SiVoDiM

Piano di Qualifica

Versione	1.0.0
Redattori	<Redattore>
Verificatori	<Verificatore>
Responsabili	<Responsabile>
Uso	<Uso>
Lista di distribuzione	*Stark Labs* Prof. Vardanega Tullio, Dr. Cardin Riccardo

Documento riguardante l'insieme di strategie di verifica adottate dal gruppo StarkLabs per il conseguimento di requisiti qualitativi per il progetto SiVoDiM.

Registro delle modifiche

Attività	Autori	Data	Versione
Revisione	Alberto Andriolo, Enrico Chiara	15/03/2016	v0.3.0
Stesura del documento	Federico Rossetto	11/03/2016	v0.0.1

Indice

Elenco delle tabelle

Elenco delle figure

1 Introduzione

1.1 Scopo del documento

Questo documento ha lo scopo di definire le strategie che il gruppo di lavoro ha deciso di adottare per perseguire obiettivi qualitativi da applicare al proprio progetto. A tale scopo è necessario un continuo processo di verifica, in modo da correggere eventuali anomalie o incongruenze sulle attività svolte in maniera tempestiva e senza spreco di risorse.

1.1.1 Scopo del Prodotto

L'obiettivo del progetto è di sperimentare e rendere disponibili su dispositivi mobili nuove funzionalità di sintesi vocale (TTS$_G$), come la possibilità di applicare effetti alle voci digitali o sintetizzare e utilizzare la voce degli utenti. Ciò sarà possibile realizzando due applicazioni per i sistemi Android$_G$.

- La prima deve permettere all'utente di interfacciarsi direttamente con il sistema operativo per configurare, salvare, modificare nuove voci campionate;
- La seconda invece permette la creazione, il salvataggio e la condivisione di veri e propri sceneggiati.

Entrambe le applicazioni devono sfruttare altre due componenti, realizzate sempre all'interno del progetto:

- Un Engine$_G$ che permetta di interfacciarsi tramite una connessione al motore di sintesi prodotto dal proponente$_G$;
- Una libreria di funzionalità.

1.2 Glossario

Al fine di aumentare la comprensione del testo ed evitare eventuali ambiguità, viene fornito un glossario (*Glossario v1.0.0*) contenente le definizioni degli acronimi e dei termini tecnici utilizzati nel documento. Ogni vocabolo contenuto nel glossario è contrassegnato dal pedice "$_G$".

1.3 Riferimenti

2 Visione generale della strategia di verifica

2.1 Definizione obiettivi

2.1.1 Qualità di processo

Per garantire la qualità del prodotto è necessario perseguire la qualità dei processi che lo definiscono. Per questo motivo si è deciso di utilizzare lo standard ISO$_G$/IEC$_G$15504 denominato SPICE, che fornisce gli strumenti necessari per valutarne l'idoneità.
Al fine di applicare correttamente questo modello si deve utilizzare il ciclo di *Deming*$_G$, il quale definisce una metodologia di controllo per i processi durante il loro ciclo di vita, per migliorarne costantemente la qualità.

2.1.2 Qualità di prodotto

Per aumentare il valore commerciale di un prodotto software, e per garantire il corretto funzionamento dello stesso, è necessario fissare degli obiettivi qualitativi e verificare che questi vengano rispettati.
Lo standard ISO$_G$/IEC$_G$9126 è stato redatto allo scopo di definire questi obiettivi e delineare alcune metriche capaci di misurare il raggiungimento degli stessi.

2.2 Procedure di controllo di qualità di processo

La qualità dei processi è garantita dall'applicazione del principio PDCA$_G$. Grazie a questo processo si può garantire un continuo miglioramento della qualità dei processi, inclusa la verifica. Questo comporta quindi un miglioramento dei prodotti creati.

Per avere il controllo dei processi, e quindi della qualità, è necessario che:

- I processi siano pianificati dettagliatamente;
- Vengano ripartite chiaramente le risorse nella pianificazione;
- Ci sia controllo sui processi.

L'attuazione di tali punti è descritta dettagliatamente nel *Piano di Progetto v1.0.0*.
La qualità dei processi viene inoltre controllata tramite l'analisi costante della qualità del prodotto. Un processo da migliorare è indicato da un prodotto di bassa qualità.
Per quantificare la qualità dei processi verranno utilizzate le metriche descritte nella sezione **(((Aggiungi sezione)))**.

2.3 Procedure di controllo di qualità di prodotto

Il controllo di qualità dei prodotti viene garantito da:

- **Quality assurance**: è l'insieme di attività realizzate al fine di garantire il raggiungimento degli obiettivi di qualità. Prevede l'attuazione di tecniche di analisi statica e dinamica, descritte nella **(((Aggiungi sezione)))**;
- **Verifica**: processo che determina se il risultato di una fase è corretto. La verifica viene eseguita costantemente durante tutta la durata del progetto. I risultati delle attività di verifica eseguiti nelle varie fasi di progetto riportate **(((Aggiungi sezione)))**;
- **Validazione**: la conferma oggettiva che il sistema risponde ai requisiti.

2.4 Organizzazione

L'organizzazione della strategia di verifica è basata sull'utilizzo di attività di controllo per ogni processo attuato. Per ognuno di questi viene verificata la qualità, ed eventualmente la qualità del prodotto ottenuto. Ognuna delle fasi del progetto descritte nel *Piano di Progetto v1.0.0* necessita di diverse attività di verifica a causa dei differenti output:

- **Analisi:** in questa fase è necessario seguire i metodi di verifica descritti nelle sezioni **(((Aggiungi sezione)))** sui documenti prodotti. La realizzazione di tali attività di verifica sono descritte nella **(((Aggiungi sezione)))** 4.1.1.

In ogni documento viene inoltre incluso il diario delle modifiche che permette di mantenere uno storico delle attività svolte e delle relative responsabilità.

2.5 Pianificazione strategica e temporale

Dato che l'obiettivo è di rispettare le scadenze fissate nel *Piano di Progetto v1.0.0*, è necessario che l'attività di verifica sia ben organizzata. Quindi l'individuazione e la correzione di errori dovrà essere tempestiva, in modo da impedire che si diffondano.
Ogni attività di redazione di documenti o di codifica deve essere preceduta da un analisi della struttura e dei contenuti. Questo allo scopo di evitare imprecisioni concettuali o tecniche, rendendo l'attività di verifica più semplice, richiedendo minori correzioni. La metodologia da seguire per individuare e correggere eventuali errori è descritta nelle *Norme di Progetto v1.0.0*.

2.6 Responsabilità

Per garantire che il processo di verifica sia efficace e sistematico, vengono attribuite le responsabilità al Responsabile di Progetto ed ai Verificatori. La suddivisione dei compiti e le modalità sono definite nelle *Norme di Progetto v1.0.0*.

2.7 Risorse

Per assicurarsi che gli obiettivi vengano raggiunti sono necessarie delle risorse sia umane che tecnologiche. Coloro che detengono la responsabilità maggiore per le attività di verifica e validazione sono il Responsabile di Progetto e il Verificatore. I ruoli sono descritti nel dettaglio nelle Norme di Progetto.
Per risorse tecniche e tecnologiche sono intesi tutti gli strumenti *software* e *hardware* che il gruppo intende utilizzare. Affinché il lavoro dei verificatori venga semplificato, sono stati impostati alcuni strumenti di controllo sistematico. Questi sono descritti in modo accurato nelle *Norme di Progetto v1.0.0*.

2.8 Tecniche di analisi

2.8.1 Analisi statica

Per analisi statica si intende una tecnica di controllo che permette di effettuare la verifica di quanto prodotto individuando errori. Essa viene svolta in due modi complementari.

2.8.1.1 Walkthrough Viene svolta una lettura critica di tutto il materiale. Questa tecnica è utile nelle prime fasi di progetto, quando i membri del gruppo non hanno ancora una adeguata esperienza che permette verifiche più mirate.
Grazie a questa tecnica, il Verificatore può stilare una lista di errori più frequenti, in modo da migliorare le attività future.
Questa attività è onerosa e richiede l'intervento di più persone per essere efficace ed efficiente. In seguito alla lettura segue una fase di discussione con il fine di esaminare i difetti e proporre le correzioni. La fase finale consiste nello stilare un rapporto che elenchi le modifiche effettuate.

2.8.1.2 Inspection In questa tecnica viene eseguita un'analisi mirata delle parti del documento o del codice che sono ritenute maggiormente fonte di errore. La lista di controllo, che contiene queste sezioni critiche, è redatta anticipatamente, ed è frutto dell'esperienza dei verificatori grazie alla tecnica precedente.
Questa strategia è più rapida del Walkthrough in quanto riduce il numero di parti da analizzare. Questo comporta che la tecnica possa essere eseguita solamente dai verificatori, che individuano e correggono eventuali errori e redigono il rapporto di verifica per tracciare il lavoro svolto.

2.8.2 Analisi dinamica

L'analisi dinamica viene applicata solamente alla produzione di codice e viene svolta durante l'esecuzione mediante l'uso di test utilizzati per verificarne il funzionamento.
Per rendere questa attività utile e generare risultati attendibili, è necessario che i test siano ripetibili. Questo significa che il programma, da un determinato input, genera sempre lo stesso output. Test di questo tipo sono utili per determinare la correttezza ed evidenziare eventuali problemi in un *software*. Devono quindi essere definiti:

- **Ambiente:** consiste sia del sistema hardware che di quello software sui quali è pianificato lo sviluppo. Di questi è necessario specificare lo stato iniziale dal quale iniziare i test;
- **Specifica:** consiste nel definire gli input e i rispettivi output che sono attesi;
- **Procedure:** ossia la definizione di come i test vengono svolti, con quale ordine e come vengono analizzati i risultati.

Esistono 5 tipi diversi di test: test di unità, test di integrazione, test di sistema, test di regressione e test di accettazione.

2.8.2.1 Test di unità Consta nel verificare ogni singola unità del software con l'utilizzo di $stub_G$, $driver_G$ e $logger_G$. Con unità è inteso il minimo quantitativo di software che sia utile verificare singolarmente, prodotto da un singolo programmatore. Con questi test si verifica il funzionamento corretto dei moduli. Questo per eliminare dal sistema possibili errori di implementazione.

2.8.2.2 Test di integrazione Consta nel verificare i componenti del sistema che vengono aggiunti. Questo per analizzare che la combinazione di due o più unità software funzionino come previsto. Il fine di questo tipo di test è quello di individuare errori residui dalla realizzazione dei moduli, o comportamenti inaspettati da componenti software forniti da terze parti. Per effettuare questi test è necessario aggiungere delle componenti fittizie per sostituire quelle ancora non sviluppate, al fine di non influenzare l'esito dell'analisi.

2.8.2.3 Test di sistema Consta nel validare il prodotto software nel momento in cui si ritiene che sia giunto ad una versione definitiva. Questo test verifica che tutti i requisiti software stabiliti nell'*Analisi dei Requisiti v1.0.0* vengano rispettati.

2.8.2.4 Test di regressione Consta nell'eseguire nuovamente i test sulle componenti software, quando subiscono modifiche. Questo per controllare che i cambiamenti non alterino il corretto funzionamento di queste componenti, o di altre che non sono state aggiornate. Questa operazione viene aiutata dal tracciamento, che permette di individuare e ripetere i test di unità, integrazione e di sistema, che siano stati influenzati dalla modifica.

2.8.2.5 Test di accettazione Consiste nel collaudare il prodotto software in presenza del proponente. Se questo collaudo viene superato, si può procedere al rilascio ufficiale del prodotto sviluppato.

2.9 Misure e metriche

Il processo di verifica deve essere quantificabile per essere informativo. Quindi è necessario stabilire a priori delle metriche su cui basare le misurazioni del processo di verifica. Essendo le metriche di natura variabile, vengono definite due tipologie di intervalli:

- **Accettazione:** sono valori che vengono richiesti affichè il prodotto sia accettato;
- **Ottimale:** sono valori entro cui è consigliabile che la misurazione si collochi. Non sono intervalli vincolanti, ma consigliati. Se tali valori si scostano, è necessaria una verifica approfondita.

2.9.1 Metriche per i processi

Come metriche per valutare i processi si è scelto di utilizzare degli indici che analizzino i costi e i tempi.

2.9.1.1 Schedule Variance (SV) Valuta se si è in linea, in anticipo o in ritardo rispetto alla pianificazione della *baseline*$_G$. Questo è un indice di efficacia.
Se SV$_G$>0 significa che il team sta producendo con maggior velocità rispetto alla pianificazione, viceversa se è negativo.

Parametri utilizzati:

- Range di accettazione: [> -(Costo preventivo fase x 5%)];
- Range ottimale: [>0].

2.9.1.2 Budget Variance (BV) Valuta nella data corrente la differenza tra la spesa attuale e quanto pianificato.
Se BV$_G$>0 significa che il progetto sta consumando il budget più lentamente di quanto pianificato, viceversa se negativo.

Parametri utilizzati:

- Range di accettazione: [> -(Costo preventivo fase x 10%)];
- Range ottimale: [>0].

2.9.2 Metriche per i documenti

Come metrica per i documenti si è scelto di utilizzare un indice di leggibilità. L'indice utilizzato è specifico per la lingua italiana.

2.9.2.1 Gulpease$_G$(?) L'indice Gulpease$_G$è un indice di leggibilità tarato sulla lingua italiana. Viene preferito rispetto ad altri poiché utilizza la lunghezza in lettere anziché in sillabe, semplificandone il calcolo. Questo indice evidenzia la complessità dello stile del documento.
L'indice è calcolato secondo la seguente formula:

aggiungere formula

I risultati sono compresi tra 0 e 100, con 100 la migliore leggibilità e 0 la peggiore. **Parametri utilizzati:**

- Range di accettazione: [40-100];
- Range ottimale: [50-100].

2.9.3 Metriche per il software

É desiderabile, per poter raggiungere obiettivi di qualità del software, applicare le metriche ora descritte.

2.9.3.1 Numero di livelli di annidamento Rappresenta il numero di livelli di annidamento dei metodi, cioè l'annidamento di strutture.
Un alto livello di questo indice può essere sintomo di alta complessità del codice o un basso livello di astrazione.
Parametri utilizzati:

- Range di accettazione: [1-6];
- Range ottimale: [1-3].

2.9.3.2 Numero di attributi per classe Rappresenta il numero di attributi contenuti in una classe. Un alto valore potrebbe indicare la necessità di suddividere la classe in più classi. E quindi potrebbe essere indice di un errore di progettazione.
Parametri utilizzati:

- Range di accettazione: [0-16];
- Range ottimale: [3-8].

2.9.3.3 Numero di parametri per metodo Rappresenta il numero di parametri dei vari metodi. Un alto valore potrebbe indicare un metodo con funzionalità eccessivamente complesse. E quindi errori di progettazione.
Parametri utilizzati:

- Range di accettazione: [0-8];
- Range ottimale: [0-4].

2.9.3.4 Linee di codice per linee di commento Viene calcolato come il rapporto tra le linee di commento e le linee di codice. Valori ottimali di questo parametro indicano un codice più mantenibile.
Parametri utilizzati:

- Range di accettazione: [>0.25];
- Range ottimale: [>0.30].

2.9.3.5 Copertura di codice Indica la percentuale di istruzioni eseguite durante i test. Maggiore è questo parametro, minore sarà la probabilità di errori nel codice. Questo valore può essere abbassato dalla presenza di metodi semplici, come *setter* o *getter*.
Parametri utilizzati:

- Range di accettazione: [42%-100%];
- Range ottimale: [65%-100%].

3 Gestione amministrativa della revisione

3.1 Comunicazione e risoluzione di anomalie

Un *anomalia* corrisponde a:

- Violazione delle norme tipografiche di un documento;
- Violazione dei range di accettazione da parte degli indici di misurazione, descritti nella Sezione 2.9;
- Incongruenze tra il codice ed il *design* del prodotto;
- Incongruenze delle funzionalità del prodotto con funzionalità indicate nell'*Analisi dei Requisiti v1.0.0*.

Se un *Verificatore* individua un'anomalia, dovrà aprire un nuovo task nella sezione Task di *Teamwork*$_G$, come descritto nelle *Norme di Progetto v1.0.0*.

4 Standard di qualità

4.1 Standard ISO/IEC 15504

Questo standard descrive come i processi debbano essere controllati costantemente, al fine di rilevare possibili rischi intrinsechi che potrebbero impedire di raggiungere gli obiettivi prefissati. I risultati delle singole valutazioni devono essere ripetibili, oggettivi e comparabili per far si che contribuiscano al miglioramento effettivo dei processi in esame.

Il modello SPICE$_G$ definisce 6 livelli di maturità del processo:

- **Incomplete process:** il processo non viene attuato o non raggiunge i risultati previsti;
- **Performed:** il processo viene completato e raggiunge i risultati previsti.
  - **Process performance attribute:** capacità di raggiungere gli obiettivi.
- **Managed process:** il processo viene eseguito in maniera controllata a seconda degli obiettivi
  - **Performance management attribute:** è la capacità di un processo di elaborare un prodotto coerente con gli obiettivi;
  - **Work product management attribute:** è la capacità di un processo di elaborare un prodotto documentato, controllato e verificato.
- **Established process:** il processo viene eseguito in base a principi dell'Ingegneria del Software
  - **Process definition attribute:** l'esecuzione di processo si basa sugli standard per raggiungere gli obiettivi;
  - **Process resource attribute:** capacità del processo di utilizzare le risorse tecniche e umane appropriate, per essere efficaci.
- **Predictable process:** il processo viene eseguito costantemente in limiti predefiniti al fine di raggiungere i risultati attesi
  - **Measurement attribute:** gli obiettivi e le misure dei prodotti e dei processi sono utilizzati per garantire il raggiungimento degli obiettivi;
  - **Process control attribute:** il processo è controllato grazie alle misure di prodotto e processo, per effettuare correzioni migliorative.
- **Optimizing process:** il processo cambia e si adatta costantemente per raggiungere gli obiettivi
  - **Process change attribute:** i cambiamenti strutturali, di gestione e di esecuzione sono gestiti in maniera controllata al fine di raggiungere gli obiettivi;
  - **Continuous improvement attribute:** ogni modifica ai processi è identificata e implementata per garantire il miglioramento nella realizzazione degli obiettivi di business.

Ogni attributo di processo descritto è misurabile e lo standard predispone quattro livelli:

- **Non posseduto:** 0% - 15%;
- **Parzialmente posseduto:** 15% - 50%;
- **Largamente posseduto:** 50% - 85%;
- **Completamente posseduto:** 85% - 100%;

4.2 Ciclo di Deming

Il ciclo PDCA$_G$ è suddiviso in 4 fasi:

- **Plan:** fase di pianificazione che definisce le attività, le risorse, le scadenze e le responsabilità;
- **Do:** fase di esecuzione delle attività programmate;
- **Check:** fase di verifica nella quale vengono controllati i risultati della fase precedente, e confronta i prodotti effettivi con quelli attesi;
- **Act:** fase che prevede il miglioramento continuo dei processi, utilizzando i risultati della verifica per modificare le parti critiche dei processi.

4.3 Standard ISO/IEC 9126

Questo standard ISO$_G$/IEC$_G$ 9126 è stato creato con lo scopo di descrivere gli obiettivi qualitativi del prodotto e definire le metriche che possono misurare il raggiungimento di questi obiettivi. Vi sono infatti criteri suddivisi in tre aree:

- **Qualità in uso:** è la qualità del prodotto *software*, visto dall'utilizzatore che ne fa uso in uno specifico contesto;
- **Qualità esterna:** è la qualità del prodotto *software*, visto dall'esterno quando viene eseguito e testato in un ambiente di prova;
- **Qualità interna:** è la qualità del prodotto *software*, visto dall'interno, facendo riferimento a caratteristiche implementative, come l'architettura o il codice che ne deriva.

Essendo inizialmente impossibile testare la qualità in uso, si è deciso di focalizzarsi sulla qualità interna ed esterna. Nello standard sono previste sei caratteristiche qualitative principali, suddivise poi a sua volta in sottocategorie che possono essere misurate quantitativamente:

- **Funzionalità:** capacità del prodotto di fornire le funzioni richieste
  - **Idoneità:** capacità del prodotto di fornire un insieme di funzioni appropriate per un attività;
  - **Accuratezza:** capacità del prodotto di fornire risultati coerenti e corretti a seconda del grado di precisione richiesto;
  - **Interoperabilità:** capacità interattiva del prodotto *software* con uno o più sistemi;
  - **Sicurezza:** capacità del prodotto *software* di proteggere i dati e le informazioni, e a consentire solo a sistemi autorizzati di fare modifiche;
  - **Conformità funzionale:** capacità del prodotto *software* di essere aderente allo standard in materia di funzionalità.
- **Affidabilità:** capacità del prodotto di garantire un livello di prestazioni adeguato
  - **Maturità:** capacità del *software* di evitare fallimenti causati da errori;
  - **Tolleranza agli errori:** capacità del *software* di mantenere un adeguato livello di prestazioni nonostante errori o violazioni dell'interfaccia;
  - **Capacità di recupero:** capacità del *software* di ristabilire un livello di *performance* adeguato e recuperare dati in caso di errori;
  - **Conformità di affidabilità:** capacità del prodotto *software* di essere aderente allo standard in materia di affidabilità.
- **Usabilità:** capacità del prodotto di essere facilmente comprensibile ed usabile, e di essere interessante per l'utente

  - **Intelligibilità:** capacità del *software* di permettere all'utente di capire se il prodotto è adeguato e possa essere utilizzato per dei compiti particolari;
  - **Apprendibilità:** capacità del *software* di consentire all'utente di imparare ad utilizzare le sue funzionalità;
  - **Operabilità:** capacità del *software* di consentire ai suoi utenti di essere usato;
  - **Attrattività:** capacità del *software* di creare interesse negli utenti;
  - **Conformità di usabilità:** capacità del prodotto *software* di essere aderente allo standard in materia di usabilità.

- **Efficienza:** capacità del prodotto di fornire prestazioni adeguate a seconda delle risorse utilizzate
  - **Comportamento temporale:** capacità del *software* di dare una risposta in tempi di elaborazione adeguati, quando svolge le sue funzionalità;
  - **Utilizzo di risorse:** capacità del *software* di utilizzare le risorse adeguate per la sua esecuzione;
  - **Conformità di efficienza:** capacità del prodotto *software* di essere aderente allo standard in materia di efficienza.

- **Manutenibilità:** capacità del prodotto di essere modificato e aggiornato. Queste modifiche o aggiornamenti possono essere correzioni, miglioramenti o adattamenti del *software* a cambiamenti ambientali, di specifiche o delle funzionalità
  - **Analizzabilità:** capacità del *software* di poter essere studiato per cercare difetti;
  - **Modificabilità:** capacità del *software* di consentire l'implementazione di una qualche modifica;
  - **Stabilità:** capacità del *software* di evitare errori o fallimenti causati da una o più modifiche;
  - **Testabilità:** capacità del *software* di permettere la validazione di una versione modificata del *software*;
  - **Conformità di manutenibilità:** capacità del prodotto *software* di essere aderente allo standard in materia di manutenibilità.

- **Portabilità:** capacità del *software* di poter essere spostato in vari ambienti di lavoro
  - **Adattabilità:** capacità del *software* di essere adattato a diversi ambienti senza necessitare di modifiche;
  - **Installabilità:** capacità del *software* di poter essere installato in ambienti differenti;
  - **Coesistenza:** capacità del *software* di coesistere con altri *software* indipendenti condividendo delle risorse;
  - **Sostituibilità:** capacità del *software* di poter sostituire un *software* simile che abbia le stesse funzionalità e che sia nello stesso ambiente;
  - **Conformità di portabilità:** capacità del prodotto *software* di essere aderente allo standard in materia di portabilità.