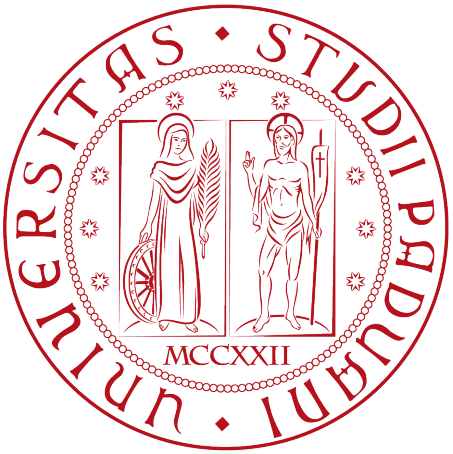
**Università degli Studi di Padova**

Dipartimento di Matematica

Corso di Laurea in Informatica




# Project Tango e PCL: un'applicazione pratica alle 'Nuvole di Punti'

*Tesi di laurea triennale*

*Relatore*

Prof. Ombretta Gaggi

*Laureando*

Davide Bortot

Anno Accademico 2015-2016

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, della durata di circa trecentoventi ore, dal laureando Davide Bortot presso l'azienda VIC S.r.l. Obbiettivo del tirocinio era lo sviluppo di un'applicazione prototipale per il tablet Project Tango per l'aquisizione di scansioni di oggetti reali sotto forma di 'Nuvola di Punti'.
In primo luogo era richiesto il filtraggio e l'elaborazione dei dati acquisiti in modo da isolare l'oggetto scansionato, e successivamente di creare una mesh tridimensionale a partire dai punti rimanenti. Scopo finale era calcolare il volume dell'oggetto tridimensionale così ottenuto.

*"Life is really simple, but we insist on making it complicated"*

— Confucius

# Ringraziamenti

*Innanzitutto, vorrei esprimere la mia gratitudine alla Prof.ssa Ombretta Gaggi, relatore della mia tesi, per l'aiuto e il sostegno fornitomi durante la stesura del lavoro.*

*In secondo luogo, ringrazio Michele Marchetto, tutor aziendale, così come tutti i colleghi Tommaso Padovan, Lorenzo Ceccon e Luca D'Ambros per aver reso lo stage un'esperienza di crescita.*

*Ringrazio col cuore i miei genitori Giulia e Carlo per aver creduto in me in ogni momento e avermi sempre sostenuto.*

*Infine ringrazio i miei amici di sempre e gli amici nuovi che ho incontrato nel mio percorso universitario, per avermi reso quello che sono.*

*Padova, Oct 2016* Davide Bortot

# Indice


**1 Introduzione** **1**
- 1.1 L'azienda . . . 1
- 1.2 L'idea . . . 1
- 1.3 Cos'è Project Tango . . . 2
- 1.4 Il Prodotto - lato client . . . 3
  - 1.4.1 Primo prototipo: Cloude . . . 3
  - 1.4.2 Secondo prototipo: Samba . . . 4
  - 1.4.3 L'applicativo attuale: VIC-Tango . . . 6
- 1.5 Il Prodotto - lato server . . . 6
- 1.6 Organizzazione del testo . . . 6

**2 Processi, tecnologie e strumenti** **9**
- 2.1 Processo sviluppo prodotto . . . 9
  - 2.1.1 Kick-Off . . . 9
  - 2.1.2 Concept Preview . . . 9
  - 2.1.3 Product Prototype . . . 9
  - 2.1.4 Fasi successive . . . 10
- 2.2 Tecnologie e Strumenti . . . 10
  - 2.2.1 Codice . . . 10
  - 2.2.2 IDE ed editor . . . 11
  - 2.2.3 Framework . . . 11

**3 Studio di fattibilità ed analisi dei rischi** **13**
- 3.1 Introduzione al progetto . . . 13
- 3.2 Analisi preventiva dei rischi . . . 13
- 3.3 Requisiti e obiettivi . . . 13
- 3.4 Pianificazione . . . 13

**4 Analisi dei requisiti** **15**
- 4.1 Casi d'uso . . . 15
- 4.2 Tracciamento dei requisiti . . . 16

**5 Progettazione e codifica** **19**
- 5.1 Tecnologie e strumenti . . . 19
- 5.2 Ciclo di vita del software . . . 19
- 5.3 Progettazione . . . 19
- 5.4 Design Pattern utilizzati . . . 19
- 5.5 Codifica . . . 19

**6 Verifica e validazione** **21**

**7 Conclusioni** **23**
7.1 Consuntivo finale . . . . . 23
7.2 Raggiungimento degli obiettivi . . . . . 23
7.3 Conoscenze acquisite . . . . . 23
7.4 Valutazione personale . . . . . 23

**A Appendice A** **25**

**Bibliografia** **29**

# Elenco delle figure

1.1 L' *hardware* di *Project Tango* . . . . . 2
1.2 Esempio di *Point Cloud*: un cestino cilindrico . . . . . 3
1.3 *Drifting* nel *Motion Tracking* . . . . . 4
1.4 Benefici della *Drift Correction* . . . . . 5
1.5 Effetti del *voxeling* . . . . . 6

4.1 Use Case - UC0: Scenario principale . . . . . 15

# Elenco delle tabelle

4.1 Tabella del tracciamento dei requisti funzionali . . . . . 17
4.2 Tabella del tracciamento dei requisiti qualitativi . . . . . 17
4.3 Tabella del tracciamento dei requisiti di vincolo . . . . . 17

# Capitolo 1

# Introduzione

## 1.1 L'azienda

VIC è stata fondata da Alessio Bisutti che, dopo aver sviluppato una lunga esperienza nel campo ispettivo, ha deciso di costituire una società in grado di offrire ai propri clienti un servizio professionale, chiaro ed affidabile, appoggiandosi alle nuove tecnologie. Si occupa di controlli per grandi ordini, sia di materie prime che di semi-lavorati, di cui individua e riporta eventuali danni, carenze nella spedizione e non conformità con quanto ordinato. VIC viene fondata a Venezia 7 anni fa come piccola società di ispezione locale. Fin dall'inizio, l'obiettivo principale di VIC è stato la riduzione del tempo tra ispezione e reporting al cliente. Ora l'obiettivo è raggiunto, perchè VIC sta fornendo ai suoi clienti tutti i risultati e le informazioni importanti in tempo reale, senza alcun ritardo, grazie agli investimenti fatti nel campo della tecnologia e delle applicazioni mobili.

## 1.2 L'idea

I più importanti obbiettivi del controllo qualità effettuato durante un ispezione sono il determinare la corretta forma, peso, quantità e dimensioni degli oggetti da esaminare. Gli ispettori possono scattare fotografie, prendere appunti e sfruttare la loro esperienza per fornire stime accurate; si è manifestata però la necessità di affiancare queste ultime a dei dati quanto più possibile oggettivi e rapidi da ottenere.
Da qui nasce l'idea di fornire agli ispettori uno strumento informatico in grado di effettuare queste stime. Grazie alla ricostruzione computerizzata resa disponibile dai *Tango device* sarà possibile non solo visualizzare su uno schermo il modello 3D del soggetto della ispezione, ma anche ottenere ulteriori dati utili quali:

* Una stima del volume, e se necessario del peso, dell'oggetto.
* L'esito del confronto dell'oggetto con un modello ideale, per evidenziare eventuali danni o deformazioni.

Con il prototipo realizzato durante lo *stage* sono rese disponibili solamente le funzionalità di ricostruzione dell'oggetto e calcolo (approssimato) del volume.
Le operazioni troppo computazionalmente intensive da effettuare su tablet, quali il filtraggio e *meshing* dei punti acquisiti, sono state delegate ad un *backend server* che effettui le elaborazioni necessarie ed invii i risultati al richiedente, mentre all'applicativo

per tablet è stato affidato il compito di aquisire e visualizzare un oggetto sotto forma di 'Nuvola di Punti' e poterne esaminare la *mesh* ottenuta.

## 1.3 Cos'è Project Tango

Project Tango è un *tablet* sperimentale prodotto da *Google* in grado di scandire tridimensionalmente l'ambiente circostante e di tracciare la propria posizione rispetto ad esso. Ciò è possibile attraverso il ricco hardware di cui è dotato (si veda la figura 1.1), tra cui:

* una fotocamera RBG-IR
* una fotocamera *Fisheye*
* un sensore di profondità IR
* accelerometro e giroscopio

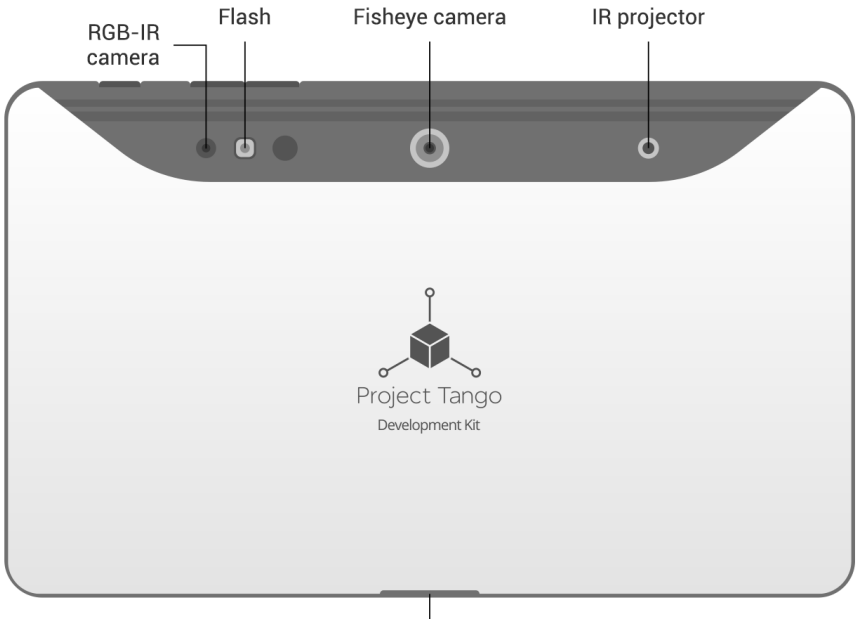



**figura 1.1:** L' *hardware* di *Project Tango*

Project Tango è quindi specificamente pensato per sviluppare applicazioni che necessitano di comprendere ed estrapolare informazioni dal mondo reale (ad es. realtà aumentata). Le funzionalità del *tablet* sono accessibili attraverso le *Tango API*, le *API* ufficiali per lo sviluppo di applicazioni *Tango*.

## 1.4 Il Prodotto - lato client

L'applicazione su *tablet* prodotta realizza, seppur non in maniera completa, le esigenze citate al punto precedente.
La sua realizzazione ha incontrato molte problematiche talvolta critiche e difficili da prevedere. Per questo durante lo sviluppo sono stati implementati più prototipi, al fine di esplorare le potenzialità e soprattutto i limiti del *tablet* e delle *Tango API*.
Lo scopo principale dell'applicazione lato *tablet* è quello di rilevare una corretta 'Nuvola di Punti' dell'oggetto che si vuole esaminare.
Una 'Nuvola di Punti' è una descrizione matematica di un oggetto tridimensionale ottenuta tramite un insieme, il più possibile fitto, di punti che lo compongono, definiti dalle loro coordinate (**x**,**y**,**z**) rispetto ad un fissato sistema di riferimento.
Tale rappresentazione, riferita spesso d'ora in poi con il più elegante termine inglese *Point Cloud*, è facilmente comprensibile all'utente se visualizzata come in figura 1.2.

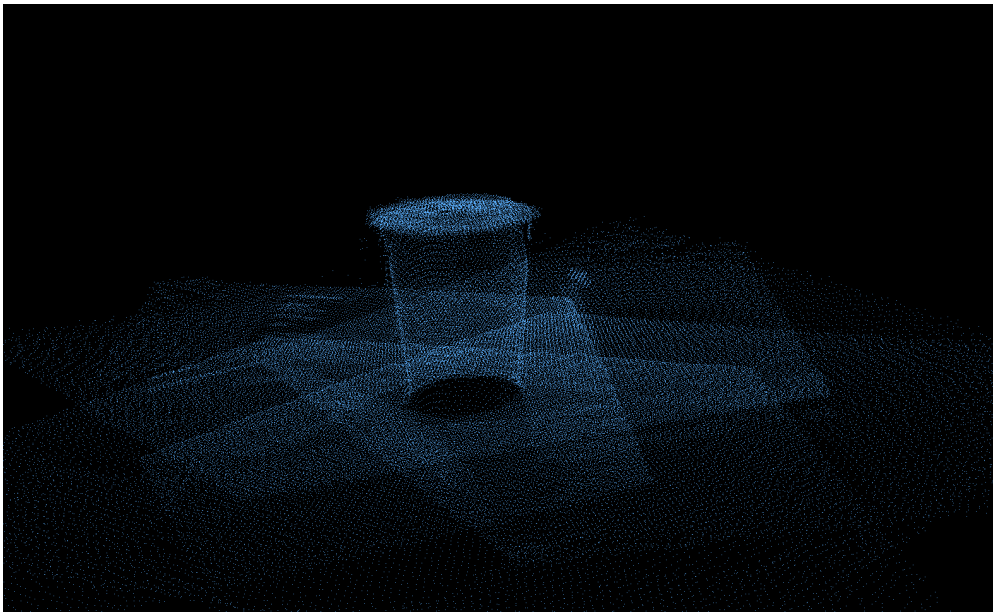


**figura 1.2:** Esempio di *Point Cloud*: un cestino cilindrico

### 1.4.1 Primo prototipo: Cloude

Il prototipo denominato *Cloude* risponde all'esigenza di catturare più *Point Cloud* da più angolazioni in modo da ricostruire l'oggetto scansionato sovrapponendo i dati acquisiti. *Cloude* è il risultato di più prototipazioni precedenti effettuate dallo stagista che ha iniziato il progetto prima di me e con cui ho strettamente collaborato.
Il prototipo permette quindi di acquisire più *Point Cloud* in successione, ognuno dei quali verrà adeguatamente ruotato e traslato rispetto allo spostamento del *tablet*, in modo che la sovrapposizione degli stessi vada infine a ricostruire tutte le parti dell'oggetto esaminato.
Una sola acquisizione da una certa angolazione non può infatti comprendere un oggetto nella sua interezza, come dimostrato nelle figure //TODO è necessario ricostruire passo per passo un unico *Point Cloud* rappresentante l'oggetto completo. Sulla base

dei risultati di *Cloude* ho iniziato il mio lavoro sulla parte di elaborazione lato *server* dei *Point Cloud* acquisiti; questo è stato inoltre il punto di inizio per lo sviluppo dei prototipi successivi in collaborazione con l'altro stagista.

### 1.4.2 Secondo prototipo: Samba

Il secondo prototipo parte dai risultati ottenuti con *Cloude* e ne migliora prestazioni e precisione, aggiungendo alcune funzionalità utili.
*Samba* affronta e risolve alcuni problemi del prototipo precedente, come il *"Drifting"* e alcuni problemi prestazionali dovuta alla mole elevata di dati acquisiti. Introduce inoltre la funzionalità di visualizzazione delle *mesh* elaborate dal *server*.

#### Drift correction

Il prototipo precedente, seppur funzionale, generava ricostruzioni di scarsa qualità, afflitte soprattutto dal problema del *"drifting"*. Il *drifting* è un problema comune nelle applicazioni di realtà aumentata, che come *Project Tango* usano la tecnica del *Motion Tracking*, cioè aggiornano costantemente la propria posizione relativamente alle coordinate acquisite nella posizione precedente, mantenendo così una storia dei movimenti del *device* rispetto all'ambiente circostante.
Ad ogni aggiornamento della posizione è normale ed inevitabile che la misurazione, per quanto precisa, introduca un piccolo errore; la catena di errori sommati porta quindi col passare del tempo ad un'importante discrepanza tra la posizione stimata del *device* e la sua posizione reale, come evidenziato in figura 1.3 Per ovviare in parte a questo

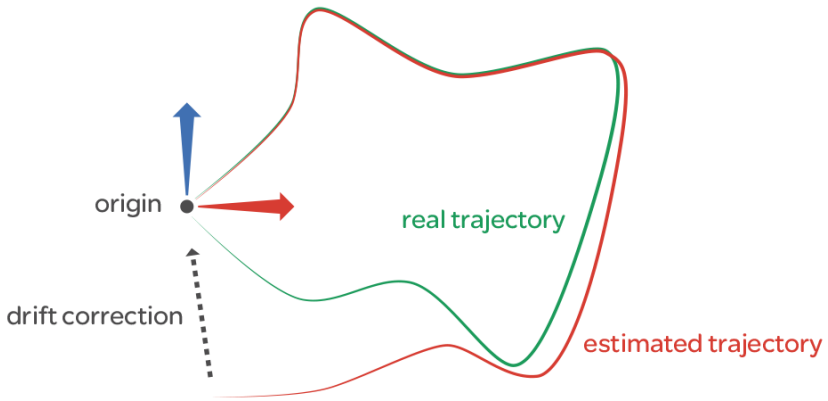
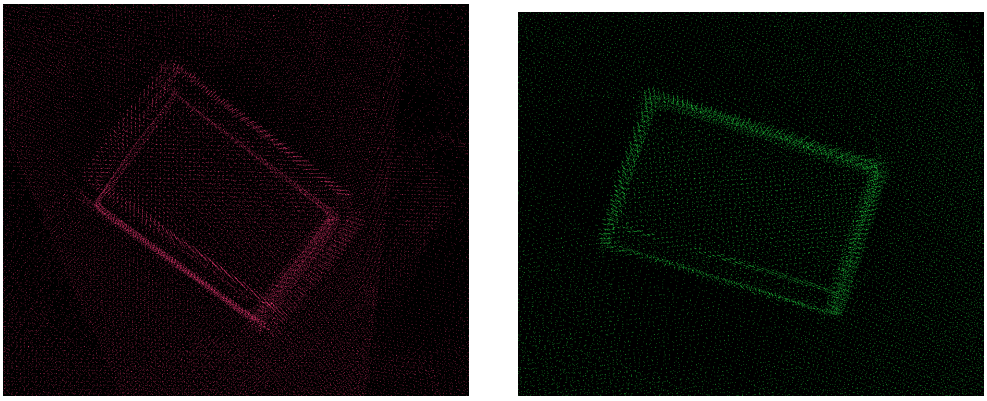



**figura 1.3:** *Drifting* nel *Motion Tracking*

problema *Samba* utilizza una tecnica chiamata *Drift Correction* ch si appoggia sulle funzionalità di *Area Learning* dei *device Tango*.
L' *Area Learning* consiste nella capacita del dispositivo di estrarre dallo spazio fisico che sta analizzando una serie di punti significativi (o *key features*), facilmente riconoscibili, e di salvare tali informazioni per confrontarle con le successive acquisizioni. In questo modo il dispositivo è capace di riconoscere un'area precedentemente visitata, e può quindi applicare le necessarie correzioni alla popria stima della traiettoria, di qui il nome *Drift Correction*.

Con l'implementazione di questa funzionalità è necessario, prima di iniziare la ricostruzione di un *Point Cloud*, riprendere l'oggetto e i suoi dintorni per un po' di tempo e da diverse angolazioni, in modo da permettere al *tablet Tango* di costruire una mappa, detta *Area description*, dell'area circostante, e di stabilizzare la traiettoria stimata con le informazioni acquisite.
I risultati si vedono confrontando queste due ricostruzioni di una scatola, vista dall'alto (fig. 1.4a e fig. 1.4b), rispettivamente senza e con *Drift correction*.



**(a)** Ricostruzione senza *Drift correction* **(b)** Ricostruzione con *Drift correction*

**figura 1.4:** Benefici della *Drift Correction*

Come si nota in figura 1.4b il risultato non è ancora ottimale, infatti un lato della scatola appare sdoppiato e spostato di qualche centimetro; si tratta di un problema di *ghosting* di cui verrà trattato più avanti.

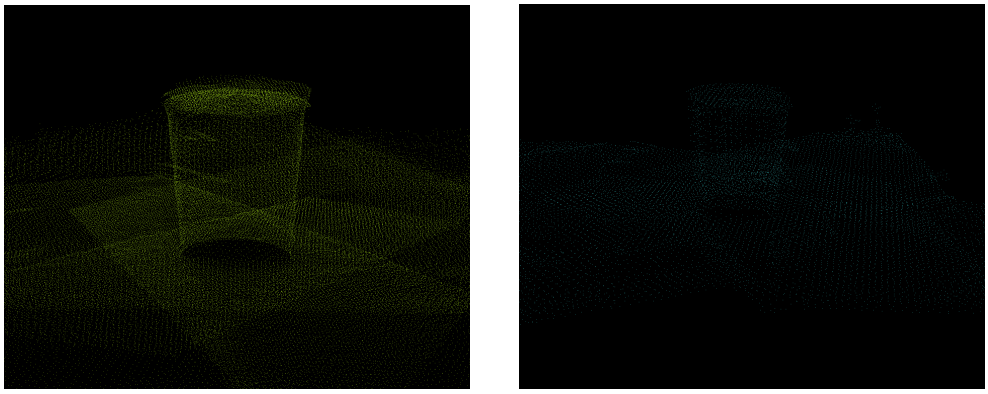
### Prestazioni

Il prototipo precedente presentava due problemi prestazionali principali:

* La ricostruzione passo per passo del *Point Cloud* finale era troppo lenta
* La mole dei dati trattati al sovrapporsi di più *Point Cloud* diventava proibitiva

Il primo problema si presentava utilizzando i metodi forniti dalla libreria *Tango* per trasformare le coordinate relative dei punti in coordinate assolute per permetterne la giusta sovrapposizione. Tale metodo, seppur di facile utilizzo, diventava troppo dispendioso all'aumentare delle dimensioni del *Point Cloud*, che raggiunge facilmente gli 80.000 punti.
Per risolvere il problema viene quindi creata, per ogni *Point Cloud*, una *matrice di rototraslazione* che rappresenta lo spostamento e la rotazione del dispositivo rispetto al sistema di riferimento, e viene moltiplicato il vettore delle coordinate di ogni singolo punto per la matrice generata.
In questo modo si sono ridotti i tempi di elaborazione dell'80%;
Il secondo problema era causato dal sovrapporsi di molti punti quasi identici, ad esempio i punti rappresentanti il pavimento. Partendo dall'osservazione che una certa quantità di punti molto vicini può essere trasformata in un unico punto, valore medio di tutti gli altri, senza una significativa perdita d'informazione, *Samba* risolve il problema attraverso una tecnica di *voxeling*.

Lo spazio viene suddiviso in tanti piccoli parallelepipedi (tipicamente cubi) di uguali dimensioni; tutti i punti che ricadono all'interno di un singolo parallelepipedo, o *voxel*, vengono considerati come un unico punto. Così facendo si riducono sensibilmente le dimensioni del *Point Cloud* senza alterarne negativamente la precisione. Di seguito la differenza visivamente evidente tra lo stesso Point Cloud filtrato prima con voxel cubici di lato 1cm (fig. 1.5a) e poi di lato 3cm (fig. 1.5b), con una differenza di circa 60.000 punti rimossi in più nella seconda elaborazione.



**(a)** Voxeling leggero **(b)** Voxeling elevato

**figura 1.5:** Effetti del *voxeling*

#### Visualizzatore di mesh

Nel prototipo è stata introdotta la possibilità di caricare e visualizzare le *mesh* risultato dell'elaborazione lato server di un *Point Cloud*. Una *mesh* poligonale è una collezione di vertici, spigoli e facce che definiscono la forma di un oggetto poliedrico; in *Samba* è ottenuta a partire dalla nuvola di punti elaborata. Tale *mesh* viene salvata dal *server* in formato *OBJ*, comunemente usato nelle applicazioni 3D. Dall'applicazione viene quindi richiesto di caricare e salvare nella memoria locale del tablet le mesh disponibili sul server, per poi poter essere visualizzate ed esaminate.

### 1.4.3 L'applicativo attuale: VIC-Tango

#### Firebase

#### JNI

#### Camera preview

## 1.5 Il Prodotto - lato server

## 1.6 Organizzazione del testo

Riguardo la stesura del testo, relativamente al documento sono state adottate le seguenti convenzioni tipografiche:

- gli acronimi, le abbreviazioni e i termini ambigui o di uso non comune menzionati vengono definiti nel glossario, situato alla fine del presente documento;
- per la prima occorrenza dei termini riportati nel glossario viene utilizzata la seguente nomenclatura: *parola*[g];
- i termini in lingua straniera o facenti parti del gergo tecnico sono evidenziati con il carattere *corsivo*.

# Capitolo 2

# Processi, tecnologie e strumenti

*Brevissima introduzione al capitolo*

## 2.1 Processo sviluppo prodotto

In ambito aziendale si è scelto di provare, per questo progetto, un processo di sviluppo *software* basato sulla filosofia *Lean*.
Dato che l'obiettivo principale era fornire un prototipo ci si è limitati solamente alle tre fasi iniziali dello sviluppo di *Lean*, ovvero *Kick-Off*, *Concept Preview* e *Product Prototype*.

### 2.1.1 Kick-Off

Questa è la prima fase dello sviluppo *software*, coincide con la prima riunione ufficiale del team di progetto, aperta anche agli *Stakeholder*.
Si pone lo scopo di iniziare la fase l'*allestimento* e l'*avviamento* in cui viene determinata la natura e lo scopo del progetto.

### 2.1.2 Concept Preview

Fase in cui è reso disponibile un primo campione di prova del prodotto, detto *concept*. Esso può essere incompleto e affetto da errori, ma deve essere in grado di dimostrare agli *stakeholder* le caratteristiche principali che avrà il prodotto finito.
Esso è soggetto a un riesame che ha lo scopo di valutare se è in linea con gli obiettivi definiti nella *Value Proposition*, la *milestone* non può essere raggiunta senza che questo riesame abbia esito positivo.
Questa fase coincide anche con l'inizio della progettazione, che deve essere portata avanti fino ad un livello di dettaglio ritenuto opportuno dal *team*.

### 2.1.3 Product Prototype

Fase in cui è messo a disposizione il primo prototipo del nuovo prodotto, completo nelle sue funzioni (sviluppo finito) ma non ancora messo a punto mediante verifiche

e correzioni, per garantirne funzionalità e prestazioni. Il prototipo deve essere ad un stato tale da poter essere dato in valutazione agli *stakeholder*.
Esso è soggetto a un riesame che ha lo scopo di valutare se è in linea con gli obiettivi definiti sia *Value Proposition* che nella *Requirements Specification*, la *milestone* non può essere raggiunta senza che questo riesame abbia esito positivo.
Questa fase coincide con il termine della fase di progettazione e l'inizio della fase di esecuzione, cioè l'insieme dei processi necessari a soddisfare i requisiti del progetto.

### 2.1.4 Fasi successive

Le fasi successive, ovvero *Product Design Freeze* e *Start Of Production*, possono essere avviate nel futuro a partire dal *Product Prototype* se ciò verrà ritenuto opportuno dall'azienda.

## 2.2 Tecnologie e Strumenti

L'azienda ha lasciato grande libertà riguardo agli strumenti da utilizzare per questo progetto, quindi essi sono stati fissati inizialmente e successivamente incrementati al crescere delle necessità.

### 2.2.1 Codice

Segue la lista degli strumenti utilizzati per la codifica.

- **Java**[1]: Il linguaggio preferito per l'applicazione lato *tablet*. È stato scelto seguendo le *Best Practice* dello sviluppo *Android*.
- **C++**[2]: Il linguaggio preferito per l'applicazione lato *Server*. È stato scelto perché tutti le più diffuse librerie per l'elaborazione dei *Point Cloud*, ed in particolare *PCL* sono disponibili in questo linguaggio.
- **PCL, Point Cloud Library**[3]: Una delle librerie di maggior rilievo nel campo della *Computer Vision*, mette a disposizione notevoli funzionalità per l'elaborazione, il filtraggio e l'ottimizzazione dei *Point Cloud*.
- **Tango API**[4]: Le *API* ufficiali per lo sviluppo di applicazioni *Tango*.
- **PHP**[5]: Il linguaggio usato per ricevere ed inviare le richieste *HTTP* necessarie alla comunicazione tra *Server* e dispositivo.
- **Python**[6]: Il linguaggio usato in combinazione con *PHP* per implementare la logica del lato *Server*.

[1] https://www.java.com
[2] www.cplusplus.com
[3] http://pointclouds.org
[4] https://developers.google.com/tango/apis/overview
[5] https://secure.php.net
[6] https://www.python.org

### 2.2.2 IDE ed editor

Segue la lista degli ambienti per la codifica utilizzati durante il progetto.

* **Android Studio**[7]: L'*IDE* ufficiale per le applicazioni *Android*.
* **QT**[8]: L'*IDE* scelto per lo sviluppo del codice *C++*.
* **Geany**[9]: L'*editor* di testo usato per scrivere gli *script php* e *Python*.

### 2.2.3 Framework

Segue la lista dei *Framework* usati per durante il tirocinio.

* **Gradle**[10]: È stato usato come *tool* di *build* per tutta l'applicazione *Android*.
* **Rajawali3D**[11]: È stato usato come *framework* grafico per la realizzazione del *render*.
* **OkHttp**[12]: È stata usato come *framework* di riferimento per le richieste *http*, come indicato nella *Android Best Practices*.
* **TangoUx**[13]: *Famework* messo a disposizione dalla *Google* assieme alle *API Tango*. È stato usato per gestire le notifiche all'utente relative ai comportamenti che deve tenere per permettere il buon funzionamento del dispositivo e dei sensori.
* **Jni**[14]: Questo *framework* permette di richiamare metodi 'nativi' (scritti in *C/C++*) dal codice *Java*.
* **Firebase**[15]: *Framework* associato all'omonimo *web service* offerto da *Google* per lo scambio di messaggi tra applicazioni *Android* e un *server*.

[7] https://developer.android.com/studio/index.html
[8] https://www.qt.io/ide
[9] https://www.geany.org
[10] https://gradle.org
[11] https://github.com/Rajawali/Rajawali
[12] http://square.github.io/okhttp
[13] https://developers.google.com/tango/ux/ux-framework
[14] http://docs.oracle.com/javase/7/docs/technotes/guides/jni
[15] https://firebase.google.com

# Capitolo 3

# Studio di fattibilità ed analisi dei rischi

*Breve introduzione al capitolo*

## 3.1 Introduzione al progetto

## 3.2 Analisi preventiva dei rischi

Durante la fase di analisi iniziale sono stati individuati alcuni possibili rischi a cui si potrà andare incontro. Si è quindi proceduto a elaborare delle possibili soluzioni per far fronte a tali rischi.

**1. Performance del simulatore hardware**
**Descrizione:** le performance del simulatore hardware e la comunicazione con questo potrebbero risultare lenti o non abbastanza buoni da causare il fallimento dei test.
**Soluzione:** coinvolgimento del responsabile a capo del progetto relativo il simulatore hardware.

## 3.3 Requisiti e obiettivi

## 3.4 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.

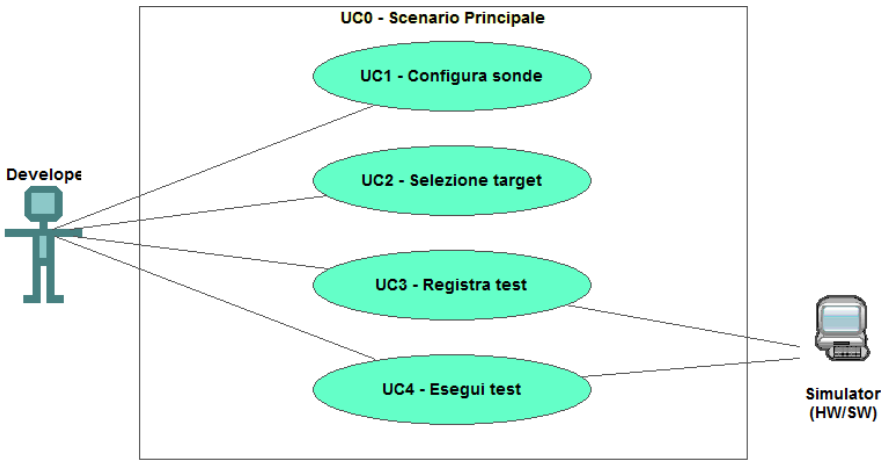



**figura 4.1:** Use Case - UC0: Scenario principale

### 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

# Appendice A

# Appendice A

Citazione

Autore della citazione

# Bibliografia

## Riferimenti bibliografici

James P. Womack, Daniel T. Jones. *Lean Thinking, Second Editon*. Simon & Schuster, Inc., 2010.

## Siti Web consultati

*Manifesto Agile*. URL: http://agilemanifesto.org/iso/it/.