Piano di Qualifica

Informazioni sul documento	
Versione	0.0.3
Redazione	Redattore
Verifica	Verificatore
Approvazione	Approvatore
Uso	Esterno
Lista di Distribuzione	ScalateKids
	Prof. Tullio Vardanega
	Prof. Riccardo Cardin

Diario delle modifiche

Versione	Autore	Ruolo	Data	Descrizione
0.0.1	Andrea Giacomo Baldan		2015-12-27	Creazione scheletro del documento

Indice

1 Sommario

1.1 Scopo del documento

Il seguente documento ha lo scopo di delineare le strategie che il gruppo *ScalateKids* ha deciso di adottare per garantire degli obiettivi qualitativi da applicare ai processi sfruttati per lo sviluppo del progetto **ActorBase**. Per raggiungere tali obiettivi è necessario svolgere un'attività di verifica continua in modo da rilevare e correggere errori o malfunzionamenti minimizzando lo spreco di risorse.

1.2 Scopo del Prodotto

Implementazione di un database NoSQL di tipo key-value orientato alla gestione di grandi moli di dati utilizzando il modello ad attori su JVM$_G$, comprensivo di un *domain specific language* (DSL$_G$) da utilizzare da riga di comando per poter interagire con il database.
Il progetto dovrà essere pubblicato su *Github*.

1.3 Glossario

Tutti i termini di carattere tecnico o fraintendibile e gli acronimi sono raccolti nel file Glossario v1.0.0; ogni occorrenza di parole nel *Glossario* è indicata da una "G" in pedice.

1.4 Riferimenti

1.4.1 Normativi

- **Capitolato d'appalto C1:** *Actorbase: a NoSQL DB based on the Actor model;* http://www.math.unipd.it/~tullio/IS-1/2015/Progetto/C1.pdf
- **Norme di Progetto:** Norme di Progetto v1.0.0.

1.4.2 Informativi

- **Piano di Progetto:** Piano di Progetto v1.0.0
- **Dispense fornite dall'insegnamento Ingegneria del Software mod. A:** http://www.math.unipd.it/~tullio/IS-1/2015/
- **Standard ISO$_G$/IEC$_G$ 15504:** https://en.wikipedia.org/wiki/ISO/IEC_15504
- **Standard ISO$_G$/IEC$_G$ 9126:** https://it.wikipedia.org/wiki/ISO/IEC_9126
- **Ciclo di Deming$_G$:** https://en.wikipedia.org/wiki/PDCA

2 Visione generale della strategia di verifica

2.1 Definizione obiettivi

2.1.1 Qualità di processo

Per perseguire la qualità del prodotto è necessario garantire la qualità dei processi attuati per la sua creazione. Per raggiungere questo scopo i *Verificatori* con approvazione del *Responsabile* hanno trovato utile adottare lo standard ISO$_G$/IEC$_G$ 15504[1] denominato SPICE (Software Process Improvement and Capability dEtermination). Il ciclo di Deming$_G$[2] (noto anche come ciclo PDCA$_G$) è stato scelto per applicare correttamente questo standard. Esso definisce un metodo di controllo costante sui processi consentendo in questo modo un miglioramento incrementale della qualità continuo.

2.1.2 Qualità di prodotto

I *Verificatori* con approvazione del *Responsabile* hanno deciso di seguire lo standard ISO$_G$/IEC$_G$ 9126[3] che ha lo scopo di delineare delle metriche per la misurazione della qualità di prodotto. Questa è fondamentale per garantire che il prodotto software$_G$ finale funzioni correttamente e rispetti degli obiettivi di qualità prefissati.

2.2 Procedure di controllo di qualità di processo

La qualità dei processi è assicurata dall'applicazione del ciclo PDCA$_G$ a ogni processo. Tale metodo consente di migliorare continuamente ogni processo coinvolto nell'attività di sviluppo. Per applicare correttamente il ciclo PDCA$_G$ è necessario avere una pianificazione di processi e risorse dettagliata. La descrizione dettagliata della pianificazione si trova nel Piano di Progetto v1.0.0.

2.3 Procedure di controllo di qualità di prodotto

Il controllo di qualità del prodotto si suddividerà in tre parti:

- Attuazione di tecniche di analisi statica e dinamica descritte nella sezione 2.7;
- Verifica costante dell'output di ogni attività, i cui risultati verranno riportati nell'appendice B;
- Validazione, ovvero la conferma che il prodotto risponda correttamente ai requisiti attesi.

2.4 Organizzazione

Per garantire la qualità del prodotto finale verranno effettuate attività di verifica su ogni processo attuato. I controlli su ogni processo avranno come obiettivo garantire la qualità sia del processo stesso sia dell'eventuale prodotto da esso ricavato.
Ogni fase descritta in Piano di Progetto v1.0.0 necessità di attività di verifica diverse a causa della differenza natura del loro output. Di seguito sono elencate le strategie di verifica usate per ciascuna fase:

[1] Per approfondimenti consultare la sezione A.1.
[2] Per approfondimenti consultare la sezione A.2.
[3] Per approfondimenti consultare la sezione A.3.

- **Analisi:** durante questa fase verrà verificato che ogni documento creato sia conforme a quanto descritto nel documento Norme di Progetto v1.0.0, inoltre sarà necessario controllare che ogni requisito sia associato a un caso d'uso e viceversa;
- **Progettazione:** durante questa fase l'attività di verifica si concentrerà sul processo incrementale dei documenti prodotti in fase di Analisi e sui processi attuati per l'attività di progettazione assicurandosi che ogni documento e ogni processo sia conforme a quanto descritto nelle Norme di Progetto v1.0.0;
- **Codifica:**durante questa fase l'attività di verifica si concentrerà sul processo incrementale dei documenti prodotti in fase di Analisi e sui processi attuati per l'attività di codifica assicurandosi che ogni documento e ogni processo sia conforme a quanto descritto nelle Norme di Progetto v1.0.0.

In ogni documento viene inoltre tenuto un diario delle modifiche per permettere di mantenere uno storico delle attività svolte su di esso e delle responsabilità per ogni attività.

2.5 Pianificazione strategica e temporale

L'attività di redazione di ogni documento sarà sempre anticipata da uno studio sulla struttura e sui contenuti del documento interessato. In questo modo si riduce la possibilità di errori concettuali e il tempo effettivo di scrittura del documento viene ridotto.
Il processo di verifica deve essere ben organizzato e pianificato per rispettare le scadenze fissate nel Piano di Progetto v1.0.0.
L'attività di verifica segue le metodologie descritte nelle Norme di Progetto v1.0.0. Esse sono state pensate in maniera tale da consentire l'individuazione e la correzione degli errori il più presto possibile evitando che si diffondano.

2.6 Responsabilità

Per organizzare al meglio il lavoro e in particolare le attività$_G$ di verifica e validazione vengono assegnati questi compiti ai soli *Verificatori* e al *Responsabile*. Inoltre le attività$_G$ di verifica non potranno essere svolte da un membro che ha partecipato allo sviluppo di tale sezione per garantire assenza di conflitto di interessi (vedi norme.

2.7 Tecniche di analisi

2.7.1 Analisi statica

L'analisi statica è una tencica atta a trovare errori e anomalie. Questa tecnica è applicabile sia alla documentazione sia al codice.

2.7.1.1 Walkthrough

Come descritto nelle *Norme di Progetto v1.0.0* verrà effettuata tecnica walkthrough nelle fasi iniziali di ogni attività$_G$ per passare il prima possibile a attività$_G$ di *Inspection* (paragrafo seguente).

2.7.1.2 Inspection

Successivamente all'attività$_G$ di 2.7.1.1Walthrough verrà fatta attività$_G$ di inspection (seguendo *Norme di Progetto* (paragrafo precedente)) rispetto alla lista di controllo ottenuta in precedenza.

2.7.2 Analisi dinamica

L'analisi dinamica si applica solo al prodotto software e serve a trovare errori o difetti nell'implementazione.
I test effettuati devono essere *ripetibili*; questa caratteristica è fondamentale poichè un test deve produrre sempre lo stesso risultato dato un input in un ambiente per trovare errori.

2.7.2.1 Test di unità

Test per verificare ogni unità del software prodotto. Per unità si intende la pià piccola quantità di software che è utile verificare singolarmente.
L'obbliettivo di questi test è trovare ed eliminare possibili errori creati dai *programmatori*.

2.7.2.2 Test di integrazione

Test per verificare che l'integrazione tra due o più unità software funzionino correttamente.
In mancanza di unità questi test possono essere svolti tramite la creazione di una o più componenti fittizie che simulano il comportamento delle unità con cui l'unità creata deve interfacciarsi.

2.7.2.3 Test di sistema

Test per verificare la totale copertura dei requisiti software emersi in fase di *Analisi dei Requisiti*. Questi test verificano la totalità del prodotto software al suo raggiungimento di una versione definitiva.

2.7.2.4 Test di regressione

Test da effettuare per verificare che una modifica del software non pregiudichi il funzionamento del sistema o di altre parti del prodotto prima correttamente funzionanti.
Questa attività dev'essere svolta tenendo conto del tracciamento dei requisiti per individuare i giusti test di unità, di integrazione e di sistema che possono essere influenzati dalle modifiche effettuate.

2.7.2.5 Test di accettazione

Questo corrisponde al collaudo vero e proprio del software da svolgersi col *Proponente*.

2.8 Metriche di valutazione

Per rendere quantificabile il processo di verifica verranno usate delle metriche elencate sotto. Soltando due tipi di range$_G$ saranno accettati:

- **Ottimale**: contiene i valori che si deve cercare di ottenere, valori al di fuori di questo range dovranno essere verificati e, se necessario, discussi dal gruppo;
- **Accettazione**: valori per avere un accetazione del prodotto.

2.8.1 Metriche per i documenti

Per la stesura dei documenti è stato scelto di utilizzare un *indice di leggibilità* per la lingua italiana.

2.8.1.1 Indice Gulpease

Questo indice indica il grado di leggibilità di un documento.
L'indice di Gulpease prende in considerazione principalmente tre variabili:

- Numero delle frasi;
- Numero delle lettere;
- Numero delle parole.

La formula applicata per calcolare il suddetto indice è:

$$89 + \frac{300 * (numerodellefrasi) - 10 * (numerodellelettere)}{numerodelleparole} \quad (1)$$

Lo script applicato su un documento produrrà in output un valore interno compreso tra zero e cento, in particolare i significati sono i seguenti:

- Indice inferiore a 80 rappresenta un documento difficile da leggere per persone con una licenza di scuola elementare;
- Indice inferiore a 60 rappresenta un documento difficile da leggere per persone con una licenza di scuola media inferiore;
- Indice inferiore a 40 rappresenta un documento difficile da leggere per persone con una licenza di diploma superiore;

Range accettabili:

- Range ottimale: 60 - 100;
- Range di accettazione: 45 - 60.

2.8.2 Metriche per il software

Per perseguire una buona qualità software sono state pensate diverse metriche di valutazione descritte nei prossimi paragrafi. Tuttavia questa sezione è una sezione che potrà subire cambiamenti durante le prossime revisioni.

2.8.2.1 Linee di codice per linee di commento

Applicato su un file produrrà un valo0re in output che rappresenta la percentuale di righe di commento rispetto alle linee di codice.
Range accettabili:

- Range ottimale: 30 - 40%;
- Range di accettazione: 20 - 30%.

2.8.3 Copertura del codice

La copertura del codice indica il numbero di istruzioni che vengono eseguite durante i test rispetto alla totalità delle istruzioni. Per avere una minor possibilità di presenza di errori nel codice si dovrà avere una percentuale il più alta possibile di istruzioni testate.
Tuttavia metodi molto semplici che non necessitano testing andranno a influire negativamente in questo ambito.
Range accettabili:

- Range ottimale: 60 - 100%;
- Range di accettazione: 40 - 60%.

2.8.4 Numero di attributi per classe

Un numero troppo elevato di attributi in una classe potrebbe indicare il non soddisfacimento del principio di *single responsability*$_G$. Questo potrebbe significare la necessità di una divisione in più classi relazionate tra loro per dividere le diverse funzionalità o un possibile errore di progettazione.
Range accettabili:

- Range ottimale: 3 - 6;
- Range di accettazione: 0 - 12.

2.8.5 Numero di parametri per metodo

Un numero troppo elevato di parametri in un metodo potrebbe indicare il non soddisfacimento del principio di *single responsability*$_G$. Questo potrebbe significare la necessità di una divisione in sotto metodi per dividere le diverse funzionalità o un possibile errore di progettazione.
Range accettabili:

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

2.8.6 Complessità ciclomatica

La complessità ciclomatica è usata per misurare la complessità di parti di codice come funzioni, metodi, classi o moduli.
Questa è calcolata attraverso un grafo di controllo del flusso del programma.
La complessità ciclomatica di una sezione di codice è il numero di cammini linearmente indipendenti attraverso il codice sorgente. Per esempio, se il codice sorgente non contiene punti decisionali come IF o cicli FOR, allora la complessità sarà 1, poiché esiste un solo cammino nel sorgente (e quindi nel grafo). Se il codice ha un singolo IF contenente una singola condizione, allora ci saranno due cammini possibili: il primo se l'IF viene valutato a TRUE e un secondo se l'IF viene valutato a FALSE.
Range accettabili:

- Range ottimale: 0 - 10;
- Range di accettazione: 10 - 15.

2.8.7 Livelli di annidamento

Indica il numero di livelli di annidamento di funzioni e metodi. Un valore alto di questo indice implica un livello di complessità molto elevato, aumentando la probabilità di commettere errori.
Range accettabili:

- Range ottimale: 1 - 4;
- Range di accettazione 5 - 8.

A Standard di qualità

A.1 Standard ISO/IEC 15504

Lo standard ISO_G/IEC_G 15504 definisce un metodo per controllare continuamente la qualità di ogni processo. Questo metodo sfrutta un modello denominato SPY (Software Process Assessment and Improvement) per la valutazione di processo.
Questo standard fornisce i seguenti nove attributi di qualità di processo:

- **Process performance:** il processo riesce a trasformare input identificabili in output identificabili.;
- **Performance management:** Il processo viene attuato e controllato seguendo una pianificazione che garantisca il raggiungimento degli obiettivi;
- **Work product management:** Il processo viene attuato e controllato seguendo una pianificazione che garantisca risultati documentati e verificati;
- **Process definition:** Il processo viene attuato seguendo standard di processo per garantire i risultati;
- **Process resource:** Il processo ha risorse adeguate per la sua attuazione;
- **Process measurement:** I risultati raggiunti e le misure rilevate durante l'attuazione di un processo sono stati usati per assicurarsi che tale processo supporti il raggiungimento di specifici obiettivi;
- **Process control:** Ogni processo è controllato attraverso la raccolta, analisi ed utilizzo delle misure di prodotto e di processo rilevate, per migliorare, eventualmente, le sue modalità di attuazione;
- **Process change:** Le modifiche alla definizione, gestione e attuazione di ogni processo sono controllate;
- **Continuous improvement:** Le modifiche ad ogni processo sono identificate ed implementate per assicurare il continuo miglioramento nel raggiungimento degli obiettivi rilevanti per l'organizzazione.

Questi attributi vengono utilizzati per suddividere la maturità di un processo nei seguenti sei livelli:

- **Livello 0 - Incompleto:** Il processo non raggiunge i suoi obiettivi;
- **Livello 1 - Attuato:** Il processo raggiunge i suoi obiettivi ma non è controllato. Il raggiungimento di questo livello è dimostrato dal possesso dell'attributo di *Process Performance*;
- **Livello 2 - Gestito:** Il processo è attuato, pianificato, tracciato, verificato e migliorato sulla base di obiettivi ben definiti. Il raggiungimento di questo livello è dimostrato dal possesso dell'attributo di *Performance management* e *Work product management*;
- **Livello 3 - Definito:** Il processo è attuato, pianificato e controllato su procedure basate sui principi del software engineering. Il raggiungimento di questo livello è dimostrato dal possesso dell'attributo di *Process definition* e *Process resource*;
- **Livello 4 - Predicibile:** Il processo è stabilizzato ed è attuato all'interno di definiti limiti riguardo i risultati attesi, le performance e le risorse impiegate. Il raggiungimento di questo livello è dimostrato dal possesso dell'attributo di *Performance measurement* e *Process control*;

- **Livello 5 - Ottimizzante:** Il processo è predicibile ed in grado di adattarsi per raggiungere obiettivi specifici e rilevanti per la organizzazione. Il raggiungimento di questo livello è dimostrato dal possesso dell'attributo di *Process change* e *Continuous improvement*.

Infine per stabilire in che livello si trovi un processo ci sono quattro livelli di possesso per ogni attributo:

- **N:** Non posseduto;
- **P:** Parzialmente posseduto;
- **L:** Largamente posseduto;
- **F:** Pientamente (Fully) posseduto.

A.2 Ciclo di Deming

Il ciclo di Deming$_G$, anche chiamato PDCA$_G$, è un modello che consente il miglioramento continuo dei processi. Si suddivide nelle seguenti quattro attività:

- **Plan:** attività di pianificazione di attività, risorse, scadenze e responsabilità;
- **Do:** attività di esecuzione delle attività pianificate;
- **Check:** attività di verifica dei risultati dell'attività di esecuzione, questi risultati verranno poi confrontati con i risultati pianificati nella fase di pianificazione;
- **Act:** attività di messa in pratica del miglioramento dei processi sfruttando i risultati dell'attività di verifica per trovare e modificare gli aspetti critici.

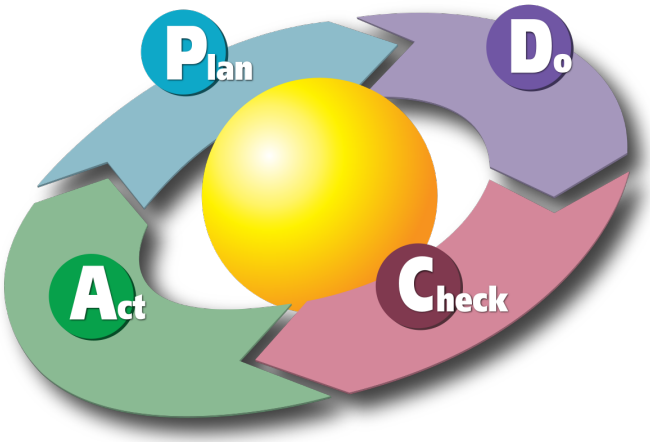

Figure 1: Ciclo di Deming

A.3 Standard ISO/IEC 9126

Lo standard ISO_G/IEC_G 9126 definisce un modello per migliorare l'organizzazione e i processi, avendo come conseguenza il miglioramento della qualità del prodotto finale. I criteri qualitativi sono divisi in tre macro aree:

- **Qualità interna:** è la qualità del prodotto software vista dall'interno. Si riferisce alla valutazione del codice sorgente e dell'architettura prima che il prodotto sia ultimato;
- **Qualità esterna:** è la qualità del prodotto software vista dall'esterno. Si riferisce al risultato dei test condotti sul prodotto ultimato in un ambiente di prova;
- **Qualità in uso:** è la qualità del prodotto vista dall'utilizzatore finale del prodotto. Si riferisce alla qualità dopo il rilascio dell'applicazione.

Non potendo testare la qualità in uso nell'ambito di questo progetto si è scelto di concentrarsi sulla qualità interna ed esterna. Queste qualità nello standard ISO_G/IEC_G 9126 sono ulteriormente suddivise in sei categorie:

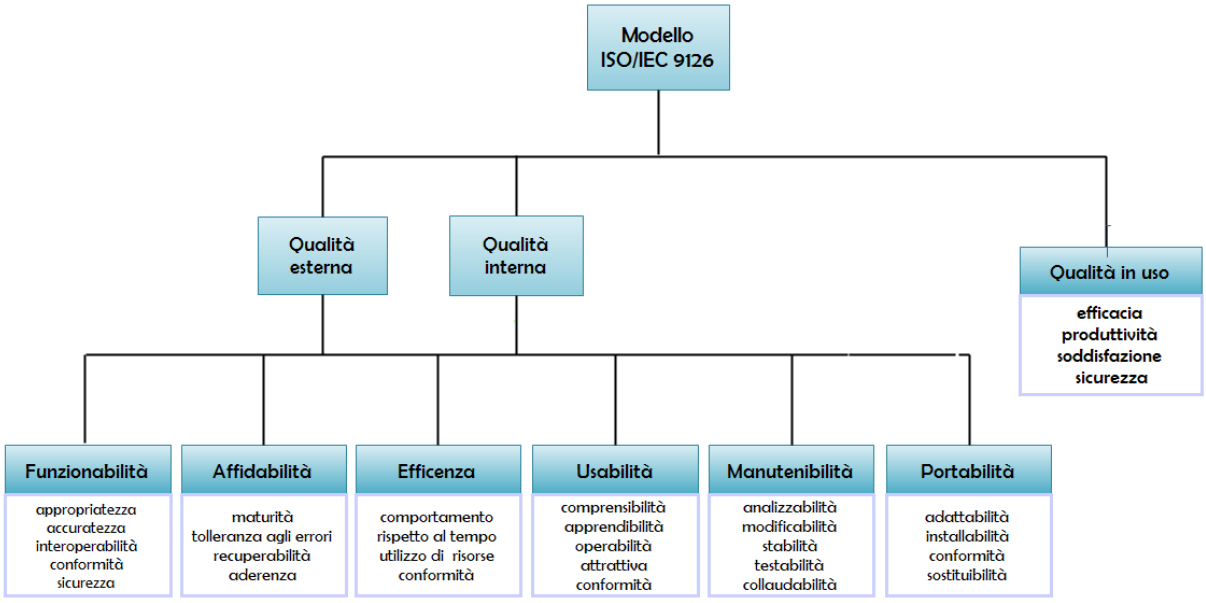

Figure 2: Modello dello standard ISO/IEC 9126

- **Funzionalità:** capacità del prodotto di fornire funzioni che soddisfino esigenze stabilite. Questa categoria si suddivide in:
  - **Appropriatezza:** capacità del prodotto di fornire un appropriato insieme di funzioni che rispondano a determinati requisiti;

  - **Accuratezza:** capacità del prodotto di fornire dei risultati corretti o con precisione concordata;
  - **Interoperabilità:** capacità del prodotto di interagire con uno o più sistemi esterni;
  - **Conformità:** capacità del prodotto di aderire a standard del settore di applicazione;
  - **Sicurezza:** capacità del prodotto di proteggere i dati, ad esempio negando l'accesso a utenti non autorizzati.

- **Affidabilità:** capacità del prodotto di mantenere uno specificato livello di prestazioni durante un determinato periodo. Questa categoria si suddivide in:
  - **Maturità:** capacità del prodotto di evitare che si verifichino risultati non corretti a causa di errori nel software;
  - **Tolleranza agli errori:** capacità del prodotto di mantenere uno specificato livello di prestazioni anche in presenza di errori nel software o di uso scorretto del prodotto;
  - **Recuperabilità:** capacità del prodotto di ristabilire un livello adeguato di prestazioni in seguito a un malfunzionamento;
  - **Aderenza:** capacità del prodotto di a standard riguardanti l'affidabilità.

- **Efficienza:** capacità del prodotto di fornire prestazioni appropriate in relazione alle risorse usate. Questa categoria si suddivide in:
  - **Comportamento rispetto al tempo:** capacità del prodotto di fornire i risultati in tempi adeguati;
  - **Utilizzo delle risorse:** capacità del prodotto di utilizzare le risorse in modo adeguato;
  - **Conformità:** capacità del prodotto di aderire a standard sull'efficienza.

- **Usabilità:** capacità del prodotto di essere capito e usato dall'utente. Questa categoria si suddivide in:
  - **Comprensibilità:** facilità di comprensione del prodotto;
  - **Apprendibilità:** capacità di ridurre l'impegno richiesto all'utente per imparare come usare correttamente il prodotto finale;
  - **Operabilità:** capacità del prodotto di fare in modo che gli utenti riescano a farne uso per i propri scopi;
  - **Attrattiva:** capacità del prodotto di essere piacevole per l'utente;
  - **Conformità:** capacità del prodotto di aderire a standard sull'usabilità.

- **Manutenibilità:** capacità del prodotto di essere modificato e migliorato. Questa categoria si suddivide in:
  - **Analizzabilità:** facilità di analisi del codice per la localizzazione di errori;
  - **Modificabilità:** capacità del prodotto di permettere la sostituzione di componenti per implementare una determinata modifica;
  - **Stabilità:** capacità del prodotto di evitare errori derivanti da modifiche errate;
  - **Testabilità:** capacità del prodotto di essere facilmente testato rispetto alle modifiche apportate.

- **Portabilità:** capacità del prodotto di essere trasportato in un altro ambiente di lavoro. Questa categoria si suddivide in:

- **Adattabilità:** capacità del prodotto di essere adattato per ambienti operativi diversi senza modifiche diverse da quelle fornite;
- **Installabilità:** capacità del prodotto di essere installato in specifici ambienti;
- **Conformità:** capacità del prodotto di aderire a standard sulla portabilità;
- **Sostituibilità:** capacità del prodotto di essere utilizzato al posto di un altro software per svolgere gli stessi compiti.

B Resoconto attività di verifica