Obelix Group
obelixswe@gmail.com

Piano di Qualifica

Versione	*v1_0_0*
Data creazione	2017-02-25
Redattori	Tomas Mali
	Riccardo Saggese
Verificatori	Silvio Meneguzzo
Approvazione	Nicolò Rigato
Stato	Approvato
Uso	esterno

Sommario

Documento contenente le strategie adottate dal gruppo Obelix per raggiungere gli obiettivi qualitativi richiesti per il prodotto Monolith.

Diario delle revisioni

Modifica	Autore e Ruolo	Data	Versione
Approvazione documento	Nicolò Rigato Responsabile	2017-03-09	1.0.0
Verifica del documento	Silvio Meneguzzo Verificatore	2017-03-09	0.1.0
Stesura sezione Resoconto delle attività di verifica	Tomas Mali Verificatore	2017-03-05	0.0.6
Stesura sezione Gestione amministrativa della revisione	Riccardo Saggese Verificatore	2017-03-04	0.0.5
Stesura sezione Strategie di Verifica	Tomas Mali Verificatore	2017-02-28	0.0.4
Stesura sezione Definizione obiettivi di qualità	Riccardo Saggese Verificatore	2017-02-27	0.0.3
Stesura sezione Introduzione	Tomas Mali Verificatore	2017-02-26	0.0.2
Creazione template	Nicolò Rigato Responsabile	2017-02-25	0.0.1

Indice

Elenco delle tabelle

1 Introduzione

1.1 Scopo del documento

Lo scopo principale di questo documento è di ottenere una buona qualità, intesa non solo come qualità di prodotto, ma anche come qualità di processo. Per raggiungere tale obiettivo è necessario un lavoro intellettuale e arduo poiché la qualità del software è diverso dagli altri prodotti manifatturieri ed è difficile misurarla in maniera oggettiva. Per rilevare e successivamente correggere le anomalie in modo efficace è necessaria un'attività costante di verifica e validazione da parte del $team_{|G|}$ Obelix.

1.2 Scopo del prodotto

Lo scopo del prodotto è quello di creare un $SDK_{|G|}$ che permetta la realizzazione di cosiddette bolle interattive di diversi tipi in base alle richieste di utenti (che nel nostro caso saranno gli sviluppatori). Il prodotto si completerà con la realizzazione di una demo presentabile mediante una web app con obiettivo di testare e provare il corretto funzionamento del SDK.

1.3 Riferimenti

1.3.1 Normativi

- **Norme di progetto**: *Norme di Progetto v1.0.0*:
- **Capitolato d'appalto C5**: *RedBabel, Monolith* `http://www.math.unipd.it/~tullio/IS-1/2016/Progetto/C5.pdf/`

1.3.2 Informativi

- **Piano di Progetto**: *Piano di Progetto*
- **PDCA (Plan-Do-Check-Act):** `http://it.wikipedia.org/wiki/PCDA`
- **Standard ISO/IEC 12207:2008-IEEE Std 12207-2008**: `http://ieeexplore.ieee.org/xpl/mostRecentIssue.jsp?punumber=4475822`
- **Standard ISO/IEC 15504**: `http://en.wikipedia.org/wiki/ISO/IEC_15504/`
- **Standard ISO/IEC 9126**: `http://it.wikipedia.org/wiki/ISO/IEC_9126`
- **Indice di Gulpease**: `http://it.wikipedia.org/wiki/Indice_Gulpease`

2 Definizione obiettivi di qualità

Vengono riportati gli obbiettivi di qualità che il team Obelix dovrebbe raggiungere durante lo svolgimento del progetto. Il team adotta degli standard, modelli e metriche per convincersi se un obiettivo è stato raggiunto o meno.

2.1 Qualità di processo

La qualità di processo è definita dallo standard ISO/IEC 15504 come SPICE. Questo standard specifica come la qualità è collegata alla maturazione dei processi. Non sarebbe possibile definire la qualità del prodotto senza garantire la qualità del processo. La qualità del prodotto nasce dunque dalla qualità del processo. Per garantire tutto ciò il team Obelix fa uso dello standard ISO/IEC 15504 denominato SPICE, il quale fornisce gli strumenti necessari a valutare l'idoneità dei processi. Per l'esattezza sono previsti sei livelli di maturità:

- **Livello 0 - Incomplete process**
  Il processo non è implementato o non riesce a raggiungere i suoi obiettivi
- **Livello 1 - Performed process**
  il processo viene messo in atto e raggiunge i suoi scopi
- **Livello 2 - Managed process**
  il processo viene eseguito sulla base di obiettivi ben definiti
- **Livello 3 - Established process**
  il processo viene eseguito in base ai principi dell'ingegneria del software
- **Livello 4 - Predictable process**
  il processo è attuato all'interno di limiti ben definiti
- **Livello 5 - Optimizing process**
  il processo è predicibile ed è in grado di adattarsi per raggiungere obiettivi specifici e rilevanti

2.1.1 Standard ISO/IEC 15504

2.1.2 Ciclo di Deming

Il ciclo di Deming, chiamato anche metodo PDCA, prevede l'iterazione ripetuta tra i quattro stadi (PLAN,DO,CHECK,ACT) assicurando un incremento della qualità ad ogni ciclo. Il team Obelix ha scelto dunque questo metodo per il controllo delle attività di processo ripetibili e misurabili e per la manutenibilità dei processi stessi.

2.2 Qualità del prodotto

Per garantire la qualità del prodotto il team Obelix cercherà di aderire al meglio allo standard di qualità SO/IEC 9126.

2.2.1 Standard ISO/IEC 9126

Aderendo a questo standard il team si impegna a garantire nel prodotto le seguenti qualità (definite con relativa metrica, misura e strumenti di controllo):

1. **Funzionalità**: il sistema prodotto deve garantire tutte le funzionalità indicate nel documento Analisi dei Requisiti v1.0.0. L'implementazione dei requisiti deve essere il più completa possibile

- **Misura**: l'unità di misura usata sarà la quantità di requisiti mappati sulle componenti del sistema create e funzionanti
- **Metrica**: la sufficienza è stabilita nel soddisfacimento di tutti i requisiti obbligatori
- **Strumenti**: per soddisfare questa qualità il sistema deve superare tutti i test previsti. Per informazioni dettagliate sugli strumenti si veda *Norme di Progetto v1.0.0*

2. **Affidabilità**: il sistema deve dimostrarsi il più possibile robusto e di facile ripristino in caso di errori

   - **Misura**: l'unità di misura utilizzata sarà la quantità di esecuzioni del sistema andate a buon fine
   - **Metrica**: le esecuzioni dovranno spaziare il più possibile sulle varie parti del codice
   - **Strumenti**: si vedano le metriche adottate per la copertura a

3. **Usabilità**: il sistema prodotto deve risultare di facile utilizzo per la classe di utenti a cui è destinato. Tale sistema deve essere facilmente apprendibile e allo stesso tempo deve soddisfare tutte le necessità dell'utente.

   - **Misura**: l'unità di misura usata sarà una valutazione soggettiva dell'usabilità. Questo è dovuto all'inesistenza di una metrica oggettiva adatta allo scopo
   - **Metrica**: non esiste una metrica adeguata che determinerà la sufficienza su questa qualità
   - **Strumenti**: si vedano le *Norme di Progetto v1.0.0*

4. **Efficienza**: il sistema deve fornire tutte le funzionalità nel più breve tempo possibile, riducendo al minimo l'utilizzo di risorse

   - **Misura**: il tempo che lo sviluppatore impiegherà per la creazione di una bolla interattiva
   - **Metrica**: non è possibile definire una soglia oggettiva di sufficienza perché non è possibile valutare ogni possibile casistica di utilizzo
   - **Strumenti**: si vedano le *Norme di Progetto v1.0.0*

5. **Manutenibilità**: il sistema deve essere comprensibile ed estensibile in modo facile e verificabile

   - **Misura**: le metriche sul codice descritte in §3.5.3 costituiscono l'unità di misura
   - **Metrica**: il software avrà le caratteristiche di manutenibilità descritte. Questo sarà possibile se il prodotto avrà la sufficienza in tutte le metriche
   - **Strumenti**: si vedano le *Norme di Progetto v1.0.0*

6. **Portabilità**: il sistema deve essere il più portabile possibile, in particolare dentro a Rocket.chat

- **Misura**: il front end deve rispettare gli standard W3C
- **Metrica**: il software avrà le caratteristiche di portabilità descritte. Questo sarà possibile se il prodotto avrà la sufficienza in tutte le metriche, descritte in §3.5
- **Strumenti**: si vedano le Norme del Progetto v1.0.0

3 Visione generale della strategia di verifica

Con le strategie utilizzate si cerca di automatizzare il lavoro di verifica. Lo scopo è quello di ottenere un riscontro affidabile ed adeguato per assicurare un grado di qualità predeterminato e ridurre il lavoro manuale permettendo cosi una validazione semplificata.

3.1 Procedure di controllo della qualità di processo

Le linee guida per la gestione della qualità di processo seguono il modello $PDCA_{|G|}$ descrivendo come devono essere attuate le procedure di controllo:

- Pianificazione dettagliata
- Monitoraggio delle attività pianificate item Definizione delle risorse necessarie al conseguimento degli obiettivi
- Utilizzo di metriche per verificare il miglioramento delle qualità dei processi

3.2 Procedure di controllo della qualità di prodotto

Con i seguenti processi verrà garantita il controllo della qualità del prodotto:

- **Software Quality Assurance (SQA)**: è l'insieme delle attività che serve per garantire il raggiungimento degli obiettivi di qualità
- **Verifica**: assicura che non siano stati introdotti errori nel prodotto con l'esecuzione dei processi
- **Validazione**: è la conferma oggettiva che assicura che i prodotti finali soddisfino i requisiti e le aspettative attese

3.3 Organizzazione

Viene verificato la qualità dei singoli processi e dei loro output.

3.3.1 Analisi

In questo periodo verrà verificata la corrispondenza tra i casi d'uso e i requisiti. Verrà inoltre controllato il rispetto dei processi e della documentazione prodotta.

3.3.2 Progettazione Architetturale

In questo periodo avviene la verifica dei processi relativi all'analisi e ai nuovi documenti di progettazione. Inoltre si verifica che i test siano adeguatamente pianificati come descritti ed eseguiti nel Norme di Progetto v1.0.0.

3.3.3 Progettazione di Dettaglio e Codifica

In questo periodo avviene la verifica dei processi relativi alla progettazione insieme alla verifica delle attività di codifica tramite tecniche di analisi statica e dinamica.

3.4 Responsabilità

La responsabilità delle verifiche è attribuita al *Responsabile* di progetto e ai *Verificatori* . All'interno del *Piano di Progetto v1.0.0* sono definiti i compiti e le modalità di attuazione.

3.5 Misure e metriche

Con lo scopo di poter monitorare in modo consapevole l'andamento dei processi e la qualità del prodotto si utilizzano delle metriche per rendere misurabili e valutabili in modo oggettivo alcune caratteristiche di documenti, processi e software.

3.5.1 Metriche per i documenti

I documenti sono di qualità se sono leggibili. La leggibilità è misurabile tramite un indice calcolabile sulle caratteristiche del testo.

Indice Gulpease L'indice Gulpease misura la leggibilità di un testo in Italiano utilizzando variabili linguistiche come la lunghezza delle parole e delle frasi.

$$G = 89 + \frac{300 \times \text{numero delle frasi} - 10 \times \text{numero delle lettere}}{\text{numero delle parole}}$$

Il valore risultante è un numero compreso tra 0 e 100, dove 100 indica la leggibilità massima e 0 la leggibilità minima. I documenti sono da considerare secondo le seguenti fasce:

- $G < 80$ sono considerati difficili per chi ha la sola licenza elementare
- $G < 60$ sono considerati difficili per chi ha la sola licenza media
- $G < 40$ sono considerati difficili per chi ha un diploma di scuola superiore

Parametri utilizzati

- Range di accettazione [40-90]
- Range ottimale [50-90]

3.5.2 Metriche per i processi

Le metriche per i processi hanno lo scopo di monitorare e rendere prevedibile l'andamento delle variabili di maggior criticità del progetto: tempo e costo. Le metriche sono utilizzate in modo consultivo per consentire un riscontro immediato sullo stato attuale del progetto; ad ogni incremento verranno valutati tali indici e, se necessario, verranno stabiliti opportuni provvedimenti da parte del *Responsabile* di progetto.

Schedule Variance Permette di calcolare le tempistiche rispetto l'organizzazione delle attività pianificate alla data corrente. É un indicatore di efficacia soprattutto a beneficio del cliente.

$$SV = BCWP - BCWS$$

Dove:

- **BCWP**: indica il valore delle attività realizzate alla data corrente
- **BCWS**: indica il costo pianificato per realizzare le attività di progetto alla data corrente

Quindi con:

- $SV > 0$: il lavoro prodotto è in anticipo rispetto quanto pianificato
- $SV < 0$: il lavoro è in ritardo
- $SV = 0$: il lavoro è in linea con quanto stabilito

Budget Variance Permette di calcolare i costi rispetto alla data corrente. É un indicatore che ha un valore unicamente contabile e finanziario.

$$BV = BCWS - ACWP$$

Dove:

- **BCWS**: indica il costo pianificato per realizzare le attività di progetto alla data corrente
- **ACWP**: indica il costo effettivamente sostenuto per realizzare le attività di progetto alla data corrente

Quindi:

- $BV > 0$: il budget speso è minore di quanto pianificato
- $BV < 0$: il budget speso è maggiore di quanto pianificato
- $BV = 0$: il budget speso è in linea con quanto stabilito

Produttività

Produttività di documentazione Indica la produttività media nel redigere i documenti.

$$\text{Produttività di documentazione} = \frac{\text{Parole}}{\text{Ore persona}}$$

Dove:

- **Parole**: indica il numero di parole presente nei documenti
- **Ore persona**: indica il numero di ore produttive impiegate per realizzare tali documenti

Parametri utilizzati

- Range-ottimale:[≥ 100]

Produttività di test Indica la produttività media nell'effettuare test.

$$\text{Produttività di test} = \frac{\text{Numero di test}}{\text{Ore persona}}$$

Dove:

- **Numero di test**: indica il numero di test eseguiti
- **Ore persona**: indica il numero di ore produttive impiegate per l'esecuzione di tali test

Produttività di codifica Indica la produttività media nelle attività di codifica.

$$\text{Produttività di codifica} = \frac{\text{LOCs}}{\text{Ore persona}}$$

Dove:

- **LOCs**: indica il numero di linee di codice prodotte (Lines Of Code)
- **Ore persona**: indica il numero di ore produttive impiegate per produrre tale codice

Copertura del test Indica la percentuale di casi coperti dai test eseguiti.

$$\text{Copertura del test} = \frac{\text{Numero di funzioni testate} \times 100}{\text{Numero totale di funzioni disponibili}}$$

Parametri utilizzati

- Range-accettazione: $[70 - 100]$
- Range-ottimale: $[80 - 100]$

3.5.3 Metriche per software

Complessità ciclomatica La complessità ciclomatica è una metrica software che indica la complessità di un programma misurando il numero di cammini linearmente indipendenti attraverso il grafo di controllo di flusso. Nel grafo sopracitato i nodi corrispondono a gruppi indivisibili di istruzioni, mentre gli archi connettono due nodi se il secondo gruppo di istruzioni può essere eseguito immediatamente dopo il primo gruppo. Tale indice può essere applicato indistintamente a singole funzioni, moduli, metodi o $package_{|G|}$ di un programma. Si vuole utilizzare tale metrica per limitare la complessità durante le attività di sviluppo del prodotto software. Può rivelarsi utile durante il testing per determinare il numero di test necessari, essendo l'indice di complessità un limite superiore al numero di test necessari per raggiungere la copertura completa.

Parametri utilizzati

- Range-accettazione: $[0 - 25]$
- Range-ottimale: $[0 - 10]$

Numero di metodi - NOM Il Number of methods è una metrica usata per calcolare la media delle occorrenze dei metodi per package. Un package non dovrebbe contenere un numero eccessivo di metodi. Valori superiori al range ottimale massimo potrebbero indicare una necessità di maggiore scomposizione del package.

Parametri utilizzati

- Range-accettazione: $[3 - 10]$
- Range-ottimale: $[3 - 7]$

Variabili non utilizzate e non definite La presenza di variabili non utilizzate viene considerata pollution, pertanto non viene tollerata. Tali occorrenze vengono rilevate analizzando l'Abstract syntax tree eseguendo un confronto tra le variabili dichiarate e quelle inizializzate. Per sua natura, $Javascript_{|G|}$ non blocca l'insorgenza di tali occorrenze, pertanto si rischia di dichiarare una variabile e poi utilizzarne una con nome leggermente diverso, oppure semplicemente dichiarare una variabile che in seguito non verrà mai utilizzata.

Parametri utilizzati

- Range-accettazione: $[0 - 0]$
- Range-ottimale: $[0 - 0]$

Numero parametri per metodo Un numero elevato di parametri per un metodo potrebbe evidenziare un metodo troppo complesso. Non c'è una regola forte per il numero di parametri possibili in un metodo, ma citando Robert Martin in Clean Code[1] :

> The ideal number of arguments for a function is zero (niladic). Next comes one (monadic), followed closely by two (dyadic). Three arguments (triadic) should be avoided where possible. More than three (polyadic) requires very special justification – and then shouldn't be used anyway.

Vengono quindi seguite le linee guida dei seguenti parametri:

Parametri utilizzati

- Range-accettazione: $[0 - 8]$
- Range-ottimale: $[0 - 4]$

[1] Robert Martin, Clean Code: A Handbook of Agile Software Craftsmanship. Prentice Hall (2008)

Halstead La metrica di $Halstead_{|G|}$ oltre ad essere un indice di complessità, permette di identificare le proprietà misurabili del software e le relative relazioni. Si basa sulla necessità che una metrica sia indipendente dalla specifica piattaforma e dal linguaggio di programmazione. Sono identificati i seguenti dati all'interno di un problema astratto:

- n_1 : indica il numero di operatori distinti
- n_2 : indica il numero di operandi distinti
- N_1 : indica il numero totale di operatori
- N_2 : indica il numero totale di operandi

Da cui si ottiene:

- $n = n_1 + n_2$: vocabolario della funzione
- $N = N_1 + N_2$: lunghezza della funzione

Data la scarsa disponibilità in rete di valori di riferimento per il Javascript, i range saranno da valutare in un momento successivo alla RR.

Halstead difficulty per-function Il livello di difficoltà di una funzione misura la propensione all'errore ed è proporzionale al numero di operatori presenti.

$$D = (\frac{n_1}{2}) \times (\frac{N_2}{n_2})$$

Parametri utilizzati

- Range-accettazione: $[0 - 30]$
- Range-ottimale: $[0 - 15]$

Halstead volume per-function Il volume descrive la dimensione dell'implementazione di un algoritmo e si basa sul numero di operazioni eseguite e sugli operandi di una funzione. Il volume di una function senza parametri composta da una sola linea è 20, mentre un indice superiore a 1000 indica che probabilmente la funzione esegue troppe operazioni.

$$V = N \times \log_2 n$$

Parametri utilizzati

- Range-accettazione: $[20 - 1500]$
- Range-ottimale: $[20 - 1000]$

Halstead effort per-function Lo sforzo per implementare o comprendere il significato di una funzione è proporzionale al volume e al suo livello di difficoltà.

$$E = V \times D$$

Parametri utilizzati

- Range-accettazione: $[0 - 400]$
- Range-ottimale: $[0 - 300]$

Maintainability index Questa metrica[2] è una scala logaritmica da $-\infty$ a 171, calcolata sulla base delle linee di codice logiche, della complessità ciclomatica e dall'indice Halstead effort. Un valore alto indica una maggiore manutenibilità.

Parametri utilizzati

- Range-accettazione: $[> 70]$
- Range-ottimale: $[> 90]$

Statement Coverage Permette di calcolare quante linee di codice di ciascun modulo delle unità sono eseguite almeno una volta nell'esecuzione dei test. Tale metrica è espressa in percentuale.

Parametri utilizzati

- Range-accettazione: $[70 - 100]$
- Range-ottimale: $[85 - 100]$

Branch Coverage Permette di calcolare quanti rami della logica di flusso sono attraversati almeno una volta durante l'esecuzione dei test. Tale metrica è espressa in percentuale.

Parametri utilizzati

- Range-accettazione: $[70 - 100]$
- Range-ottimale: $[85 - 100]$

4 Gestione amministrativa della revisione

4.1 Comunicazione delle anomalie

Identificare le anomalie permette la correzione dei difetti ricercati dal processo di Software Quality Management e informa il *Responsabile* di progetto sullo stato del prodotto. Analizzare e catalogare le anomalie è utile per discutere, durante revisioni e riunioni, su quali modifiche e correzioni applicare e con quale priorità. Di seguito è presente la lista delle definizioni di anomalie (IEEE 610.12-90) adottate dal gruppo:

- **Error**: differenza riscontrata tra il risultato di una computazione e il valore teorico atteso

[2]Definita nel 1991 da Paul Oman e Jack Hagemeister alla University of Idaho.

- **Fault**: un passo, un processo o un dato definito in modo erroneo. Corrisponde a quanto viene comunemente definito come bug
- **Failure**: il risultato di un fault
- **Mistake**: azione umana che produce un risultato errato

4.2 Procedure di controllo per la qualità di processo

Le procedure di controllo per la qualità di processo hanno il fine di migliorare la qualità del prodotto e diminuire i costi e tempi di sviluppo. Esistono due approcci principali:

- **A maturità di processo**: riflette le buone pratiche di management e tecniche di sviluppo. L'obiettivo primario è la qualità del prodotto e la prevedibilità dei processi
- **Agile**: sviluppo iterativo senza l'overhead della documentazione e di tutti gli aspetti predeterminabili. Ha come caratteristica la responsività ai cambiamenti dei requisti cliente e uno sviluppo rapido.

Il team adotterà il primo approccio, essendo più adatto ad un gruppo inesperto. Con una visione proattiva si cerca di avere maggior controllo e previsione sulle attività da svolgere. Questa viene anche indicata come best practice per gruppi poco esperti. Il processo con maggiore influenza sulla qualità del sistema non è quello di sviluppo ma quello di progettazione. È qui che le capacità e le esperienze dei singoli danno un contributo decisivo. Il miglioramento dei processi è un processo ciclico composto da tre sotto-processi:

- **Misurazione del processo**: misura gli attributi del progetto, punta ad allineare gli obiettivi con le misurazioni effettuate. Questo forma una $baseline_{|G|}$ che aiuta a capire se i miglioramenti hanno avuto effetto
- **Analisi del processo**: vengono identificate le problematiche ed i colli di bottiglia dei processi
- **Modifiche del processo**: i cambiamenti vengono proposti in risposta alle problematiche riscontrate

Il team procederà nel seguente modo:

- Nell'appendice del seguente documento verranno inserite le misurazioni rilevate sulle metriche descritte in §3.5
- Le modifiche al processo vengono attuate all'inizio del processo incrementale successivo. Queste attività sono programmate nel *Piano di Progetto v1.0.0*

A Resoconto delle attività di verifica

A.1 Revisione dei Requisiti

L'attività di verifica svolta dai *Verificatori* è avvenuta come determinato dal Piano di Progetto v1.0.0 al termine della stesura di ogni documento previsto. La verifica svolta sui documenti e sui processi è avvenuta seguendo le indicazioni delle *Norme di Progetto v1.0.0* e misurando le metriche indicate in §3.5

A.2 Dettaglio delle verifiche

Documenti Vengono qui riportati i valori dell'indice Gulpease per ogni documento durante l'analisi e relativo esito basato sui range stabiliti in §3.5.1

Documento	Valore indice	Esito
Analisi dei Requisiti v1.0.0	70	superato
Norme di Progetto v1.0.0	58	superato
Piano di Progetto v1.0.0	65	superato
Piano di Qualifica v1.0.0	65	superato
Studio di Fattibilità v1.0.	67	superato

Tabella 2: Tabella risultati test Gulpease