mirata del codice. Un prerequisito per questa metodologia di verifica è la definizione di una lista di controllo che elenca le possibili sezioni o passaggi maggiormente soggetti ad errori. verifica deve essere condotta da soggetti nettamente distinti dai programmatori. La correzione degli errori individuati va eseguita in ogni fase e documentata tramite un rapporto delle attività svolte;
  - **Walkthrough**: l'obiettivo di questa tecnica di analisi è l'individuazione di difetti eseguendo una lettura integrale di tutto il codice senza l'assunzione di presupposti. Viene eseguita da gruppi misti di ispettori e sviluppatori. Per evitare incomprensioni è importante che al termine della lettura gli elementi coinvolti discutano i difetti trovati e che nessuna persona possa coprire entrambi i ruoli allo stesso tempo. Al termine della fase di discussione si applicherà la fase di correzione dei difetti che apporterà le modifiche concordate. Anche in questo caso è importante tenere un rapporto delle attività svolte.
- **Analisi dinamica**: consiste nel verificare e validare il software o un suo componente osservandone il comportamento in esecuzione durante lo svolgimento di test. Tali test devono essere svolti in maniera ripetibile ossia devono venir eseguiti nello stesso ambiente e con gli stessi ingressi in modo da potersi aspettare i medesimi risultati.
  - **Test di unità**: esamina la correttezza di piccole unità di codice, generalmente prodotte da un singolo programmatore, in modo da verificare che esse rispettino i loro requisiti. E' possibile svolgerlo con un alto grado di parallelismo possibilmente servendosi di un automa.

- **Test di integrazione**: verifica che l'integrazione delle unità che hanno superato il test precedente non produca problemi. Tali problemi, non potendo essere relativi alle singole unità, saranno da ricercare nell'interfaccia che le aggrega.
- **Test di sistema**: accerta la copertura dei requisiti software individuati nell'analisi dei requisiti permettendo la validazione del sistema prodotto
- **Test di regressione**: stabilisce se modifiche all'implementazione di un programma alterano elementi precedentemente funzionanti. Per far ciò si eseguono nuovamente i test di unità e integrazione sulle parti modificate.
- **Test funzionali**: Mettono alla prova le funzionalità del sistema ,simulando l'iterazione tra Utente e sistema.
- **Test Prestazionali**: Valuta le prestazioni dell'applicazione in molti modi e da molti punti di vista. Questo tipo di test mostra ciò che proverà l'utente in termini di caricamento e velocità del sito. Le prestazioni sono importanti anche epr motivi SEO , in quanto un sito lento verrà analizzato molto meno frequentemente dai search engine crawlers dei motori di ricerca ;
- **Test di collaudo**: attività formale supervisionata dal committente il cui buon esito comporta la possibilità di rilasciare il prodotto.

1.5.3 Metodi

Il gruppo ha deciso di utilizzare i seguenti metodi per applicare le tecniche sopra descritte, aiutandosi con gli strumenti elencati:

- **Documenti LaTeX**:
  1. rilettura approfondita;
  2. controllo ortografico tramite lo strumento Aspell;
  3. controllo dell'applicazione delle regole tipografiche esposte nel documento Norme di progetto;
  4. verifica della corretta formattazione del file pdf prodotto.
- **Codice**:

  il codice verrà analizzato dagli strumenti integrati all'interno dell' IDE Aptana e dagli strumenti di sviluppo forniti dai singoli browser .
- **Schemi UML**:
  1. data l'impossibilità di controllare la correttezza ortografica degli schemi; con Aspell è necessario esaminare attentamente e più volte i nomi, gli identificativi e i testi nei diagrammi;
  2. controllo della correttezza degli identificativi dei casi d'uso rispetto alla nomenclatura stabilita nel documento Norme di progetto e rispetto alle sezioni dell'Analisi dei requisiti in cui sono inseriti;
  3. controllo della numerazione dei casi d'uso rispetto la loro gerarchia;
  4. controllo che i casi d'uso soddisfino tutte le esigenze espresse nel capitolato.

1.5.4 Metriche

Una metrica è una misura di una qualche proprietà relativa ad una porzione di software , allo scopo di fornire informazioni significative sulla qualità del codice prodotto. Non bisogna tuttavia basarsi solamente sulle metriche, che sono solamente indicatori a posteriori della bontà del lavoro svolto: un'importanza ancora maggiore la riveste il controllo sulla qualità del processo.

- **Complessità ciclomatica di McCabe**: è un'indicazione del numero di segmenti lineari in un metodo (ad esempio sezioni di codice senza ramificazioni) e quindi può essere usato per determinare il numero di test necessari per ottenere una copertura completa dei possibili cammini. Un metodo senza ramificazioni ha Complessità Ciclomatica pari a 1. Tale valore è incrementato ogni qualvolta si incontra una ramificazione. Con "ramificazione" si intendono i cicli e i costrutti "if" e simili.

- **Numero di istruzioni**: numero di istruzioni all'interno di un metodo. Un indice elevato non rappresenta necessariamento un cattivo codice ma suggerisce la possibilità di estrarre metodi contenenti gruppi di istruzioni correlate, aumentando il livello di astrazione