

Università degli Studi di Padova

DIPARTIMENTO DI MATEMATICA "TULLIO LEVI-CIVITA"

CORSO DI LAUREA IN INFORMATICA



**Riconoscimento del linguaggio naturale per  
assistenti virtuali: interpretazione e  
apprendimento di comandi vocali**

*Tesi di laurea triennale*

*Relatore*

Prof. Paolo Baldan

*Laureando*

Massimo Toffoletto

---

ANNO ACCADEMICO 2019-2020

Massimo Toffoletto: *Riconoscimento del linguaggio naturale per assistenti virtuali: interpretazione e apprendimento di comandi vocali*, Tesi di laurea triennale, © Luglio 2020.

# Sommario

Il presente documento descrive il lavoro svolto durante il periodo di stage, della durata di 320 ore, dal laureando Massimo Toffoletto presso l'azienda Zucchetti S.p.A. Gli obiettivi da raggiungere sono stati molteplici e suddivisi in due parti.

Nella prima parte ho studiato il funzionamento dei tre principali assistenti virtuali nel seguente ordine: Assistant, Alexa e Siri. Gli obiettivi sono stati quindi l'analisi dei singoli assistenti e lo svolgimento di una comparazione tra gli stessi, sia delle funzionalità offerte agli sviluppatori che delle capacità riconosciute del linguaggio naturale. A questo proposito ho realizzato un proof of concept dimostrativo per ciascuno di essi.

Nella seconda parte gli obiettivi sono stati lo studio di regole per la realizzazione di grammatiche che interpretano il linguaggio naturale, implementate da Zucchetti ma ancora in via di sviluppo, e la costruzione di un'applicazione che le utilizzi. Infine ho esplorato ed implementato un'interfaccia vocale con un modo conversazionale di interagire che si integri con le regole già realizzate da Zucchetti.



*“Il computer non è una macchina intelligente che aiuta le persone stupide, anzi, è una macchina stupida che funziona solo nelle mani delle persone intelligenti.”*

— Umberto Eco

## Ringraziamenti

*Innanzitutto, vorrei esprimere la mia gratitudine al Prof. Paolo Baldan, relatore della mia tesi, per l'aiuto e il sostegno che mi ha fornito durante lo svolgimento del lavoro.*

*Desidero ringraziare con affetto tutta la mia famiglia per il loro sostegno e per essere stati sempre presenti durante gli anni di studio.*

*Voglio ringraziare inoltre i miei amici per gli anni meravigliosi passati assieme ed in particolare Alberto, Lorenzo e Fiammetta che mi hanno sempre sostenuto nei momenti più difficili ma anche in quelli più felici.*

*Padova, Luglio 2020*

Massimo Toffoletto



# Indice

<b>1</b>	<b>Introduzione</b>	<b>1</b>
1.1	Zucchetti S.p.A . . . . .	1
1.2	Lo stage proposto . . . . .	1
1.3	Organizzazione del testo . . . . .	2
<b>2</b>	<b>Lo stage</b>	<b>3</b>
2.1	Descrizione del progetto . . . . .	3
2.2	Obiettivi . . . . .	4
2.2.1	Classificazione . . . . .	4
2.2.2	Obiettivi obbligatori . . . . .	4
2.2.3	Obiettivi desiderabili . . . . .	5
2.2.4	Obiettivi facoltativi . . . . .	5
2.3	Pianificazione delle attività . . . . .	5
2.4	Tecnologie e strumenti utilizzati . . . . .	8
2.4.1	Tecnologie . . . . .	8
2.4.2	Strumenti . . . . .	8
2.5	Motivazioni personali . . . . .	9
<b>3</b>	<b>Gli assistenti virtuali</b>	<b>11</b>
3.1	Premessa . . . . .	11
3.2	Assistant . . . . .	11
3.2.1	Introduzione . . . . .	11
3.2.2	Casi d'uso . . . . .	12
3.2.3	Conversational Actions . . . . .	12
3.2.4	Content Actions . . . . .	16
3.2.5	App Actions . . . . .	16
3.2.6	Proof of concept . . . . .	16
3.3	Alexa . . . . .	16
3.3.1	Introduzione . . . . .	16
3.3.2	Casi d'uso . . . . .	16
3.3.3	Skill di conversazione . . . . .	16
3.3.4	Proof of concept . . . . .	17
3.4	Siri . . . . .	17
3.4.1	Introduzione . . . . .	17
3.4.2	Casi d'uso . . . . .	17
3.4.3	Comandi di conversazione . . . . .	17
3.4.4	Proof of concept . . . . .	18
3.5	Analisi comparativa . . . . .	18

<b>4</b>	<b>Analisi dei requisiti</b>	<b>19</b>
4.1	Casi d'uso . . . . .	19
4.2	Tracciamento dei requisiti . . . . .	20
<b>5</b>	<b>Progettazione e codifica</b>	<b>23</b>
5.1	Tecnologie e strumenti . . . . .	23
5.2	Ciclo di vita del software . . . . .	23
5.3	Progettazione . . . . .	23
5.4	Design Pattern utilizzati . . . . .	23
5.5	Codifica . . . . .	23
<b>6</b>	<b>Conclusioni</b>	<b>25</b>
6.1	Consuntivo finale . . . . .	25
6.2	Raggiungimento degli obiettivi . . . . .	25
6.3	Conoscenze acquisite . . . . .	25
6.4	Valutazione personale . . . . .	25
<b>A</b>	<b>Appendice A</b>	<b>27</b>
	<b>Bibliografia</b>	<b>31</b>



# Elenco delle figure

4.1	Use Case - UC0: Scenario principale . . . . .	19
-----	---	----

# Elenco delle tabelle

4.1	Tabella del tracciamento dei requisiti funzionali . . . . .	21
4.2	Tabella del tracciamento dei requisiti qualitativi . . . . .	21
4.3	Tabella del tracciamento dei requisiti di vincolo . . . . .	21



# Capitolo 1

## Introduzione

### 1.1 Zucchetti S.p.A

Zucchetti è un'azienda italiana fondata più di 40 anni fa che produce soluzioni software, hardware e servizi per soddisfare le esigenze tecnologiche dei propri clienti, anche a livello internazionale. Le sedi sono dislocate in numerose città italiane, tra cui Padova dove ho svolto lo stage e Lodi in cui risiede la sede amministrativa.



Domenico Zucchetti, fondatore dell'azienda, ha avuto la geniale intuizione di costruire un software per agevolare il lavoro dei commercialisti, allora completamente cartaceo e manuale. Con il passare degli anni il suo prodotto ha avuto un successo sempre maggiore tanto da ottenere collaborazioni con aziende del calibro di IBM. A partire da questo, Zucchetti ha continuato a perseguire l'innovazione del proprio prodotto integrandolo con nuovi moduli quali ERP e la più recente fatturazione elettronica, con l'obiettivo di conferire maggiore flessibilità e adattabilità per ogni tipologia di impresa, senza più limitarsi alla categoria dei commercialisti. Forte di questo, Zucchetti si è espansa a livello nazionale ed internazionale ed ora si pone sul mercato con una vasta gamma di servizi per numerosi settori quali industrie manifatturiere, trasporti, logistica, sanità, fitness e molti altri.

### 1.2 Lo stage proposto

Uno dei pilastri di Zucchetti che ne ha caratterizzato la costante crescita è l'innovazione e la propensione alla ricerca di nuove tecnologie.

A questo proposito la sede di Padova è caratterizzata da un reparto di ricerca e sviluppo nel quale sono stato inserito durante la mia esperienza di stage. Il coordinatore di

questo reparto è il dott. Gregorio Piccoli che mi ha proposto il progetto di tirocinio e si è offerto come mio tutor.

L'azienda sta cercando di introdurre nei propri prodotti, in particolare nel software gestionale, un'interfaccia vocale che permetta agli utenti un'interazione più veloce e spontanea per le operazioni più comuni. L'obiettivo è quindi implementare un'interfaccia diversa da quella grafica, con caratteristiche proprie, che esprima un nuovo modo di comunicare con le applicazioni, ancora poco sviluppato ma dalle grandi potenzialità. Il principale ostacolo da superare per il raggiungimento di tale obiettivo è il riconoscimento del linguaggio naturale attraverso un'applicativo software. Per fare ciò l'azienda ha sviluppato delle regole per la creazione di grammatiche in grado di riconoscere comandi espressi con il linguaggio naturale.

Tuttavia è una tecnologia ancora in fase di sviluppo ed in merito a ciò, come progetto di stage, mi sono stati proposte due tematiche pensate per suddividere lo stage in due parti:

1. analizzare i tre assistenti virtuali attualmente più diffusi sul mercato, *Assistant*<sup>[g]</sup>, *Alexa*<sup>[g]</sup> e *Siri*<sup>[g]</sup>, per comprenderne le abilità e, qualora esistesse la possibilità, permettere agli utenti di utilizzarli per impartire comandi al proprio software gestionale Zucchetti;
2. esplorare la possibilità di aggiungere la capacità conversazionale in un'applicazione che utilizzi una grammatica costruita con la nuova tecnologia dell'azienda, ispirandosi anche agli assistenti virtuali precedentemente studiati.

### 1.3 Organizzazione del testo

**Il secondo capitolo** descrive ...

**Il terzo capitolo** approfondisce ...

**Il quarto capitolo** approfondisce ...

**Il quinto capitolo** approfondisce ...

**Il sesto capitolo** approfondisce ...

**Nel settimo capitolo** descrive ...

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*<sup>[g]</sup>;
- \* i termini in lingua straniera o facenti parti del gergo tecnico sono evidenziati con il carattere *corsivo*.

# Capitolo 2

## Lo stage

### 2.1 Descrizione del progetto

Il progetto di stage è legato ad uno degli ambiti più innovativi della ricerca scientifica e informatica: il riconoscimento e l'elaborazione del linguaggio naturale da parte di un calcolatore.

Nel corso degli ultimi anni, l'azienda sta affrontando la sfida della comprensione di comandi vocali per permettere agli utenti di comunicare con i propri prodotti. È infatti in via di sviluppo una loro tecnologia in grado di creare grammatiche atte a riconoscere il linguaggio naturale mediante regole ben precise. In merito a questo argomento si è costruito il progetto di stage articolato poi in due parti:

1. studio degli assistenti virtuali più comuni ed utilizzati;
2. creazione di un'applicazione sotto forma di *proof of concept* capace di intrattenere una conversazione.

La prima parte richiede un'analisi dettagliata e comparativa dei tre assistenti virtuali più utilizzati e, secondo diversi studi, migliori:

- \* Assistant: assistente di Google presente in quasi tutti i dispositivi Android e nei prodotti della linea Google Home e Nest;
- \* Alexa: assistente di Amazon presente in tutta la linea di dispositivi Echo ed integrabile anche nei dispositivi con sistema operativo Android e IOS;
- \* Siri: assistente di Apple presente esclusivamente in tutti i prodotti Apple.

Lo studio deve comprendere tutte le capacità che posseggono, con particolare attenzione alla costruzione di *Skill* personalizzate ed integrabili nei propri progetti, ad eccezione di quelle relative alla costruzione di *smart home* in quanto non sono di diretto interesse per i prodotti aziendali; l'azienda infatti punta alla realizzazione di un assistente utile solo nei propri prodotti e non su larga scala. Inoltre, per ogni assistente, è richiesto un *proof of concept* atto a dimostrare le funzionalità più significative che sono state riscontrate.

Sulla base dei risultati ottenuti nel corso dell'analisi si concretizza la seconda parte del lavoro: creare un'applicazione con un'interfaccia vocale, progettata secondo i canoni studiati in precedenza dagli altri assistenti e basata su grammatiche costruite mediante la tecnologia di Zucchetti, che permetta di intrattenere una conversazione con l'utente

rispondendo alle sue richieste.

Il contenuto dell'applicazione non è vincolato dall'azienda in quanto l'obiettivo è approfondire l'abilità conversazionale e non costruire un'applicazione direttamente utilizzata nel software Zucchetti. In comune accordo è stato scelto il contesto della data di nascita poiché risulta ricca di sfumature sia nella richieste che nelle risposte e possiede parametri specifici e funzionali alla sperimentazione sulla capacità di conversazione. Lo scopo generale del progetto di stage è quindi svolgere un'attività di ricerca e di approfondimento nella tematica del linguaggio naturale poiché è una tecnologia non ancora pienamente affermata ma con ampi margini di sviluppo.

## 2.2 Obiettivi

### 2.2.1 Classificazione

La classificazione degli obiettivi permette di identificarli univocamente e rispetta la seguente notazione:

- \* *OB-x*: obiettivi obbligatori, vincolanti e primario richiesti dall'azienda;
- \* *OD-x*: obiettivi desiderabili, non vincolanti o strettamente necessari ma dal riconoscibile valore aggiunto;
- \* *OF-x*: obiettivi facoltativi dal valore aggiunto riconoscibile ma difficilmente raggiungibili a causa di agenti esterni all'attività di stage.

Il simbolo 'x' rappresenta un numero progressivo intero maggiore di zero.

### 2.2.2 Obiettivi obbligatori

- \* *OB-1*: studio e analisi delle capacità e delle modalità di utilizzo lato sviluppatore di Assistant;
- \* *OB-2*: implementazione sottoforma di *proof of concept* di una *Action* basata su una funzionalità particolarmente rilevante di Assistant, scoperta durante la ricerca;
- \* *OB-3*: studio e analisi delle capacità e delle modalità di utilizzo lato sviluppatore di Alexa;
- \* *OB-4*: implementazione sottoforma di *proof of concept* di una *Skill* basata su una funzionalità particolarmente rilevante di Alexa, scoperta durante la ricerca;
- \* *OB-5*: redazione di un documento che tratti l'analisi dettagliata e comparativa in merito alle tecnologie degli assistenti virtuali studiati;
- \* *OB-6*: studio di regole e caratteristiche dell'algoritmo *HPP (Hidden Probabilistic Parser)* di Zucchetti per la costruzione di grammatiche che riconoscono il linguaggio naturale;
- \* *OB-7*: realizzazione di un'applicazione basata su una grammatica generata da *HPP* con interfaccia vocale e capacità conversazionale.

### 2.2.3 Obiettivi desiderabili

Inizialmente era stato identificato come obiettivo desiderabile l'implementazione di una componente di addestramento per l'applicazione finale. Tuttavia nel corso dello stage, date anche alcune interessanti funzionalità trovate, questo obiettivo è stato modificato nel seguente:

- \* *OD-1*: implementazione di un meccanismo che permetta lo scambio di informazioni specifiche e possibilmente memorizzate durante la conversazione.

### 2.2.4 Obiettivi facoltativi

- \* *OF-1*: studio e analisi delle capacità e delle modalità di utilizzo lato sviluppatore di Siri;
- \* *OF-2*: implementazione sottoforma di *proof of concept* di una *Skill* basata su una funzionalità particolarmente rilevante di Siri, scoperta durante la ricerca.

## 2.3 Pianificazione delle attività

Lo stage è stato svolto in modalità *smart working* con la possibilità di rimanere in comunicazione con il tutor aziendale attraverso Skype. In particolare l'orario di lavoro è stato 9:00-13:00, 14:00-18:00 ed sono stati stabiliti sia una chiamata al termine di ogni giornata che la compilazione di un registro delle attività giornaliero per tracciare e monitorare il lavoro svolto oltre a ricevere feedback costanti dal tutor.

La pianificazione è stata svolta su un totale di 320 ore suddivise secondo quanto riportato nella tabella seguente:

Durata in ore	Date (inizio - fine)	Attività
40	04/05/2020 - 08/05/2020	Studio di Assistant e implementazione di un proof of concept.
40	11/05/2020 - 15/05/2020	Studio di Alexa e implementazione di un proof of concept.
40	18/05/2020 - 22/05/2020	Studio di Siri e implementazione di un proof of concept.
40	25/05/2020 - 29/05/2020	Test e documentazione comparativa di quanto svolto nelle settimane precedenti.
40	01/06/2020 - 05/06/2020	Apprendimento di sistema creato da Zucchetti per l'interazione mediante comandi vocali.
40	08/06/2020 - 12/06/2020	Realizzazione di un'applicazione che implementi una grammatica costruita utilizzando la tecnologia di Zucchetti.
40	15/06/2020 - 19/06/2020	Implementazione del meccanismo per lo scambio di informazioni con capacità di memorizzazione.
40	22/06/2020 - 26/06/2020	Test e documentazione di quanto svolto nelle settimane precedenti.

La pianificazione di dettaglio di ogni settimana è stata strutturata sulla base degli obiettivi stabiliti e viene ora riportata.

### Prima Settimana

Gli obiettivi prefissati sono:

- \* OB-1;
- \* OB-2.

Le attività previste sono l'analisi di Assistant sotto alcuni aspetti principali:

- \* modalità di implementazione per gli intenti relativi ad *Action* di ogni tipologia;
- \* progettazione del prodotto software soprattutto all'interfaccia vocale;
- \* interazione dell'utente con l'assistente;
- \* modalità e sicurezza nel trasferimento dei dati;
- \* sviluppo di un *proof of concept* che implementi una *Action* per dare concretezza allo studio fatto.

### Seconda Settimana

Gli obiettivi prefissati sono:

- \* OB-3;
- \* OB-4.

Le attività previste sono l'analisi di Alexa sotto alcuni aspetti principali:

- \* modalità di implementazione per gli intenti relativi ad *Skill* di ogni tipologia;
- \* progettazione del prodotto software soprattutto all'interfaccia vocale;
- \* interazione dell'utente con l'assistente;
- \* modalità e sicurezza nel trasferimento dei dati;
- \* sviluppo di un *proof of concept* che implementi una *Skill* per dare concretezza allo studio fatto.

### Terza Settimana

Gli obiettivi prefissati sono:

- \* OF-1;
- \* OF-2.

Le attività previste sono l'analisi di Siri sotto alcuni aspetti principali:

- \* modalità di implementazione per gli intenti relativi ad *Skill* di ogni tipologia;
- \* progettazione del prodotto software soprattutto all'interfaccia vocale;



- \* interazione dell'utente con l'assistente;
- \* modalità e sicurezza nel trasferimento dei dati;
- \* sviluppo di un *proof of concept* che implementi una *Skill* per dare concretezza allo studio fatto

### Quarta Settimana

L'obiettivo prefissato è:

- \* OB-5.

Le attività previste sono:

- \* verifica del funzionamento dei *proof of concept* realizzati;
- \* stesura della documentazione atta a riportare in modo dettagliato la comparazione tra le tecnologie studiate nelle settimane precedenti.

### Quinta Settimana

L'obiettivo prefissato è:

- \* OB-6.

Le attività previste sono:

- \* studio dell'algoritmo *HPP* di Zucchetti;
- \* esecuzione di alcune prove concrete per verificarne il corretto apprendimento.

### Sesta Settimana

L'obiettivo prefissato è:

- \* OB-7.

Le attività previste sono:

- \* progettazione di un'applicazione, anch'essa sotto forma di *proof of concept*, che implementi un'interfaccia vocale per il riconoscimento della data di nascita.
- \* realizzazione dell'applicazione.

### Settima Settimana

L'obiettivo prefissato è:

- \* OD-01.

Le attività previste sono:

- \* analisi del meccanismo che permette lo scambio di informazioni con memorizzazione sulla base di quanto già implementato negli assistenti virtuali studiati;
- \* implementazione del meccanismo.

## Ottava Settimana

In questo ultimo periodo è prevista l'implementazione di eventuali test e la stesura della documentazione finale sull'applicazione realizzata.

## 2.4 Tecnologie e strumenti utilizzati

### 2.4.1 Tecnologie

#### Kotlin

*Kotlin*<sup>[§]</sup> è un linguaggio di programmazione fortemente tipizzato, multi-paradigma, multi-piattaforma e open source, sviluppato da JetBrains nel 2011 e strutturato per interagire con la *Java Runtime Environment*. Grazie a quest'ultima caratteristica, è garantita piena compatibilità con ogni applicazione basata sulla *JVM*<sup>[§]</sup>, comprese quelle Android. Nel mio caso ho utilizzato Kotlin per costruire un'applicazione Android sotto forma di *proof of concept* che implementi una particolare funzionalità di Assistant.

#### ECMAScript

ECMAScript versione 6 è lo standard di javascript, un linguaggio debolmente tipizzato, orientato agli oggetti e agli eventi che viene utilizzato nella programmazione Web lato client e lato server con Node.js. Nel mio caso ho utilizzato ECMAScript per costruire un *proof of concept* che implementi una particolare funzionalità di Alexa nella sua console dedicata. Inoltre ho realizzato la parte principale dell'applicazione che fa uso dell'algoritmo HPP di Zucchetti.

#### Swift

Swift è un linguaggio di programmazione orientato agli oggetti e utilizzato esclusivamente per costruire applicazioni per i dispositivi Apple. Nel mio caso l'ho utilizzato per costruire un'applicazione che mi permetta di esplorare una funzionalità di Siri.

#### HTML

HTML è un linguaggio di markup principalmente pensato per pagine o applicazioni basate su web. Io ho utilizzato HTML5, l'ultima versione, in quanto è più semplice e veloce nella sintassi e non necessita di piena compatibilità con tutti i browser o le versioni meno recenti.

#### CSS

CSS è un linguaggio per la realizzazione di fogli di stile da applicare a pagine o applicazioni basate su web. Io ho utilizzato CSS3, l'ultima versione, per l'interfaccia grafica minimale necessaria.

### 2.4.2 Strumenti

#### Android Studio

Android Studio è un ambiente di sviluppo integrato per realizzare applicazioni da utilizzare nella piattaforma Android e basato sul software IntelliJ IDEA di JetBrains.

Pubblicato a fine 2014 ha sostituito l'utilizzo dei plug-in specifici di Eclipse per lo sviluppo di applicazioni Android diventando il più diffuso. Durante il mio lavoro è stato ausiliario nell'implementazione del *proof of concept* per Assistant.

Tra gli strumenti di Android Studio è rilevante App Actions Test Tool che permette di simulare l'utilizzo di Assistant nell'applicazione che si sta sviluppando, in un dispositivo fisico.

#### **Alexa developer console**

Alexa developer console è un ambiente di sviluppo di Amazon dedicato alla costruzione di *Skill* per Alexa. Durante il mio lavoro è stato ausiliario all'implementazione del *proof of concept* per Alexa

#### **Xcode**

Xcode è un ambiente di sviluppo integrato per realizzare applicazione da utilizzare nelle piattaforme Apple. Durante il mio lavoro è stato ausiliario nell'implementazione del *proof of concept* per Siri.

#### **WebStorm**

WebStorm è un ambiente di sviluppo integrato per realizzare applicazioni web con il supporto ad esempio di linguaggi quali HTML, CSS, Javascript e PHP. Durante il mio lavoro è stato ausiliario per l'implementazione dell'intera applicazione finale.

## **2.5 Motivazioni personali**

Durante l'iniziativa StageIT, dedicata agli stage in azienda, ho inizialmente cercato una proposta che trattasse argomenti innovativi, possibilmente di ricerca, rientranti in una delle seguenti tematiche:

- \* intelligenza artificiale;
- \* blockchain;
- \* analisi e previsioni di dati.

La scelta è ricaduta sulla proposta di Zucchetti ovvero il riconoscimento e l'elaborazione del linguaggio naturale.

La motivazione principale per cui ho scelto questo progetto è l'impiego di uno dei rami dell'intelligenza artificiale che rappresenta anche una sfida intrapresa da decenni: superare il *test di turing*<sup>[g]</sup>. È un argomento non trattato nel percorso di studi della laurea triennale in informatica ma a mio parere molto interessante.

La seconda motivazione risiede sul contenuto specifico del progetto ovvero svolgere un'attività di ricerca sugli assistenti virtuali più diffusi nel mercato e sull'implementazione della capacità conversazionale mediante una tecnologia Zucchetti ancora in via di sviluppo. Ho personalmente trovato stimolante poter imparare e realizzare qualcosa di innovativo ma in gran parte basato su conoscenze apprese nel corso della mia attività di ricerca.

La terza motivazione risiede nell'azienda stessa in quanto è una delle software house più grandi in Italia, dalle possibilità lavorative in numerosi ambiti volti anche alla ricerca, con cui ho avuto un ottimo rapporto di collaborazione per lo svolgimento del progetto didattico inerente al corso di Ingegneria del Software.



## Capitolo 3

# Gli assistenti virtuali

### 3.1 Premessa

Un assistente virtuale è un software capace di interpretare il linguaggio naturale e dialogare con gli utenti che ne fanno uso, eseguendo determinati compiti. Generalmente necessita di un'attività di addestramento, precedente al suo utilizzo, che gli permetta di apprendere e migliorare le proprie abilità.

Gli assistenti virtuali analizzati sono Assistant, Alexa e Siri ma, nonostante siano software di tre aziende diverse, il loro funzionamento è molto simile. Inizialmente lo sviluppatore deve costruire un'abilità che viene richiamata attraverso l'assistente virtuale assegnando nome, frasi di richiamo e azioni da eseguire. Queste abilità sono chiamate *Action* per Assistant, *Skill* per Alexa e *Comandi* per Siri. Dato che gli assistenti virtuali attuali sono pensati per un utilizzo di breve durata, è molto importante progettare le abilità considerando questo fattore, con l'obiettivo di dare agli utenti una migliore esperienza d'uso.

Una caratteristica comune alle abilità di tutti gli assistenti successivamente trattati è il meccanismo di funzionamento su cui si fondano: gli intenti. Questi consistono in un'azione che soddisfa una richiesta vocale effettuata da un utente e il servizio che la esegue, più concretamente il codice, viene detto *fulfillment*. Questo concetto sarà spiegato più dettagliatamente per ciascun assistente nel paragrafo dedicato al funzionamento.

### 3.2 Assistant

#### 3.2.1 Introduzione

Assistant è l'assistente virtuale di Google ed è capace di riconoscere un comando vocale, elaborarlo attraverso un ragionamento e fornire una risposta. È una tecnologia in continuo miglioramento grazie anche all'immensa mole di dati che Google ha a disposizione per il suo addestramento.

Reso pubblico dal 2016, Assistant è ora integrato in tutti i dispositivi con sistema operativo Android, a partire dalla versione 6.0 se hanno a disposizione almeno 1.5 GB di memoria RAM oppure dalla versione 5.0 se hanno a disposizione almeno 1 GB di memoria RAM. È inoltre installabile nei dispositivi con sistema operativo iOS a partire dalla versione 10 e iPadOS; tuttavia l'integrazione è pessima in quanto per richiamarlo

è necessario invocare Siri. Infine è fruibile nei dispositivi della linea Home e Nest di Google, costruiti e pensati appositamente per ottimizzarne le funzionalità.

### 3.2.2 Casi d'uso

- \* controllo di dispositivi intelligenti all'interno di un ambiente domestico: *Home Actions*;
- \* dialogo con l'utente eseguendo anche operazioni specifiche: *Conversational Actions*;
- \* integrazione di contenuti multimediali pensati per Assistant all'interno di pagine web: *Content Actions*.
- \* integrazione di Assistant nelle applicazioni Android per eseguirne determinate funzionalità: *App Actions*.

Le *Home Actions* non sono oggetto di analisi in quanto non di interesse per lo stage proposto.

### 3.2.3 Conversational Actions

#### Descrizione

Le *Conversational Actions* estendono le funzionalità di Assistant consentendo di creare esperienze di conversazione personalizzata con gli utenti. Esse infatti permettono di gestire le richieste rivolte all'Assistente e le risposte costruite a seguito dell'elaborazione. Una caratteristica importante è la possibilità di comunicare con servizi Web o applicazioni esterne che forniscono una logica di conversazione aggiuntiva grazie alle apposite *SDK*.

#### Funzionamento

Il principio di funzionamento si articola nei seguenti passi:

1. un utente lancia una richiesta sotto forma di comando vocale al dispositivo che ospita Assistant;
2. il dispositivo attiva un apposito elaboratore che riconosce le parole pronunciate trasformandole in stringhe di testo;
3. il dispositivo invia la stringa riconosciuta ad un server remoto di Google, tramite protocollo *HTTPs*, dove risiede la *NLU* per l'elaborazione;
4. la business logic contenuta nel server remoto prova a verificare la possibile corrispondenza tra la stringa ricevuta e l'insieme di frasi che lo sviluppatore ha inserito durante la costruzione della propria abilità;
5. se non vi è alcuna corrispondenza viene subito ritornata una risposta negativa e riferito all'utente. Si ricomincia quindi dal punto 1;
6. se una corrispondenza dà esito positivo viene selezionato l'intento associato con il suo servizio di *fulfillment* che si occupa dell'esecuzione;

7. la conclusione dell'esecuzione prevede la costruzione e l'invio della risposta al dispositivo mittente;
8. il dispositivo riferisce la risposta all'utente che potrà poi procedere con una nuova richiesta, fino al termine dell'esecuzione dell'abilità previsto dal programmatore durante la costruzione, oppure interrompere forzatamente la conversazione.

Qualora la richiesta dell'utente fosse di invocazione, prima di scegliere l'intento da eseguire viene cercata una corrispondenza tra le Action a disposizione per capire quale avviare. Infine, affinché la Action tenga un comportamento adeguato, è necessario applicare correttamente i principi di costruzione e svolgere una consistente attività di addestramento durante lo sviluppo.

### Costruzione

Nella costruzione di una Conversational Action la prima attività da svolgere è la progettazione che deve essere mirata a tre aspetti:

- \* modalità di invocazione;
- \* tipologia e formato delle richieste accettate;
- \* tipologia e formato delle risposte che l'utente si aspetta.

Per la progettazione di richieste e risposte è necessario ragionare sullo scopo dell'Action che si vuole implementare e svolgere un'analisi statistica e probabilistica sulle frasi che l'utente potrebbe pronunciare o aspettarsi da Assistant, cercando di rendere la conversazione più naturale possibile. Una volta inserite le frasi, Assistant viene addestrato su di esse al fine di interpretarle correttamente.

Per la modalità di invocazione Google fa una distinzione:

- \* invocazione esplicita;
- \* invocazione implicita.

L'invocazione esplicita consiste nell'esprimere una frase che riporti la seguente struttura:

1. parola di attivazione: "Hey Google" oppure "Ok Google";
2. parola di avvio: chiedi, fai, dimmi, raccontami e vocaboli simili;
3. nome di invocazione: nome deve identificare la Action;
4. tips: parametri aggiuntivi, possibilmente opzionali, implementati come variabili che specificano ulteriormente la richiesta dell'utente;
5. elementi aggiuntivi: l'utente può pronunciare parole aggiuntive con lo scopo di contestualizzare o precisare il dominio della richiesta.

Grazie a questa struttura fissa, Assistant riesce a comprendere quale Action attivare per avviare la conversazione.

L'invocazione implicita, invece, si verifica quando l'utente effettua una richiesta senza aver esplicitato l'Action da eseguire. In questo caso la business logic di Assistant ha il compito di comprendere la richiesta e associare l'Action che ritiene più corretta. Qualora non ne trovasse alcuna, effettuerà una ricerca in Internet inserendo come testo la richiesta stessa e ne ritornerà i risultati come risposta. Tuttavia il funzionamento

di questa modalità non è garantito da Google in quanto è ancora in via di sviluppo e richiede che lo sviluppatore abbia inserito un numero di frasi per l'addestramento sufficientemente ampio e completo.

L'attività che segue la progettazione è l'implementazione e per svolgerla Google offre due strumenti:

- \* Dialogflow;
- \* Conversational Actions SDK.

Dialogflow è uno strumento utilizzato per creare conversazioni personalizzate in modo semplice ed intuitivo. Si basa sulla *NLU* di Assistant per la comprensione del linguaggio naturale e si appoggia ad un *webhook* per la gestione dei dati, *Firebase* è quello predefinito. Per costruire una Conversational Action con *Dialogflow* è necessario creare un agente ovvero un modulo di comprensione del linguaggio naturale che gestisce le conversazioni con gli utenti sgravando lo sviluppatore di numerosi oneri.

Le Conversational Actions SDK consistono in un'interfaccia HTTPs che permette di costruire ed eseguire le Conversational Actions all'interno di una propria applicazione per elaborare le richieste effettuate ad Assistant. Grazie all'interfaccia HTTPs infatti Assistant può comunicare con applicazioni terze. Il requisito che ne deriva è possedere una propria *NLU* per la comprensione del linguaggio naturale. L'architettura ad alto livello del sistema che rappresenta l'utilizzo delle Conversational Actions attraverso SDK è così composta:

- \* dispositivo fisico con Assistant integrato che riceve la richiesta vocale dell'utente;
- \* *NLU* di Google che ha il compito di riconoscere la richiesta dell'utente e associare l'intento corretto;
- \* servizio cloud remoto che, ricevuta la richiesta di esecuzione dell'intento, esegue il servizio di fulfillment associato e ritorna la risposta alla *NLU* che a sua volta la ritorna al dispositivo.

Le componenti di un agente di Dialogflow oppure una Conversational Actions costruita con le *SDK* sono uguali:

- \* Default Actions: azione cui corrisponde l'evento chiamato *GOOGLE\_ASSISTANT\_WELCOME* per Dialogflow e *actions.intent.MAIN* per le *SDK* che rappresenta la prima interazione con l'Action; una condizione necessaria che la caratterizza è l'esistenza di uno ed un solo intento per gestire questo evento. La risposta predefinita è statica e preconfigurata ma è comunque possibile renderla dinamica costruendo un servizio di *fulfillment* che la compone a tempo di esecuzione;
- \* Additional Actions: azione aggiuntiva alla Default Actions utilizzata per aggiungere e specificare le capacità dell'Action. Possono essere molteplici e sono automaticamente indicizzate per l'invocazione implicita.

Ognuna di queste componenti corrisponde ad un intento che l'Action può gestire.

Una volta stabilite queste componenti, è necessario definire l'interfaccia della propria conversazione. Per poterlo fare si deve creare un intento definendo:

- \* nome;
- \* contesto;



- \* evento scatenante;
- \* frasi di input per l'addestramento;
- \* azioni da eseguire;
- \* eventuali parametri aggiuntivi per la conversazione e formato della risposta.

Le azioni da eseguire corrispondono alla creazione di un *fulfillment* che fornisce la logica per processare la richiesta dell'utente e genera una risposta.

Qui emerge una grande differenza tra Dialogflow e *SDK*: mentre il primo utilizza la *NLU* di Google rivelandosi quindi di poco interesse per i progetti dell'azienda, il secondo prevede l'utilizzo obbligatorio di una *NLU* proprietaria dello sviluppatore che invece si è rivelato utile per l'azienda.

### Comunicazione

Un'altra differenza riscontrata è una diretta conseguenza della prima in quanto per le *SDK*, utilizzando la propria *NLU*, diventa obbligatorio un metodo di comunicazione tra le componenti dell'architettura illustrata. Google infatti impone lo scambio di dati tramite oggetti JSON con la seguente struttura:

- \* *isInSandbox*: attributo booleano, se vale *true* indica l'utilizzo in un ambiente di prova e, qualora sia utilizzato, vieta lo scambio di denaro;
- \* *Surface*: oggetto che contiene la modalità di interazione, scelta tra solo testuale, solo visiva e multimediale;
- \* *Inputs*: oggetto che contiene l'insieme degli input specificati per la *Actions* tra cui la richiesta dell'utente sotto forma di stringa testuale;
- \* *User*: oggetto che contiene le informazioni dell'utente che ha eseguito la richiesta;
- \* *Device*: oggetto che contiene le informazioni del dispositivo da cui è arrivata la richiesta;
- \* *Conversation*: oggetto che contiene i dati strettamente legati alla conversazione in corso tra cui *conversationId* che la identifica univocamente e *conversationToken* che salva i dati durante la conversazione.

Di particolare rilevanza è la possibilità di salvare i dati durante la conversazione in quanto permette sia di richiedere all'utente determinati dati importanti per la corretta esecuzione dell'Action ma soprattutto di dare all'utente la sensazione di interagire con un'intelligenza che ha capacità di memoria. Questo è molto importante perché permette di definire e mantenere il contesto della conversazione, qualunque esso sia, incrementando notevolmente la qualità dell'esperienza d'uso dell'utente.

Per capire quali elementi devono essere salvati, Assistant utilizza delle variabili all'interno delle possibili frasi di richiesta dette *tips*. Essi devono essere definiti dallo sviluppatore durante la progettazione così che, quando l'utente pronuncia parole o dati in una posizione all'interno della frase corrispondente a quella di una variabile, vengono automaticamente salvati nel *conversationToken*. Il loro limite è rappresentato dalla conversazione stessa: al suo termine tutti i dati salvati vengono persi. Perciò, se si vuole una persistenza duratura nel tempo, bisogna storicizzare i dati in un database.

### 3.2.4 Content Actions

**Descrizione**

**Funzionamento**

**Costruzione**

### 3.2.5 App Actions

**Descrizione**

**Funzionamento**

**Costruzione**

### 3.2.6 Proof of concept

## 3.3 Alexa

### 3.3.1 Introduzione

Alexa è l'assistente virtuale di Amazon ed è capace di riconoscere un comando vocale, elaborarlo attraverso un ragionamento e fornire una risposta. È una tecnologia in continuo sviluppo grazie anche alla consistente mole di dati che Amazon ha a disposizione per il suo addestramento.

La prima versione di Alexa risale al 2014 e da allora ha fatto notevoli miglioramenti. È integrato in tutti i dispositivi della linea Echo di Amazon costruiti appositamente per ottimizzarne l'utilizzo; tuttavia è installabile in tutti i dispositivi con sistema operativo Android in versione 5.0 o maggiore, iOS in versione 9.0 o maggiore e iPadOS.

### 3.3.2 Casi d'uso

- \* controllo di dispositivi intelligenti all'interno di un ambiente domestico attraverso le *Skill*;
- \* dialogo con l'utente eseguendo anche operazioni specifiche attraverso le *Skill*.

Le *Skill* per il controllo di dispositivi intelligenti non sono oggetto di analisi in quanto non di interesse per lo stage proposto.

### 3.3.3 Skill di conversazione

**Descrizione**

Le *Skill* consistono in funzionalità personalizzate e aggiuntive per Alexa con l'obiettivo di migliorare l'esperienza d'uso degli utenti. Attraverso le *Skill* lo sviluppatore può ricevere le richieste rivolte ad Alexa, soddisfarle e restituire una risposta.

Una caratteristica importante è la possibilità di comunicare con servizi Web o applicazioni esterne che forniscono una logica di conversazione aggiuntiva grazie alle *API* presenti nell'Alexa Skill Kit.

**Funzionamento**

Il principio di funzionamento si articola nei seguenti passi:

1. un utente lancia un comando vocale al dispositivo che ospita l'assistente;
2. il dispositivo attiva un apposito elaboratore che riconosce le parole pronunciate trasformandole in stringhe di testo;
3. il dispositivo manda la stringa riconosciuta ad un server remoto per l'elaborazione;
4. il server remoto attiva un apposito componente software che prova ad eseguire un match tra la stringa ricevuta e l'insieme di frasi che lo sviluppatore ha inserito nella propria abilità;
5. se il match ha dato esito positivo viene selezionato l'intento corretto, sulla base del contenuto della stringa;
6. viene richiamato il servizio di *fulfillment*, rappresentante il codice da eseguire, che porterà a termine l'intento;
7. viene costruita la risposta e ritornata al dispositivo che ospita l'assistente;
8. il dispositivo riferisce la risposta all'utente che potrà procedere con una nuova richiesta fino al termine dell'esecuzione dell'abilità.

## Costruzione

### 3.3.4 Proof of concept

## 3.4 Siri

### 3.4.1 Introduzione

Siri è l'assistente virtuale di Apple ed è capace di riconoscere un comando vocale, elaborarlo attraverso un ragionamento e fornire una risposta. È una tecnologia in continuo sviluppo grazie anche alla contingente mole di dati a disposizione di Apple per il suo addestramento.

La prima versione è stata introdotta in iOS 5 nel 2012, senza ancora offrire il supporto a tutte le lingue e a tutti i dispositivi. Ora invece è integrato in Homepod e tutti i dispositivi con sistema operativo iOS versione 8.0 o superiore, iPadOS, watchOS, tvOS e MacOS versione 10.12 o superiore. Rimane comunque un'esclusiva di Apple e non è installabile in altri sistemi.

### 3.4.2 Casi d'uso

- \* controllo di dispositivi intelligenti all'interno di un ambiente domestico attraverso i *comandi*.
- \* dialogo con l'utente eseguendo anche operazioni specifiche attraverso i *comandi*.
- \* integrazione di Siri nelle applicazioni per i dispositivi Apple per eseguire determinate funzionalità.

Le *Skill* per il controllo di dispositivi intelligenti non sono oggetto di analisi in quanto non di interesse per lo stage proposto.

### 3.4.3 Comandi di conversazione

#### Descrizione

I *Comandi* consistono in funzionalità aggiuntive e personalizzate per Siri che migliorano l'esperienza d'uso degli utenti. Permettono di ricevere le richieste effettuate a Siri, soddisfarle e fornire delle risposte adeguate.

#### Funzionamento

Il principio di funzionamento si articola nei seguenti passi:

1. un utente lancia un comando vocale al dispositivo che ospita l'assistente;
2. il dispositivo attiva un apposito elaboratore che riconosce le parole pronunciate trasformandole in stringhe di testo;
3. il dispositivo manda la stringa riconosciuta ad un server remoto per l'elaborazione;
4. il server remoto attiva un apposito componente software che prova ad eseguire un match tra la stringa ricevuta e l'insieme di frasi che lo sviluppatore ha inserito nella propria abilità;
5. se il match ha dato esito positivo viene selezionato l'intento corretto, sulla base del contenuto della stringa;
6. viene richiamato il servizio di *fulfillment*, rappresentante il codice da eseguire, che porterà a termine l'intento;
7. viene costruita la risposta e ritornata al dispositivo che ospita l'assistente;
8. il dispositivo riferisce la risposta all'utente che potrà procedere con una nuova richiesta fino al termine dell'esecuzione dell'abilità.

#### Costruzione

##### 3.4.4 Proof of concept

### 3.5 Analisi comparativa

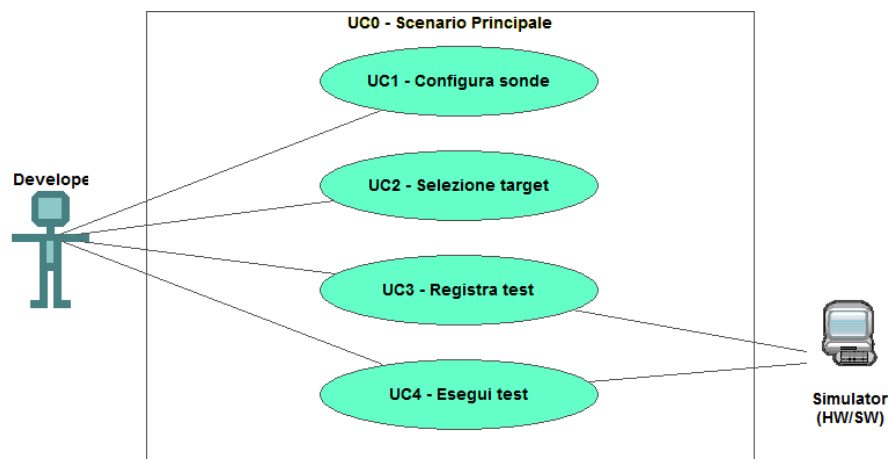
## Capitolo 4

# 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'I-DE.

**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 requisiti 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

# Conclusioni

6.1 Consuntivo finale

6.2 Raggiungimento degli obiettivi

6.3 Conoscenze acquisite

6.4 Valutazione personale



Appendice A

Appendice A

Citazione

---

Autore della citazione









# Bibliografia