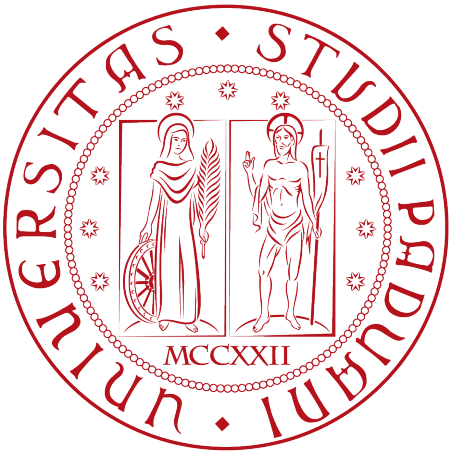
Università degli Studi di Padova

Dipartimento di Matematica "Tullio Levi-Civita"

Corso di Laurea in Informatica



Realizzazione di una applicazione di process mining per il filtraggio degli eventi

Tesi di laurea triennale

Relatore

Prof.Tullio Vardanega

Laureando

Fontolan Carlo

Anno Accademico 2018-2019

Lorem ipsum dolor sit amet, consectetuer adipiscing elit.

— Oscar Wilde

Dedicato a ...

Sommario

Il presente documento descrive il lavoro svolto durante il periodo di stage, dalla durata di circa trecentoquaranta ore, del laureando Carlo Fontolan presso l'azienda Siav S.p.A. Lo studente vuole descrivere con riflessioni critiche e oggettive quanto appreso e maturato dal punto di vista professionale e delle competenze in tutto l'arco di durata dello stage. In primo luogo era richiesto lo sviluppo di una libreria di filtraggio in abito process mining, che verrà poi inserità all'interno di una architettura a microservizi. In secondo luogo, è stata richiesta le riscrittura di alcune sezioni di un loro vecchio prodotto, in cui la libreria sopradescrittà farà da protagonista, per quanto riguarda la sezione di filtraggio. Le caratteristiche richieste dall'azienda proponente erano che il prodotto fosse idoneo a girare su cloud e che fosse costituito da un'architettura a microservizi e sviluppato con framework all'avanguardia. Siccome in corso d'opera sono sorte alcune criticità in merito allo sviluppo della libreria, in accordo con il tutor aziendale, abbiamo deciso di rinegoziare alcuni requisiti andando ad eliminare la sezione riguardante il *backend*, facendo spazio ad alcuni stub in modo da poter osservare il comportamento dell'interfaccia, soddisfando così gli obiettivi sia a livello aziendale che personale.

Ringraziamenti

Innanzitutto, vorrei esprimere la mia gratitudine al Prof. Tullio Vardanega, relatore della mia tesi, per l'aiuto e il sostegno fornitomi durante la stesura del lavoro.

Desidero ringraziare con affetto i miei genitori per il sostegno, il grande aiuto e per essermi stati vicini in ogni momento durante gli anni di studio.

Ho desiderio di ringraziare poi i miei amici per tutti i bellissimi anni passati insieme e le mille avventure vissute.

Padova, Dicembre 2019 Fontolan Carlo

Indice

Elenco delle figure

Elenco delle tabelle

Capitolo 1

Il contesto aziendale

1.1 L'azienda

Siav è una delle più importanti realtà italiane di sviluppo software e di servizi informatici specializzata nella dematerializzazione e nella gestione documentale e nei processi digitali. Presenta diverse filiali nel suolo italiano che cooperano tra loro per un fine comune. Siav inoltre punta molto sulla collaborazione tra aziende, in campo nazionale sia a livello pubblico che privato, ma anche a livello internazionale. Una fra tutti Microsoft. I loro servizi puntano ad una miglior gestione, controllo ed automazione di tutti i principali processi aziendali.

Figura 1.1: I prodotti offerti da Siav

1.2 Dominio Applicativo

Siav si colloca all'interno del mercato come una tra la più importanti realtà italiane a livello di *Enterprise Content Management* offrendo servizi per poter migliorare processi e gestioni aziendali spaziando da un gestore di caselle PEC, ad un applicativo di process mining denominato Bipod, fino ad arrivare al loro prodotto di punta: Archiflow/Silloge: un software in continuo sviluppo, mantenuto e aggiornato da piccoli team che lavorano in sinergia per raggiungere un obiettivo comune. Archiflow/Silloge offre una soluzione alla gestione di una cospicua mole di documenti, categorizzandoli in varie sezioni permettendone un facile reperimento.

1.3 Tecnologie utilizzate

Le tecnologie utilizzate dell'azienda per la realizzazione dei propri prodotti sono fondate principalmente sulla possibilità di un utilizzo tramite cloud. Si sono dedicati principalmente sullo sviluppo di *Webapp*. Tali tecnologie possono essere raggruppate in due macrocategorie: *Frontend* e *Backend*.

1.3.1 *Frontend*

Da quel che ho potuto constatare durante l'attività di stage, per quanto riguarda lo sviluppo web lato *client* viene utilizzata prevalentemente la piattaforma Angular. Tramite il consistente numero di componenti prefabbricati presenti in rete e una vasta gamma di funzionalità e servizi disponibili, è possibile sviluppare un'interfaccia frontend di qualità a proprio piacimento, soddisfando nel miglior modo possibile le aspettative del cliente.

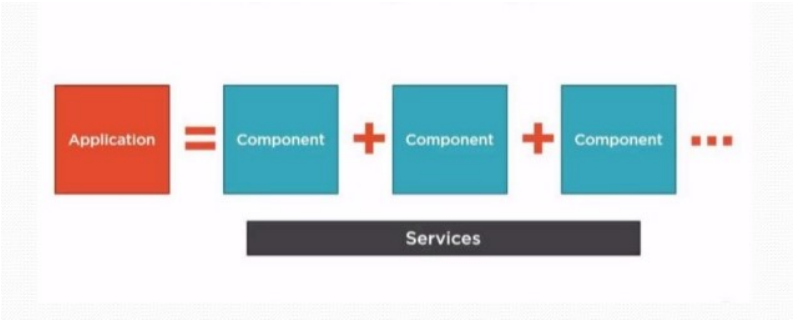


Figura 1.2: Esempio di struttura di un applicativo Angular

1.3.2 *Backend*

Per quanto riguarda lato backend vengono utilizzate molteplici tecnologie in base alle necessità che prevede il software di sviluppo; *RabbitMQ*: questa tecnologia è stata pensata principalmente per un'architettura a microservizi e viene utilizzata in più prodotti all'interno dell'azienda. *RabbitMQ* viene definito come un *Broker* di messaggisticia: ossia un sistema che monitora la trasmissioni di messaggi tra servizi. Un'altro sistema utilizzato, restano in ambito di microservizi è *Kubernetes*: un sistema per la gestione di *container*.

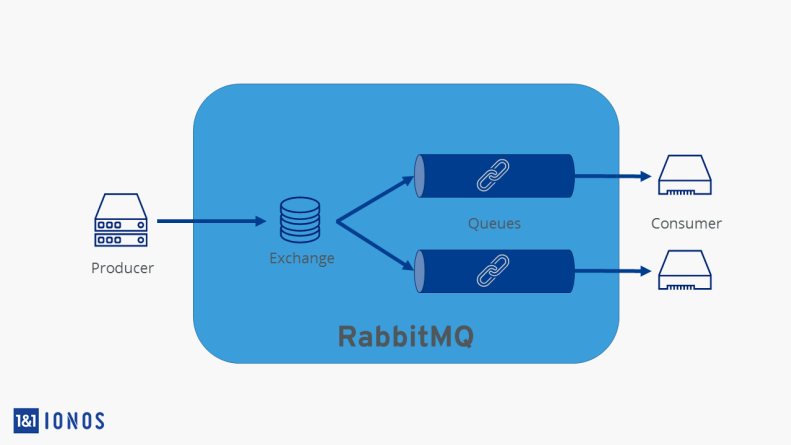


Figura 1.3: Diagramm illustrativo sul funzionamento di RabbitMQ

1.4 Processi aziendali

1.4.1 Metodologie si sviluppo

L'azienda normalmente si ritrova a dover far fronte ad alcune problematiche derivate da bug, criticità o variazioni importanti di requisiti che possono essere riscontrate durante i vari processi aziendali ed incidere in modo significativo sull'andamento dello sviluppo. Per cercare di venire incontro a ciò l'azienda ha adottato una metodologia di tipo *Agile*, cercando di mantenere un atteggiamento flessibile rispetto ai processi e gli obiettivi. Risulta quindi fondamentale l'organizzazione di incontri durante vari momenti della giornata o settimana in modo da confrontarsi sull'andamento dei processi e lo stato di avanzamento del prodotto. È consuetudine dei team di lavoro un confronto giornaliero tramite *daily meeting*, per discutere su quanto fatto durante la giornata precedente indicando le eventuali criticità riscontrate, pianificando cosi gli obiettivi per la giornata odierna. Ad inizio settimana invece viene organizzato *weekly meeting* per constatare gli stati di avanzamento rispetto alla settimana precedente, per poi fissare gli obiettivi per la settimana successiva. Qui di seguito sono riportati i principi cardine del Manifesto Agile:

* La nostra massima priorità è soddisfare il cliente rilasciando software di valore, fin da subito e in maniera continua.
* Accogliamo i cambiamenti nei requisiti, anche a stadi avanzati dello sviluppo. I processi *Agile* sfruttano il cambiamento a favore del vantaggio competitivo del cliente.
* Consegniamo frequentemente software funzionante, con cadenza variabile da un paio di settimane a un paio di mesi, preferendo i periodi brevi.
* Committenti e sviluppatori devono lavorare insieme quotidianamente per tutta la durata del progetto.
* Fondiamo i progetti su individui motivati. Diamo loro l'ambiente e il supporto di cui hanno bisogno e confidiamo nella loro capacità di portare il lavoro a termine.

* Una conversazione faccia a faccia è il modo più efficiente e più efficace per comunicare con il team ed all'interno del team.

* Il software funzionante è il principale metro di misura di progresso.

* I processi agili promuovono uno sviluppo sostenibile. Gli sponsor, gli sviluppatori e gli utenti dovrebbero essere in grado di mantenere indefinitamente un ritmo costante.

* La continua attenzione all'eccellenza tecnica e alla buona progettazione esaltano l'agilità.

* La semplicità - l'arte di massimizzare la quantità di lavoro non svolto - è essenziale.

* Le architetture, i requisiti e la progettazione migliori emergono da team che si auto-organizzano.

* A intervalli regolari il team riflette su come diventare più efficace, dopodiché regola e adatta il proprio comportamento di conseguenza.

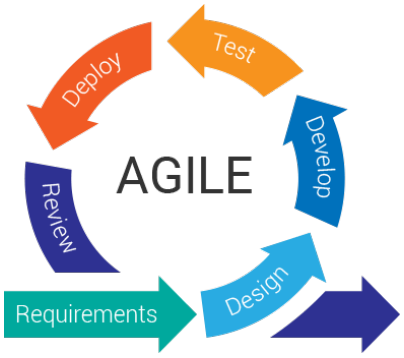


Figura 1.4: Ciclo di vita della metodologia Agile

1.4.2 Strumenti di supporto

Gestione di progetto: Per tenere traccia del lavoro svolto e delle modifiche apportate viene utilizzato *Evernote*: un software che permette la scrittura e la condivisione di note ad un gruppo cospicuo di utenti. Cosi facendo ogni membro del team di sviluppo ha sotto controllo ogni attività svolta dagli altri membri. Un ulteriore strumento utilizzato per il tracciamento dello attività è *Google Docs*, ma al contrario di *Evernote*, è possbilie assegnare task agli utenti che hanno i permessi di accesso al documento. Solitamente i task assegnati presentano una struttura tanto semplice quanto incisiva: è presente innanzitutto la data di creazione, la possibilità o meno di menzionare direttamente l'interessato a cui risulta assegnato il task ed un casella di risposta per poter descrivere la sua effettiva terminazione, oppure un messaggio di testo di altro genere, che solitamente indica le problematiche per cui il task non è stato possibile risolverlo.

Ambienti di sviluppo Gli ambienti di sviluppo adottati in Siav sono vari. Non è presente una direttiva specifica che indica gli *editor* e i compilatori da utilizzare. Solitamente, per lo sviluppo di interfacce web viene utilizzato *Visual Studio Code*: permette la creazione di più terminali all'interno della stessa sessione in modo da poter effettuare diverse operazioni in modo simulataneo. Oltre a questo permette una visualizzazione semplficiata delle modifiche apportate al repository, offrendo la possibilità di effettuare tutte le principale operazioni di *Git*. Per quanto riguarda lo sviluppo di applicativi in *Python* viene spesso utilizzato *PyCharm*, mentre per lo sviluppo in ambiente *Java* viene utilizzato *IntelliJ IDEA* oppure *Eclipse*, ma anche *Subilme text* è risultata una valida alternativa.

Controllo versionamento: Come strumento di versionamento l'azienda utilizza Git. In particolare viene utilizzata una versione *Enterprise* di *gitLab* con un dominio interno all'azienda. Non risulta quindi possibile l'esportazione di un repository al di fuori di questa. Ciò garantisce una sicurezza in più rispetto ai repository utilizzati per scopo accademico. I principali aspetti che hanno portato l'azienda all'utilizzo sono:

* *branch/merge:* Permette con facilità la creazione di *branch*, in modo che ogni sviluppatore possa lavorare nel proprio ramo senza intaccare quello principale (stabile) del rapository, evitando quindi la generazioni di conflitti o errori. Nel momento in cui il software sviluppato all'interno branch viene definito stabile è possibile riallinearlo con il ramo principale tramite un'operazione di *merge*.

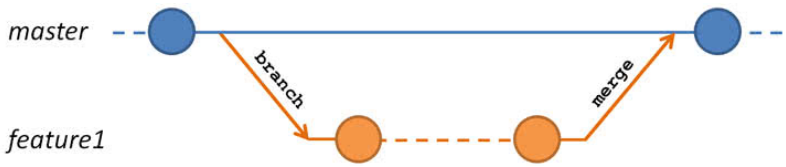


Figura 1.5: Esempio di creazione di un branch seguito da un'oprazione di merge

* Reperibilità: Anche in assenza di connessione è possibile effettuare commit in locale, in modo da non perdere i progressi fatti finora e salvare il lavoro fatto.

* Ridondanza: Ogni Sviluppatore contiene una copia del repository: Il rischio di perdita dei dati è dunque diviso tra i *contributors* che possiedono l'ultima versione del software in locale.

1.5 Clientela rivolta a Siav

La maggior parte dei clienti che si affidano a Siav sono aziende che presentano il bisogno di automatizzarsi o migliorare i propri processi interni andando cosi ad incrementare la propria efficenza sotto l'aspetto lavoartivo. Tali aziende sono di vario genere, dal semplice ristorante ad una nota catena di supermercati fino ad aziende metalmeccaniche. Per ogni settore, Siav offre diverse opportunità di gestione e miglioramento aziendale. Siav è catalogata come *Software house* e presenta un ampio catalogo di prodotti atti a soddisfare le principali necessità organizzative di un'azienda, sta al potenziale cliente poi, valutarne l'acquisto in base alle proprie necessità.

1.6 Propensione all'innovazione

Siav è in costante crescita ed è in continua ricerca di nuove tecnologie da implementare all'interno dei propri prodotti. Presentano un ottimo reparto di Ricerca e Sviluppo nel quale sono costantemente impegnati in nuove tematiche da affrontare facedo sempre riferimento ai bisogni dei clienti. Essendo, quello dell'informatica, un settore in continua crescita è necessario che il team di Ricerca e Sviluppo operi costantemente al fine di trovare nuove risorse e tecnologie da affrontare e sperimentare. Una cospicua fetta di questi studi e approfondimenti vengono concretizzati tramite progetti sperimentali e attività di stage, in modo da poter osservare nel concreto se tali ricerche possano produrre risultati che giovino all'azienda.

Capitolo 2

L'attività di stage all'interno della strategia aziendale

Introduzione al capitolo
PROVA In questo capitolo andranno affrontati i processi e le motodologie affrontate per lo sviluppo del progetto. Più precisamente verranno descritti i principi cardine dell'*Agile programming* e successivamente verrà illustrato come questi siano stati applicati durante l'attività di stage.

Capitolo 3

Descrizione dello stage

In questo capitolo andrò a descrivere i principali aspetti che sono stati trattati durante lo stage

3.1 Introduzione al progetto

Visto in scala più ampia, il progetto finale risulterà una rivisitazione di Bipod, un applicativo di Siav, in ambito process mining, utilizzato come gestore di processi aziendali. Tale software, dopo averlo provato con mano, risulta funzionale in ogni sua parte, anche se la sua struttura risulta poco estendibilea. Per questo motivo mi è stato proposto di sviluppare una parte di software che si andrà poi ad integrare con il nuovo applicativo che adrà a rimpiazzare l'attuale esistente. Nello specifico le richieste a cui ho fatto fronte sono state le seguenti:

* Libreria di process mining per filtraggio su log degli eventi.
* Interfaccia fronted tenendo come punto di riferimento il vecchio applicativo.
* Stub per verificare il correntto comportamento dell'interfaccia di filtraggio

3.2 Studio tecnologico

3.2.1 Ambito process mining

Fin dal primo momento sono stato formato dal tutor tramite videolezioni e l'utilizzo di alcuni tool di process mining, cercando di entrare a fondo nel contensto dell'analisi dei processi. I software utilizzati sono stati i seguenti:

* Bipod : Software proprietario di *Siav* per la gestione ed il miglioramento di processi aziendali.
* ProM : Software di mining, modulare e flessibile sviluppato da *Eindhoven University of Technology*
* Disco: Software leader nel settore del process mining sviluppato da *Fluxicon*

3.2.2 Framework java per applicazione cloud

Un framework si definisce come un set di istruzioni pre-compilato, che fornisce funzionalità generiche e che possono essere adattate alle necessità degli utenti. L'utilizzo dei framework garantisce al programmatore un ingente risparmio di tempo e la certezza di utilizzare strumenti standard, dovuto ad un way of working solido che aderisce alle best practice. Nel mio caso tale nozione è stata fondamentale per lo sviluppo della libreria, garantendomi una buona solidità e mantanibilità del codice

3.2.3 OpenXes

OpenXes è una libreria di process mining aderente allo standard XES. In questa libreria i log sono visti come file ".xes". Il log è rappresentato da una classe denominata XLog, contente al suo interno una lista di tracce denominata XTrace, che rappresenta un singolo processo all'interno del log. Ogni XTrace contiene la lista di eventi che formano quel dato processo. La classe che rappresenta il singolo evento è denominata Xevent

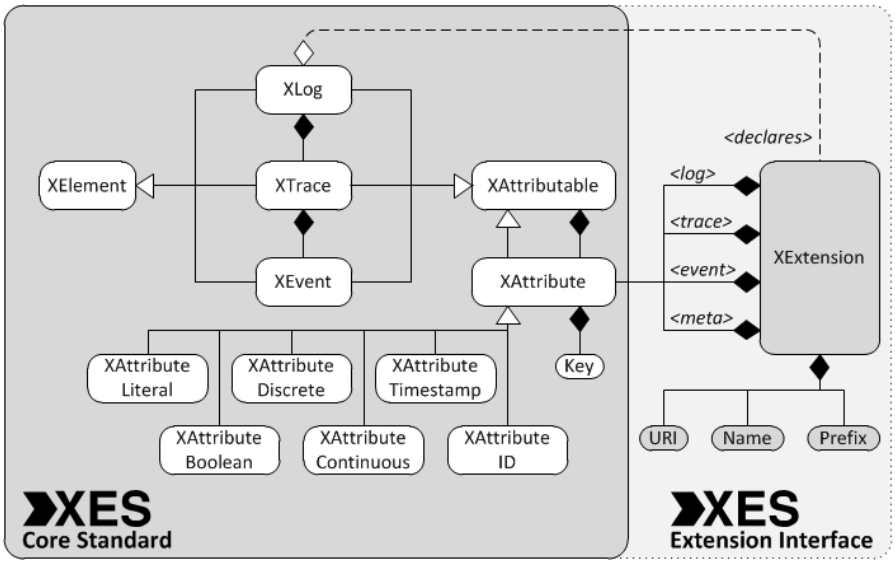


Figura 3.1: Diagramma dei package dello standard openXes

3.2.4 Docker

Docker è una piattaforma che automatizza il deployment di applicazioni all'interno di contenitori software tramite virtualizzazioni a livello di sistema operativo. Tali contenitori, aventi ognuno la propria logica ed implementazione, garantiscono funzionalità di isolamento rispetto agli altri. Lo studio di questa tecnologia non è stata approfondita a pieno in quanto, dopo la negoziazione dei requisiti, non vi è stata la necessità di entrare più nel dettaglio. È stata sufficente una infarinatura generale per poter comprendere l'architettura finale che presenterà il prodotto.

3.2.5 Web socket

Web Socket è una tecnologia che fornisce canali di comunicazione bidirezionali attraverso una singola connessione. Questi canali sono sempre "aperti" durante tutto il lifecycle dell'applicativo, effettuando richieste, e ricevento risposte in un unico canale. Tale tecnologia è stata concretizzata lato broswer avvalendosi della Liberia RxJS in modo da gestire sie le chiamate che gli eventi di ritorno. Per quanto riguarda il lato server, tramite l'implementazione di alcuni stub scritti in Python, è stato possibile gestire l'interfaccia in base ai messaggi ritorno inviati dallo stub.

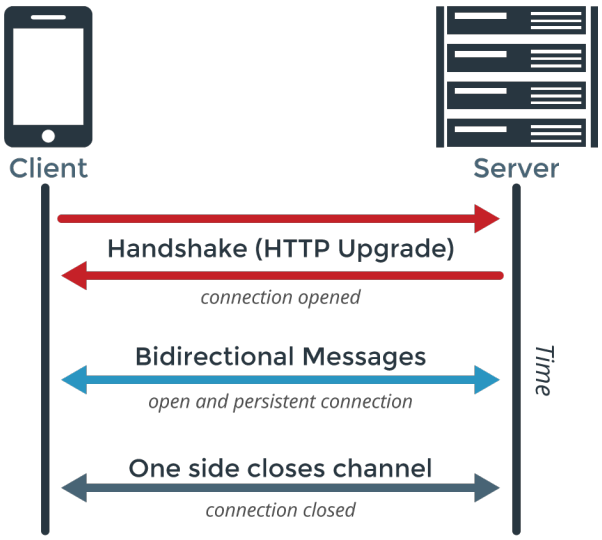


Figura 3.2: Ciclo di vita di una connessione tramite websocket

3.2.6 Angular

Angular è una piattaforma opensource per lo sviluppo di web application scritto in TypeScript. Una applicazione Angular può essere vista come un insieme di componenti visivi a sè stanti i quali occupano una porizione dell'intera applicazione. Ciascun componenete rappresenta un'entità configurabile e personalizzabile e può essere inserita all'interno di altri componenti.

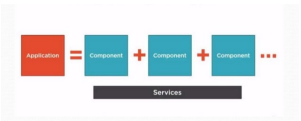


Figura 3.3: Logo di Angular e TypeScript

3.3 Analisi preventiva dei rischi

Durante la fase di analisi iniziale sono stati individuati alcuni possibili rischi a cui si potrà andare incontro. Innanzitutto mi sono documentato sulle tipologie di filtraggio

possibili dato un log degli eventi. Da qui, sotto consiglio del tutor, ho iniziato a studiare le librerie di process mining presenti all'interno del repository di ProM, un applicativo di process mining modulabile in cui è possibile l'integrazione con varie librerie. Tali librerie risultavano però scarsamente documentate, non aggiornate, e di difficile comprensione.

1. Reperimento package dal repository di ProM
Descrizione: All'interno del repository di ProM sono presenti centinaia di librerie dedicate al process mining, quest'ultime però risultano scarsamente documentate e non aggiornate.
Soluzione: la maggior parte delle librerie presenti fanno riferimento ad uno Standard denominato openXes, tale standard permette la rappresentazione delle principali strutture derivanti dai log degli eventi. Sono quindi partito da un progetto ex novo basandomi sullo standard sopracitato per la realizzazione della libreria.

2. Reperimento dettagli implementativi di filtraggio
Descrizione: Usando Bipod come riferimento ho discusso con il tutor in merito a tutte le possibili richieste di filtraggio che sarebbero state implementate. Ne è emerso che, sotto alcuni aspetti, alcune richieste saranno riviste e riscritte.
Soluzione: Mi sono avvalso del software Disco, uno dei più noti tool di process mining per poter studiare il comportamento di tutte le richieste filtraggio all'interno di un log degli eventi, stilando una lista dettagliata dei possibili metodi che sarebbero serviti per ricorprire tutti requisiti concordati con il tutor..

3.4 Requisiti e obiettivi

I requisiti all'interno del progetto sono stati tracciati utilizzando come riferimento il vecchio applicativo Bipod e il software Disco. Da qui è stato possibile tracciare tutte le tipologie di filtraggio richieste ed il loro funzionamento:

* Filtaggio per tempo : filtraggio di un log sulla base di attributi temporali
* Filtraggio per performance: filtraggio di un log sulla base della durata di un singolo evento, o sulla latenza tra due eventi
* Filtraggio per Attività di Inizio e Fine: filtraggio sulle tracce

3.5 Pianificazione

Capitolo 4

Analisi dei requisiti

Breve introduzione al capitolo

4.1 Casi d'uso

Per lo studio dei casi di utilizzo del prodotto sono stati creati dei diagrammi. I diagrammi dei casi d'uso (in inglese *Use Case Diagram*) sono diagrammi di tipo Unified Modeling Language (UML) dedicati alla descrizione delle funzioni o servizi offerti da un sistema, così come sono percepiti e utilizzati dagli attori che interagiscono col sistema stesso. Essendo il progetto finalizzato alla creazione di un tool per l'automazione di un processo, le interazioni da parte dell'utilizzatore devono essere ovviamente ridotte allo stretto necessario. Per questo motivo i diagrammi d'uso risultano semplici e in numero ridotto.

UC0: Scenario principale

Attori Principali: Sviluppatore applicativi.

Precondizioni: Lo sviluppatore è entrato nel plug-in di simulazione all'interno dell'IDE.

Descrizione: La finestra di simulazione mette a disposizione i comandi per configurare, registrare o eseguire un test.

Postcondizioni: Il sistema è pronto per permettere una nuova interazione.

4.2 Tracciamento dei requisiti

Da un'attenta analisi dei requisiti e degli use case effettuata sul progetto è stata stilata la tabella che traccia i requisiti in rapporto agli use case.
Sono stati individuati diversi tipi di requisiti e si è quindi fatto utilizzo di un codice identificativo per distinguerli.
Il codice dei requisiti è così strutturato R(F/Q/V)(N/D/O) dove:

R = requisito

F = funzionale

Q = qualitativo

V = di vincolo

N = obbligatorio (necessario)

D = desiderabile

Z = opzionale

Nelle tabelle 4.1, 4.2 e 4.3 sono riassunti i requisiti e il loro tracciamento con gli use case delineati in fase di analisi.

Tabella 4.1: Tabella del tracciamento dei requisti funzionali

Requisito	Descrizione	Use Case
RFN-1	L'interfaccia permette di configurare il tipo di sonde del test	UC1

Tabella 4.2: Tabella del tracciamento dei requisiti qualitativi

Requisito	Descrizione	Use Case
RQD-1	Le prestazioni del simulatore hardware deve garantire la giusta esecuzione dei test e non la generazione di falsi negativi	-

Tabella 4.3: Tabella del tracciamento dei requisiti di vincolo

Requisito	Descrizione	Use Case
RVO-1	La libreria per l'esecuzione dei test automatici deve essere riutilizzabile	-

Capitolo 5

Progettazione e codifica

Breve introduzione al capitolo

5.1 Tecnologie e strumenti

Di seguito viene data una panoramica delle tecnologie e strumenti utilizzati.

Tecnologia 1

Descrizione Tecnologia 1.

Tecnologia 2

Descrizione Tecnologia 2

5.2 Ciclo di vita del software

5.3 Progettazione

Namespace 1

Descrizione namespace 1.

Classe 1: Descrizione classe 1

Classe 2: Descrizione classe 2

5.4 Design Pattern utilizzati

5.5 Codifica

Capitolo 6

Verifica e validazione

Capitolo 7

Conclusioni

7.1 Consuntivo finale

7.2 Raggiungimento degli obiettivi

7.3 Conoscenze acquisite

7.4 Valutazione personale

Glossario

PROVA in ingegneria RPVAO del software *PROVA, Unified Modeling Language* (ing. linguaggio di modellazione unificato) è un linguaggio di modellazione e specifica basato sul paradigma object-oriented. L'*PROVA* svolge un'importantissima funzione di "lingua franca" nella comunità della progettazione e programmazione a oggetti. Gran parte della letteratura di settore usa tale linguaggio per descrivere soluzioni analitiche e progettuali in modo sintetico e comprensibile a un vasto pubblico. 1

UML in ingegneria del software *UML, Unified Modeling Language* (ing. linguaggio di modellazione unificato) è un linguaggio di modellazione e specifica basato sul paradigma object-oriented. L'*UML* svolge un'importantissima funzione di "lingua franca" nella comunità della progettazione e programmazione a oggetti. Gran parte della letteratura di settore usa tale linguaggio per descrivere soluzioni analitiche e progettuali in modo sintetico e comprensibile a un vasto pubblico. 27

Acronimi

UML Unified Modeling Language. 13, 25

Bibliografia