MegAlexa

Arricchitore di skill di Amazon Alexa

Piano di Qualifica

Gruppo ZeroSeven

Versione	2.0.0
Data Redazione	2018-12-24
Redazione	Stefano Zanatta
	Andrea Deidda
	Bianca Andreea Ciuche
Verifica	Matteo Depascale
	Mirko Franco
	Andrea Deidda
Approvazione	Gian Marco Bratzu
Uso	Esterno
Distribuzione	Prof. Tullio Vardanega
	Prof. Riccardo Cardin
	Gruppo ZeroSeven
	Zero12 s.r.l.
Email di contatto	zerosevenswe@gmail.com

Registro delle modifiche

Versione	Data	Descrizione	Autore	Ruolo
2.0.4	2019-04-10	stesura resoconto prog. dett. codifica §B.4	Stefano Zanatta	Analista
2.0.3	2019-04-03	Spostato riferimento Ciclo di Deming da §1.4.1 a 1.4.2	Mirko Franco	Analista
2.0.2	2019-03-20	Sistemati range metriche in §2.2, §3.2 e §3.3	Andrea Deidda	Analista
2.0.1	2019-03-19	Aggiunte descrizioni in §B.3	Andrea Deidda	Analista
2.0.0	2019-03-07	Approvazione per il rilascio RP	Gian Marco Bratzu	Responsabile
1.4.0	2019-03-07	Verifica documento	Matteo Depascale	Verificatore
1.3.1	2019-03-06	Stesura §B.3, §A	Stefano Zanatta	Progettista
1.3.0	2019-02-08	Verifica documento	Andrea Deidda	Verificatore
1.2.1	2019-02-08	Stesura §B.2	Stefano Zanatta	Analista
1.2.0	2019-02-07	Verifica §3	Mirko Franco	Verificatore
1.1.0	2019-02-05	Verifica §2	Stefano Zanatta	Verificatore
1.0.6	2019-02-03	Stesura §??, §?? e §??	Andrea Deidda	Analista
1.0.5	2019-02-01	Stesura §2	Stefano Zanatta	Analista

1.0.4	2019-02-03	Modifica grammatica §3.2	Stefano Zanatta	Analista
1.0.3	2019-02-02	Stesura §3.3	Bianca Andreea Ciuche	Analista
1.0.2	2019-01-27	Stesura §3.2	Stefano Zanatta	Analista
1.0.1	2019-01-27	Modifica struttura documento	Stefano Zanatta	Analista
1.0.0	2019-01-10	Approvazione per il rilascio RR	Stefano Zanatta	Responsabile
0.2.0	2019-01-09	Verifica documento	Andrea Deidda	Verificatore
0.1.0	2019-01-09	Verifica §B.3	Andrea Deidda	Verificatore
0.0.6	2018-01-08	Stesura Standard di qualità	Matteo Depascale	Analista
0.0.5	2018-01-07	Stesura §B, mancano le metriche	Stefano Zanatta	Analista
0.0.4	2019-01-05	Stesura Visione generale e Gestione Amministrativa della revisione	Ludovico Brocca	Analista
0.0.3	2019-01-03	Stesura §2.2 e §B.3	Ludovico Brocca	Analista
0.0.2	2018-12-29	Stesura §1	Stefano Zanatta	Amministratore
0.0.1	2018-12-24	Struttura documento	Ludovico Brocca	Amministratore

Indice

Elenco delle tabelle

Capitolo 1

Introduzione

1.1 Scopo del documento

Il *Piano di Qualifica* ha lo scopo di definire gli obbiettivi di qualità che il gruppo perseguita per il proprio prodotto. Per ottenere tali obbiettivi è necessario un processo di verifica continua di ogni attività. Questo consente di rilevare e correggere le anomalie riscontrate tempestivamente.
Questo documento descrive nel dettaglio la qualità dei processi più vicini nel tempo e ad alto livello quelli più lontani, per poi essere aggiornato con nuovi contenuti ogni volta che il gruppo lo ritiene necessario.

1.2 Scopo del prodotto

Lo scopo del progetto è sviluppare un applicativo mobile in grado di creare delle routine personalizzate per gli utenti gestibili tramite $Alexa_G$ di $Amazon_G$. L'obbiettivo è creare $skill_G$ in grado di avviare $workflow_G$ creati dagli utenti fornendogli dei $connettori_G$.

1.3 Glossario

Al fine di evitare ogni ambiguità di linguaggio e massimizzare la comprensione dei documenti, i termini tecnici, di dominio, gli acronimi e le parole che necessitano di essere chiarite, sono riportate nel *Glossario v3.0.0*.
Ogni occorrenza di vocaboli presenti nel *Glossario* è marcata da una "G" maiuscola in pedice.

1.4 Riferimenti

1.4.1 Normativi

- **Norme di Progetto**: *Norme di Progetto v3.0.0*;
- **Capitolato$_G$ C4**: *MegAlexa$_G$*: arricchitore di skill di Amazon Alexa[1].

1.4.2 Informativi

- **Piano di Progetto**: *Piano di Progetto v3.0.0*;
- **Complessità ciclomatica**[2];
- **Software Testing Fundamentals: Methods and Metrics** di Marnie L. Hutcheson, Wiley Publishing, Inc. capitolo 2,4 e 5.
- **Ciclo di Deming**; [3].

[1] https://www.math.unipd.it/~tullio/IS-1/2018/Progetto/C4.pdf
[2] https://it.wikipedia.org/wiki/Complessit%C3%A0_ciclomatica
[3] https://it.wikipedia.org/wiki/Ciclo_di_Deming

Capitolo 2

Qualità di Processo

2.1 Scopo

Al fine di garantire la massima efficacia del prodotto finale è necessario controllare e misurare i processi che concorrono alla sua realizzazione; a tal fine si fa riferimento allo standard ISO/IEC 15504 conosciuto anche come SPICE per una valutazione complessiva sulla maturità dei processi e il loro miglioramento continuo.
Viene applicato inoltre il ciclo di Deming (PDCA), al fine di assicurare un miglioramento continuo delle attività di processo.

2.2 Processi

Vengono elencati gli obiettivi perseguiti dal gruppo per assicurare la qualità di processo, insieme alle metriche adottate per misurare ciascuno di essi.

2.2.1 QP001: Pianificazione delle attività di progetto, valutazione e controllo dei processi

Verificare che il processo di pianificazione di progetto risulti adeguato è un obiettivo di primaria importanza: è opportuno quindi assicurarsi che la pianificazione temporale (comprendente l'assegnazione di task in conformità agli incrementi previsti nel calendario) e i costi sostenuti siano supportati da metriche in grado di verificare che lo stato del progetto sia conforme alle pianificazioni precedentemente prodotte.

2.2.1.1 Obiettivi

- **Pianificazione temporale**
  - **Gestione delle task**: verificare che l'assegnazione e il completamento delle task sia conforme a quanto pianificato e che non si presentino scostamenti;
  - **Calendario**: la pianificazione delle task da assegnare deve essere adatta alla quantità di lavoro richiesto per il suo completamento.
- **Budget**: accertarsi che i costi attuali derivanti dal lavoro svolto siano il più possibile conformi a quanto preventivato.

2.2.1.2 Metriche utilizzate

- **MP001**: Schedule Variance
  - **Range di accettazione**: >= -3 giorni;
  - **Range di ottimalità**: >= 0 giorni.
- **MP002**: Cost Variance
  - **Range di accettazione**: >= -4;
  - **Range di ottimalità**: >= 0.

2.2.2 QP002: Miglioramento continuo delle attività di processo

Lo standard SPICE definisce dei livelli di maturità da perseguire per assicurare il miglioramento continuo delle attività di processo, obiettivo da perseguire se viene adottato il modello PDCA per lo sviluppo di un progetto software.

2.2.2.1 Obiettivi

- **Maturità dei processi**: un processo diventa maturo quando il suo risultato è prevedibile prima della sua attuazione e risulta ottimizzato quando le risorse da lui impiegate vengono utilizzate con la massima efficienza possibile: vista l'inesperienza del gruppo, ottenere la piena ottimalità di tutti i processi è un obbiettivo molto difficile da raggiungere, tuttavia, attraverso il processo di miglioramento continuo è possibile prevedere una piena maturità degli stessi.
  A tal scopo si fa riferimento alle metriche definite dallo standard ISO/IEC 15504 per la verifica della qualità dei processi.

2.2.2.2 Metriche utilizzate

- **MP003**: SPICE capability level
  - **Range di accettazione**: 3-5;
  - **Range di ottimalità**: 4-5.
- **MP004**: SPICE process attributes
  - **Range di accettazione**: L-(>50%-85%);
  - **Range di ottimalità**: F >85%-100%.

2.2.3 QP003: Analisi e prevenzione dei rischi

Il processo intende individuare e prevenire i rischi che potrebbero insorgere durante l'attività di progetto:

2.2.3.1 Obiettivi

- **Individuazione dei rischi**: per ogni fase del progetto, verranno analizzati e attualizzati i rischi che potrebbero insorgere, cercando, ove possibile, di automatizzare le procedure che mitigano la loro occorrenza; non vengono definiti range di accettazione e ottimalità per le metriche adottate in questo processo.

2.2.3.2 Metriche utilizzate

- **MP005**: Occorrenza rischi non previsti
  - **Range di accettazione**: 0-4;
  - **Range di ottimalità**: 0.
- **MP006**: Indisponibilità dei servizi
  - **Range di accettazione**: 0;
  - **Range di ottimalità**: 0.

2.2.4 QP004: Verifica del software

Il processo si occupa di verificare che il software prodotto sia conforme ai $requisiti_G$ stabiliti con la proponente e la committente, che sia privo di errori e che il codice scritto risulti chiaro, conciso ed efficiente.

2.2.4.1 Obiettivi

- **Chiarezza del codice**: il codice prodotto deve risultare il più possibile chiaro, devono essere seguite le norme descritte nelle *Norme di Progetto* ed esso deve essere supportato da commenti che chiariscano il funzionamento delle unità di codice a cui fanno riferimento;

- **Prevenzione degli errori**: ogni unità di codice deve risultare il più possibile privo di errori e di bug_G prima del suo utilizzo;

- **Passaggio dei test**: ogni unità di codice fa riferimento a un test di unità a lui assegnato, è obbligatorio che tutti i test siano definiti pre-sviluppo e che ogni unità di codice passi il test ad essa riferita.
  Per ulteriori informazioni si rimanda alla sezione A.

2.2.4.2 Metriche utilizzate

Le seguenti metriche vengono utilizzate da SonarCloud per calcolare l'indice di manuntenibilità secondo le sue best practice.[1].

- **MP007**: Complessità ciclomatica
  - **Range di accettazione**: 1-15;
  - **Range di ottimalità**: 1-10.

- **MP008**: Numero di parametri per metodo
  - **Range di accettazione**: 0-8;
  - **Range di ottimalità**: 0-4.

- **MP009**: Numero di livelli di annidamento
  - **Range di accettazione**: 1-6;
  - **Range di ottimalità**: 1-3.

- **MP010**: Attributi per classe
  - **Range di accettazione**: 0-16;
  - **Range di ottimalità**: 3-8.

[1] https://sonarcloud.io/documentation/user-guide/metric-definitions/

2.2.5 QP005: Gestione dei test

L'obiettivo di tale processo è misurare l'efficacia del piano di test adottato: esso deve fornire risultati quantificabili sulla qualità del codice prodotto e permettere azioni correttive mirate.
Si fa riferimento, a scopo informativo, al libro *Software Testing Fundamentals: Methods and Metrics* scritto da Marnie L. Hutcheson per la definizione delle metriche adottate e per la stesura del piano dei test di unità (vedi sezione **??**).

2.2.5.1 Obiettivi

- **Qualità del piano di test.**

2.2.5.2 Metriche utilizzate

- **MP011**: Tempo medio del team di sviluppo per la risoluzione di errori
  - **Range di accettazione**: 0h-4h;
  - **Range di ottimalità**: 0h-2h.
- **MP012**: Efficienza della progettazione dei test
  - **Range di accettazione**: 0.5h-3.5h;
  - **Range di ottimalità**: 1h-2h.

2.2.6 QP006: Versionamento e build

2.2.6.1 Obiettivi

- **Correttezza dei commit**: Ogni commit deve superare i controlli automatici, nel caso in cui un commit presenti degli errori essi dovranno essere corretti immediatamente;
- **Dimensione dei commit**: Le modifiche apportate tramite un commit alla repository dovranno avere una dimensione opportunamente scelta, commit troppo piccoli potrebbero risultare di scarsa utilità, mentre quelli troppo grandi non permettono di fornire sufficienti informazioni sullo stato del ciclo di vita del software;
- **Frequenza dei commit**: I commit effettuati dai membri del gruppo devono essere frequenti, poichè essi mantengono aggiornato il software con le modifiche più recenti e favoriscono il dialogo tra i membri del gruppo.

2.2.6.2 Metriche utilizzate

- **MP013:** Percentuale build superate
  - **Range di accettazione**: 60%;
  - **Range di ottimalità**: 80%.
- **MP014:** Media commit giornaliera
  - **Range di accettazione**: >=25;
  - **Range di ottimalità**: >=35.

2.2.7 QP007: Conformità dei requisiti

Per assicurare la conformità del prodotto finale(e dei requisiti individuati che ne fanno parte), il gruppo utilizzerà le seguenti metriche:

2.2.7.1 Metriche utilizzate

- **MP015:** Percentuale requisiti obbligatori soddisfatti
  - **Range di accettazione**: 100%;
  - **Range di ottimalità**: 100%.
- **MP016:** Percentuale requisiti desiderabili soddisfatti
  - **Range di accettazione**: >=50%;
  - **Range di ottimalità**: >=50%.

Capitolo 3

Qualità di Prodotto

3.1 Scopo

Per riuscire a garantire una buona qualità di prodotto, sono state individuate nello standard ISO/IEC 25010 le principali caratteristiche che i prodotti devono avere definendone le sotto-caratteristiche che le compongono e individuandone delle metriche adeguate per poter misurare ogni aspetto.

3.2 Qualità documento

Il team si impegna a produrre dei documenti di alta qualità, rispettando le seguenti caratteristiche.

3.2.1 Ortografia

3.2.1.1 Obbiettivi

Un documento, per essere privo di errori grammaticali e ortografici, viene controllato su diversi ambienti: durante la redazione, tramite il controllo automatico integrato nell'ambiente di lavoro; nel repository condiviso, tramite il correttore automatico eseguito da *Travis* CI_G (con notifica in caso di errori); durante la verifica, da parte di un *Verificatore*.
Le metriche utilizzate per la valutazione, definite nelle *Norme di Progetto* in Appendice B, sono le seguenti:

- **MPR001**: numero di errori ortografici
  - **Range di accettazione**: 0%;
  - **Range di ottimalità**: 0%.

3.2.2 Comprensibilità e leggibilità

3.2.2.1 Obbiettivi

Per misurare la leggibilità di un documento il gruppo ha scelto di utilizzare l' *Indice di Gulpease*$_G$. Questo viene calcolato automaticamente ogni volta che il documento viene modificato nel repository condiviso.
Le metriche utilizzate per la valutazione, definite nelle Norme di Progetto in Appendice B, sono le seguenti:

- **MPR002**: Indice di Gulpease
  - **Range di accettazione**: 40-100;
  - **Range di ottimalità**: 50-100.

3.2.3 Correttezza dei contenuti

3.2.3.1 Obbiettivi

La correttezza del documento è data anche dalla coerenza dei contenuti. Ogni membro del gruppo deve redigere dei buoni documenti, i verificatori devono controllarli e seguire le procedure definite nelle *Norme di Progetto v3.0.0*.
Le metriche utilizzate per la valutazione, definite nelle Norme di Progetto in Appendice B, sono le seguenti:

- **MPR003**: Numero di errori inerenti alla correttezza dei documenti
  - **Range di accettazione**: 80-100;
  - **Range di ottimalità**: 90-100.

3.2.4 Adesione alla norme interne

3.2.4.1 Obbiettivi

I documenti devono rispettare le *Norme di Progetto*. I *Verificatori* hanno il compito di avvisare il *Responsabile* come definito nelle *Norme di Progetto v3.0.0*.
Le metriche utilizzate per la valutazione, definite nelle *Norme di Progetto* in Appendice B, sono le seguenti:

- **MPR004**: Numero di errori inerenti alle *Norme di Progetto*
  - **Range di accettazione**: 85-100;
  - **Range di ottimalità**: 90-100.

3.3 Qualità del software

3.3.1 Funzionalità

Rappresenta la capacità del prodotto software di provvedere le funzionalità necessarie a soddisfare i requisiti individuati nel documento *Analisi dei Requisiti v3.0.0*.

3.3.1.1 Obbiettivi

Il prodotto dovrà possedere le seguenti caratteristiche:

- **Efficacia funzionale:** Indice che determina il grado di copertura dei requisiti;
- **Correttezza:** Indice che determina la correttezza dei risultati forniti dal software.

Le metriche utilizzate per la valutazione, definite nelle *Norme di Progetto* in Appendice B, sono le seguenti:

- **MPR005**: Completezza dell'implementazione funzionale
  - **Range di accettazione**: 100%;
  - **Range di ottimalità**: 100%.
- **MPR006**: Correttezza rispetto alle attese
  - **Range di accettazione**: 90%-100%;
  - **Range di ottimalità**: 100%.

3.3.2 Affidabilità

Rappresenta la capacità del prodotto software di svolgere correttamente le sue funzionalità mantenendo delle buone prestazioni al verificarsi di situazioni anomale.

3.3.2.1 Obbiettivi

Il prodotto dovrà possedere le seguenti caratteristiche:

- **Tolleranza agli errori**: Il $prodotto_G$ software continua a lavorare correttamente in presenza di errori dovuti a uno scorretto uso dell'applicativo;

- **Recuperabilità:** Nel caso in cui si presenta un'anomalia, l'applicativo è in grado di recuperare i dati e ripristinare lo stato interrotto.

Le metriche utilizzate per la valutazione, definite nelle Norme di Progetto in Appendice B, sono le seguenti:

- **MPR007**: Totalità di failure
  - **Range di accettazione**: 0%-10%;
  - **Range di ottimalità**: 0%.

3.3.3 Efficienza

Rappresenta la capacità di un prodotto software di realizzare le funzioni richieste nel minor tempo possibile e con l'uso del minimo numero di risorse necessarie.

3.3.3.1 Obbiettivi

Il prodotto dovrà possedere le seguenti caratteristiche:

- **Comportamento rispetto al tempo:** per svolgere le funzioni richieste il prodotto software deve fornire adeguati tempi di risposta ed elaborazione;
- **Utilizzo delle risorse:** il software nello svolgimento delle funzionalità deve utilizzare un appropriato numero e tipo di risorse.

Le metriche utilizzate per la valutazione, definite nelle Norme di Progetto in Appendice B, sono le seguenti:

- **MPR008**: Tempo di risposta
  - **Range di accettazione**: 0-8 sec;
  - **Range di ottimalità**: 0-3 sec.

3.3.4 Usabilità

L'usabilità rappresenta il grado di facilità e soddisfazione con cui si compie l'interazione tra l'uomo e il *prodotto*$_G$, ovvero l'efficacia, l'efficienza e la soddisfazione con le quali gli utenti raggiungono determinati obbiettivi in determinati contesti.

3.3.4.1 Obbiettivi

Il prodotto dovrà possedere le seguenti caratteristiche:

- **Apprendibilità:** livello di facilità con cui il prodotto può essere appreso dagli utenti per portare a termine determinati obiettivi con efficacia, efficienza, sicurezza e soddisfazione;
- **Comprensibilità:** livello a cui gli utenti riescono a riconoscere se il prodotto è adeguato per i loro bisogni;
- **Protezione dall'errore:** Rappresenta il grado con cui il *prodotto*$_G$ protegge l'utente dal commettere errori;
- **Estetica dell'interfaccia utente:** livello a cui un'interfaccia utente risulta piacevole per l'utente che la utilizza;
- **Accessibilità:** Si intende la possibilità di fornire i servizi anche a coloro che sono affetti da disabilità temporanee e non, che quindi utilizzano tecnologie ausiliarie. Nel caso dell'echo alcune disabilità sono per ora vincolanti in quanto presuppongono necessariamente l'utilizzo della voce.

Le metriche utilizzate per la valutazione, definite nelle *Norme di Progetto* in Appendice B, sono le seguenti:

- **MPR009**: Comprensibilità delle funzioni offerte
  - **Range di accettazione**: 75%-100%;
  - **Range di ottimalità**: 90%-100%.
- **MPR010**: Facilità di apprendimento
  - **Range di accettazione**: 0-20 min;
  - **Range di ottimalità**: 0-10 min.

3.3.5 Manutenibilità

Rappresenta la capacità del *prodotto*$_G$ di essere modificato tramite correzioni, miglioramenti e adattamenti.

3.3.5.1 Obbiettivi

Il prodotto dovrà possedere le seguenti caratteristiche:

- **Analizzabilità:** Il software deve poter essere analizzato per poter trovare gli errori;
- **Modificabilità:** Il prodotto deve permettere la modifica delle sue parti;
- **Modularità:** Il prodotto è diviso in parti che svolgono compiti ben precisi;
- **Riusabilità:** Le parti del software possono essere riusate in altre applicazioni;
- **Testabilità:** Il software deve essere testabile per consentire la validazione e l'approvazione di modifiche.

Le metriche utilizzate per la valutazione, definite nelle Norme di Progetto in Appendice B, sono le seguenti:

- **MPR011**: Capacità di analisi failure
  - **Range di accettazione**: 60%-100%;
  - **Range di ottimalità**: 80%-100%.
- **MPR012**: Impatto delle modifiche
  - **Range di accettazione**: 0%-20%;
  - **Range di ottimalità**: 0%-15%.

Appendice A

Specifica test

A.0.1 Test di Sistema

Id Test	Descrizione	Stato
TSFO1	Viene verificato che la connessione della skill al suo account Amazon possa avvenire correttamente	*Implementato*
TSFO2	Il gruppo esegue l'applicazione e controlla che il login funzioni correttamente	*Implementato*
TSFO15	Eseguire l'applicazione, creare un nuovo workflow, aggiunge dei blocchi a piacere e salva il workflow. Bisogna verificare che il workflow sia presente nel database	*Non Implementato*
TSFO16	Avviare l'applicazione ed eliminare un workflow	*Non Implementato*
TSFO17.1	avviare l'applicazione e modificare il nome di un workflow, poi verificare che sia stato effettivamente modificato nel database	*Non Implementato*
TSFO17.2	Il test consiste nell' eseguire l'applicazione e modificare un workflow aggiungendo un blocco, modificandone un altro ed eliminandone un terzo. In seguito verificare che il database sia stato effettivamente modificato.	*Non Implementato*
TSFO45	Viene verificato che il collegamento della skill all' account dell'utente Amazon avvenga correttamente	*Implementato*

Id Test	Descrizione	Stato
TSVO1	Viene verificato che l'applicazione Android funzioni correttamente in un dispositivo Android	*Non Implementato*

Tabella A.1: Test di Sistema

A.0.2 Tracciamento Requisiti-Test di Sistema

Requisito	Test
RFO1	TVFO1
RFO2	TVFO2
RFO15	TVFO15
RFO16	TVFO16
RFO17.1	TVFO17.1
RFO17.2	TVFO17.2
RFO45	TVFO45
RVO1	TVVO1

Tabella A.2: Tracciamento Requisiti-Test di Sistema

A.0.3 Test di Integrazione

Id Test	Descrizione	Stato
TI1	Viene testata l'applicazione Android ad ogni sua modifica nel repository Github, attraverso Travis-CI. L'applicazione deve compilare e non produrre warning.	*Implementato*
TI2	La skill android è soggetta a integrazione continua, attraverso Travis-CI.	*Non Eseguito*
TI3	La skill viene pubblicata in automatico (ad ogni modifica nel repository) in Aws Lambda e viene eseguito un test per controllare che funzioni. Questo test non assicura il funzionamento della skill, quindi è necessario un controllo umano (più controlli sono richiesti in caso di un numero alto di commit nel repository).	*Implementato*
TI4	Ad ogni commit nel repository, Travis-CI controlla che il collegamento tra la Skill e AWS API-Gateway funzioni. Questo viene fatto attraverso una semplice chiamata post a una funzione MOCK (per ridurre il traffico a dynamoDB e alle lambda).	*Implementato*
TI5	Controllare che il collegamento tra API-gateway, Lambda e DynamoDB funzioni, eseguendo dei test automatici appositi. Questa operazione deve essere eseguita poco frequentemente, in quanto potrebbe aumentare i costi dei servizi AWS.	*Implementato*
TI6	Viene creato un workflow contenente solamente blocchi con connettori e vengono valutati i risultati. I valori ritornati cambiano costantemente, quindi viene fatto un controllo solamente sul tipo dei dati ritornati.	*Implementato*

Id Test	Descrizione	Stato
TI7	Viene eseguita l'applicazione e viene controllato se le activities vengono rappresentate correttamente a schermo. Questo test è molto pesante, in quanto richiede l'esecuzione del simulatore Android, inoltre richiede supervisione di un membro del gruppo.	*Implementato*
TI8	Vengono create delle istanze di ogni blocco e vengono chiamati i loro metodi pubblici.	*Non Implementato*
TI9	Viene creato un workflow contenente tutti i blocchi, poi viene chiamato il metodo "response()" di workflow.	*Non Implementato*

Tabella A.3: Test di Integrazione

A.0.4 Tracciamento Test di Integrazione-Componenti

Test	Componente
TI1	`megalexa`
TI2	`MegAlexaSkill`
TI3	`MegAlexaSkill::lambda`
TI4	`MegAlexaSkill::lambda::connection`
TI5	`megalexa::adapters`
TI6	`megalexa::adapters::connectors`
TI7	`megalexa::activities`
TI8	`megalexa::models::blocks`
TI9	`MegAlexaSkill::lambda::blocks`

Tabella A.4: Tracciamento Test di Integrazione-Componenti

A.1 Test di unità

Id Test	Descrizione	Stato
TUS1	Viene controllato che ritorni un testo	*Implementato*
TUS2	Viene controllato che venga lanciata un'eccezione sul tipo di parametro passato	*Implementato*
TUS3	Viene controllato che venga lanciata un'eccezione sul valore di ritorno (check sui falsi positivi)	*Implementato*
TUS4	Viene controllato che ritorni una lista	*Implementato*
TUS5	Viene controllato che venga lanciata un'eccezione sul tipo di parametro passato	*non Implementato*
TUS6	Viene controllato che venga lanciata un'eccezione sul valore di ritorno	*Implementato*
TUS7	Viene controllato che ritorni una richiesta di inserimento pin	*Implementato*
TUS8	Viene controllato che ritorni una risposta di pin corretto	*Implementato*
TUS9	Viene controllato che ritorni una risposta di pin errato	*Implementato*
TUS10	Viene controllato che ritorni un workflow corretto	*Implementato*
TUS11	Viene controllato che ritorni un workflow non corretto	*Implementato*

Id Test	Descrizione	Stato
TUS12	Viene controllato che ritorni una eccezione sul parametro durante la creazione del workflow	*non Implementato*
TUS13	Viene controllato che ritorni un workflow corretto	*Implementato*
TUS13	Viene controllato che ritorni una eccezione sulla costruzione dell'utente	*Implementato*
TUS14	Viene controllato che ritorni un FeedRSS corretto	*Implementato*
TUS15	Viene controllato che venga lanciata un'eccezione sul tipo di parametro URL	*Implementato*
TUS16	Viene controllato che venga lanciata un'eccezione di Timeout sulla chiamata internet	*non Implementato*
TUS17	Viene controllato che ritorni una previsione corretta	*Implementato*
TUS18	Viene controllato che venga lanciata un'eccezione sul parametro position	*Implementato*
TUS19	Viene controllato che ritorni un testo vuoto	*Implementato*
TUS20	Viene controllato che venga lanciata un'eccezione sul parametro position	*Implementato*
TUS21	Viene controllato che venga lanciata un'eccezione di Timeout sulla chiamata internet alle API	*non Implementato*
TUS22	Viene controllato che ritorni una Tweet corretto	*Implementato*

Id Test	Descrizione	Stato
TUS23	Viene controllato che ritorni un Tweet vuoto	*Implementato*
TUS24	Viene controllato che venga lanciata un'eccezione sul parametro userID	*Implementato*
TUS25	Viene controllato che venga lanciata un'eccezione di Timeout sulla chiamata internet alle API di Twitter	*non Implementato*
TUS26	Viene controllato che ritorni un workflow vuoto	*non Implementato*
TUS27	Viene controllato che venga lanciata un'eccezione sulla mancanza di un blocco filtrabile dopo un filtro	*Implementato*
TUS28	Viene controllato che ritorni un workflow correttamente filtrato	*Implementato*
TUS29	Viene controllato che ritorni una lista correttamente filtrata	*Implementato*
TUS30	Viene controllato che ritorni un Feed RSS correttamente filtrato	*Implementato*
TUS31	Viene controllato che ritorni una lista di email correttamente filtrate	*non Implementato*
TUS32	Viene controllato che ritorni una lista di email corrette	*Non implementato*
TUS33	Viene controllato che ritorni una lista di email vuota	*Non implementato*

Id Test	Descrizione	Stato
TUS34	Viene controllato che venga lanciata un'eccezione sull'e-mail non corretta	*Non implementato*
TUS35	Viene controllato che ritorni una lista di Email corrette	*Non implementato*

Tabella A.5: Test di Unità

Id Test	Descrizione	Stato
TUA1	Viene controllato che l'uri sia un feedRSS	*Implementato*
TUA2	Viene controllato che un blocco filtrabile restituisca il numero filtrato di elementi	*Implementato*
TUA3	Viene controllato che un blocco non filtrabile non restituisca un numero filtrato di elementi	*Implementato*
TUA4	Viene controllato che venga aggiunto il giusto numero di blocchi ad un workflow	*Implementato*
TUA5	Viene controllato che il Database restituisca correttamente i dati di un utente	*Implementato*
TUA6	Viene controllato che ci sia un'unica istanza del model	*Implementato*
TUA7	Viene controllato che una città sia presente nella lista delle città gestite dalla API di Openweather	*Implementato*
TUA8	Viene controllato che un utente sia presente tra gli utenti di Twitter	*Implementato*
TUA9	Viene controllato che un hashtag sia un'espressione regolare valida	*Implementato*
TUA10	Viene controllato che l'account di Amazon Music sia valido	*Non implementato*
TUA11	Viene controllato che l'account di Google sia valido per il Calendar	*Non implementato*

Id Test	Descrizione	Stato
TUA12	Viene controllato che l'account di Google Mail sia valido per la lettura delle email	*Non implementato*
TUA13	Viene controllato che il numero di telefono sia associato ad un account Telegram	*Non implementato*
TUA14	Viene controllato che l'account Twitter dell'utente sai valido	*Non implementato*
TUA15	Viene controllato che il brano cercato su Amazon Music esista	*Non implementato*
TUA16	Viene controllato che la playlist cercata su Amazon Music esista	*Non implementato*
TUA17	Viene controllato che l'utente abbia inserito qualcosa nel testo del messaggio da inviare con Telegram	*Non implementato*
TUA18	Viene controllato che il destinatario del messaggio Telegram esista	*Non implementato*

Tabella A.6: Test di Unità

Id Test	Funzione
TUS1	Lambda::blocks::TextToSpeechBlock::text()
TUS2	Lambda::blocks::TextToSpeechBlock::text()
TUS3	Lambda::blocks::TextToSpeechBlock::text()
TUS4	Lambda::blocks::BlockList::text()
TUS5	Lambda::blocks::BlockList::text()
TUS6	Lambda::blocks::BlockList::text()
TUS7	Lambda::blocks::BlockPIN::text()
TUS8	Lambda::blocks::BlockPIN::text()
TUS9	Lambda::blocks::BlockPIN::text()
TUS10	Lambda::workflow::alexaResponse()
TUS11	Lambda::workflow::alexaResponse()
TUS12	Lambda::services::WorkflowService::create()
TUS13	Lambda::user::workflow()
TUS13	Lambda::user::workflow()
TUS14	Lambda::blocks::BlockFeedRSS::text()
TUS15	Lambda::blocks::BlockFeedRSS::text()
TUS16	Lambda::connectors::BlockFeedRSS::connect()
TUS17	Lambda::blocks::BlockWeather::text()
TUS18	Lambda::blocks::BlockWeather::text()
TUS19	Lambda::blocks::BlockWeather::text()
TUS20	Lambda::connectors::ConnectorWeather::connect()
TUS21	Lambda::connectors::ConnectorWeather::connect()
TUS22	Lambda::blocks::BlockTwitterRead::text()
TUS23	Lambda::blocks::BlockTwitterRead::text()
TUS24	Lambda::connectors::ConnectorTwitterRead::connect()
TUS25	Lambda::connectors::ConnectorTwitterRead::connect()
TUS26	Lambda::user::workflow()
TUS27	Lambda::user::filter()
TUS28	Lambda::user::filter()
TUS29	Lambda::blocks::BlockList::filterBlocks()

Id Test	Funzione
TUS30	Lambda::blocks::BlockList::filterBlocks()
TUS31	Lambda::blocks::BlockList::filterBlocks()
TUS32	Lambda::blocks::BlockEmailRead::text()
TUS33	Lambda::blocks::BlockEmailRead::text()
TUS34	Lambda::connectors::ConnectorEmailRead::connect()
TUS35	Lambda::connectors::ConnectorEmailRead::connect()

Tabella A.7: Test di Unità

Id Test	Funzione
TUA1	package com.megalexa::ConnectorFeedTest::valid()
TUA2	package com.megalexa::BlockFilterTest::withFilterableBlock()
TUA3	package com.megalexa::BlockFilterTest::withNonFilterableBlock()
TUA4	package com.megalexa::AppBuilderTest::valid()
TUA5	package com.megalexa::GetUserTest::valid()
TUA6	package com.megalexa::ModelSingletonTest::valid()
TUA7	package com.megalexa::ConnectorWeatherTest::valid()
TUA8	package com.megalexa::ConnectorTwitterTest::AccountValid()
TUA9	package com.megalexa::ConnectorTwitterTest::HashtagValid()
TUA10	package com.megalexa::ConnectorAmazonMusicTest::Valid()
TUA11	package com.megalexa::ConnectorCalendarTest::Valid()
TUA12	package com.megalexa::ConnectorReadMailTest::Valid()
TUA13	package com.megalexa::ConnectorTelegramTest::Valid()
TUA14	package com.megalexa::ConnectorTwitterTest::AccountValid()
TUA15	package com.megalexa::ConnectorAmazonMusicTest::Exist()
TUA16	package com.megalexa::ConnectorAmazonMusicTest::Exist()
TUA17	package com.megalexa::ConnectorTelegramTest::isEmpty()
TUA18	package com.megalexa::ConnectorTelegramTest::Valid()

Tabella A.8: Test di Unità

Appendice B

Resoconto delle attività di verifica

B.1 Analisi

Nel periodo antecedente la Revisione dei Requisiti sono stati verificati i documenti ed i processi applicando quanto descritto nelle *Norme di Progetto v3.0.0*.
L'analisi statica è stata effettuata secondo i criteri e le modalità indicate nelle *Norme di Progetto*.
Per gli errori riscontrati effettuando *walkthrough*$_G$, si è provveduto a correggere le anomalie riscontrate e sono stati riportati nella lista di controllo nelle *Norme di Progetto v3.0.0* per permettere di effettuare inspection successivamente.
L'*inspection*$_G$ viene effettuata utilizzando la lista di controllo precedentemente stilata.
Si sono poi calcolate le metriche descritte nelle *Norme di Progetto*.
L'avanzamento dei processi viene poi valutato secondo le metriche descritte nelle *Norme di Progetto*.

B.1.1 Verifica dei processi

Per il *processo*$_G$ di stesura dei documenti, il calcolo delle metriche di Budget Variance e di Schedule Variance è stato effettuato sul valore complessivo delle ore impiegate dal totale dei componenti del gruppo.
Per le successive fasi del *progetto*$_G$, il gruppo si propone di automatizzare il processo di calcolo delle ore impiegate, con il dettaglio puntuale dei singoli processi. Lo Schedule Variance totale è di -1 ore e il Budget Variance totale equivale a -25€.

B.1.2 Verifica dei documenti

Documento	Indice di Gulpease	Esito
Norme di Progetto	76	Superato
Piano di Progetto	64	Superato
Studio di Fattibilità	61	Superato
Analisi dei Requisiti	80	Superato
Piano di Qualifica	67	Superato
Glossario	68	Superato

Tabella B.1: Esito della verifica documenti

B.2 Revisione Analisi

Durante il breve periodo di Revisione Analisi, il gruppo si è preparato allo sviluppo del POC e ha apportato delle correzione ai documenti, migliorando i propri processi.

B.2.1 Verifica dei processi

I miglioramenti principali (tutti descritti nelle *Norme di Progetto v3.0.0*) sono stati:

- Automatizzato il calcolo delle ore di lavoro integrando $Harvest_G$ ad $Asana_G$;
- Automatizzato il calcolo dell'*Indice di Gulpease*$_G$, tramite script;
- Se dei documenti contenenti degli errori grammaticali raggiungono la repository, un bot avvisa per email chi ha commesso l'errore e invia una notifica al gruppo.

B.2.1.1 MP001 Schedule variance

Durante il periodo di Revisione Analisi si è verificato un aumento del ritardo dello stato del progetto, che dopo un certo punto si è stabilizzato, superando di molto la soglia di ottimalità prestabilita.

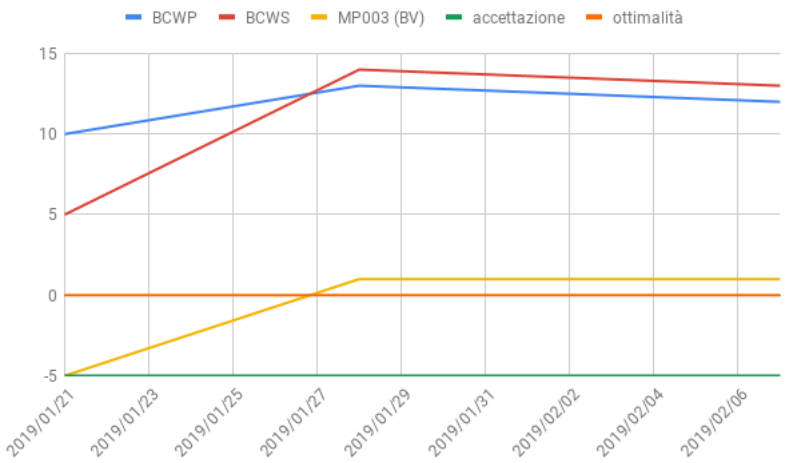

Figura B.1: *MP001 - Revisione Analisi*

B.2.1.2 MP002 Budget variance

Con l'aumento dello schedule variance di conseguenza anche il budget variance ha superato di molto le nostre aspettative.

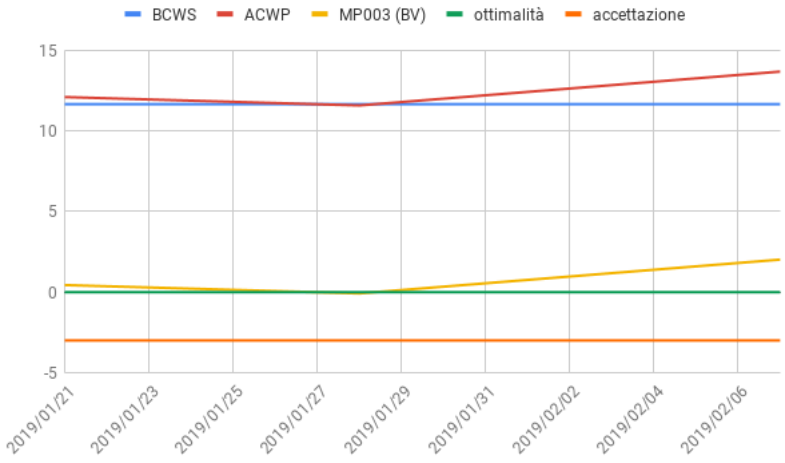

Figura B.2: *MP002 - Revisione Analisi*

B.3 Progettazione della base tecnologica

B.3.1 MP001: Schedule variance

A causa di un imprevisto successo all'interno del gruppo con la tecnologia *API Gateway*$_G$ c'è stato un notevole innalzamento dello schedule variance che ha raggiunto il picco massimo ai primi di marzo.

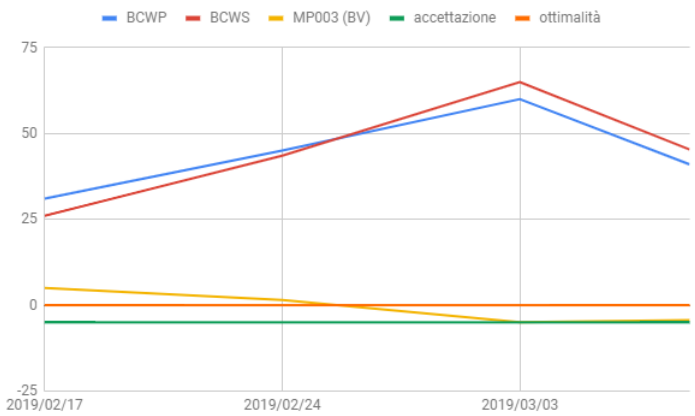

Figura B.3: *MP001 - Progettazione della base tecnologica*

B.3.2 MP002: Budget variance

In questo periodo il budget variance è stato più o meno sempre stabile ma comunque di molto al di sopra della soglia ottimale.

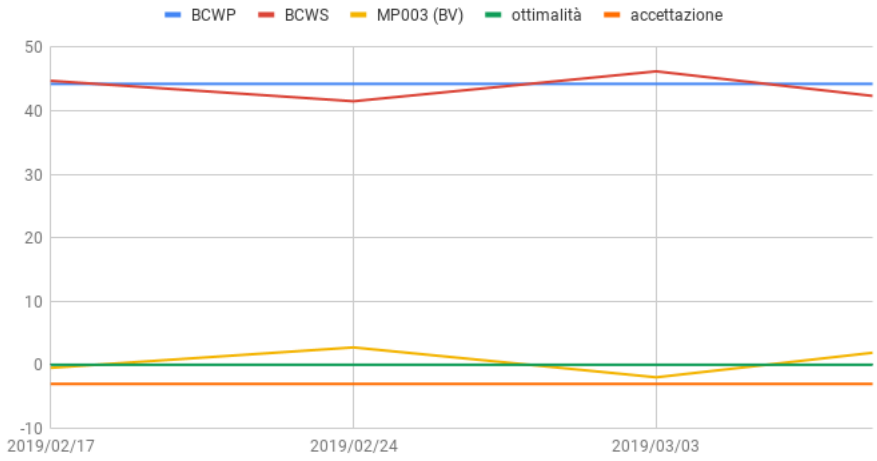

Figura B.4: *MP002 - Progettazione della base tecnologica*

B.3.3 MP003: SPICE capability level

Di seguito vengono riportati i livelli di maturità raggiunti dai processi eseguiti durante lo sviluppo del *Proof of Concept*$_G$. Data l'inesperienza, non viene raggiunto il livello di accettazione richiesto (3) per la maggior parte dei processi, ma il gruppo sta lavorando per migliorare.

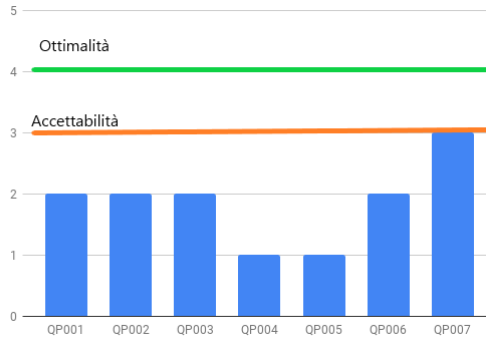

Figura B.5: *MP003 - ISO/IEC 15504*

B.3.4 MP005: Occorrenza rischi non previsti

Rischi non previsti: 2

- Un aggiornamento automatico di *AndroidStudio*$_G$ ha completamente rimosso una libreria utilizzata dall'applicazione mobile, quindi il gruppo ha perso tempo per implementane una alternativa. Questo è successo perché la libreria in questione era deprecata. Per evitare problemi simili, l'utilizzo di librerie deprecate è stato vietato, come descritto nelle *Norme di Progetto*;

- É stata inserita una chiave di accesso Amazon nel repository. Il gruppo è stato avvisato da Amazon, e ha dovuto creare nuove chiavi per tutti i membri.

B.3.5 MP006: Indisponibilità dei servizi

Indisponibilità dei servizi: 1
Durante il periodo di progettazione della base tecnologica, il gruppo non ha riscontrato problemi riguardanti il downtime di servizi esterni.

B.3.6 MP013: Percentuale build superate

Viene fatta distinzione tra Android e Skill, in quanto vengono contenute in repository diversi.
Le build non superate sono 24 su 134 per la Skill e 29 su 232 per Android. Entrambe superano il range di ottimalità (80%).

B.3.7 MP014: Media commit giornaliera

Come si può vedere dai grafici, il numero di commit è stato abbastanza costante, con un aumento del carico di lavoro durante la fine di febbraio.

B.3.8 MP015, MP016: Percentuale requisiti soddisfatti

B.3.9 MPR001 Ortografia

Grazie allo script per la segnalazione automatica degli errori, questi vengono corretti a ogni push nel develop. Durante la verifica, comunque, sono stati trovati da zero a due errori per documento passati allo script.

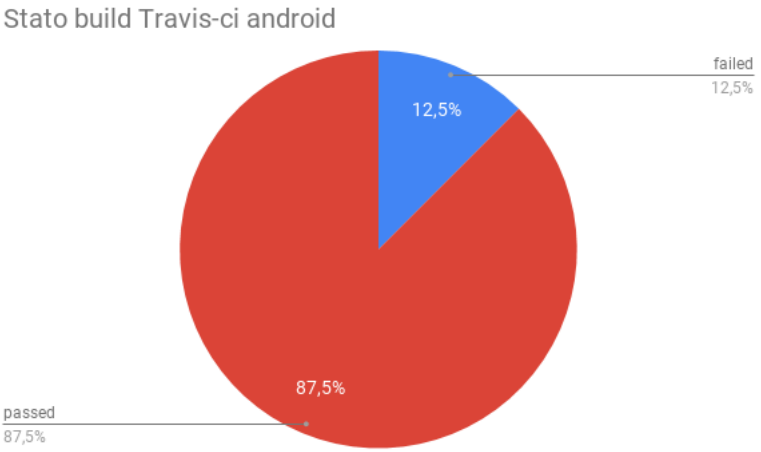

Figura B.6: *MP013 - Android - Progettazione della base tecnologica*

B.3.10 MPR002 Indice di Gulpease

B.4 Progettazione di dettaglio e codifica

B.4.1 Revisione complessiva

Durante il periodo di progettazione di dettaglio e codifica, il gruppo non ha trovato molte difficoltà a svolgere i compiti pianificati. Molte delle attività di codifica sono state supportate da script automatici e tool online attivati nel periodo precedente. C'è stata una buona collaborazione tra i membri del team. Il problema principale si è verificato per l'inesperienza del gruppo sul test driven development, che inizialmente ha rallentato la codifica, ma col tempo questo problema si è ridotto.

B.4.2 MP001: Schedule variance

Passato
La schedule variance è stata abbastanza pertinente con quanto preventivato, questo è dato dal fatto che non ci sono stati seri problemi durante il periodo di Progettazione di dettaglio e codifica.

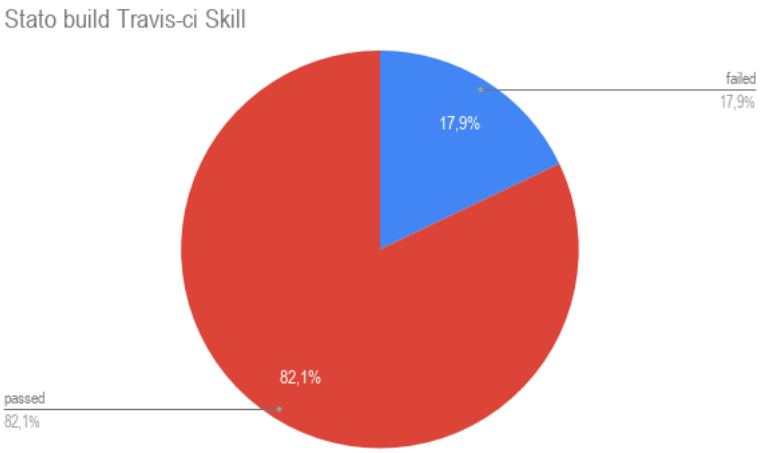

Figura B.7: *MP013 - Skill - Progettazione della base tecnologica*

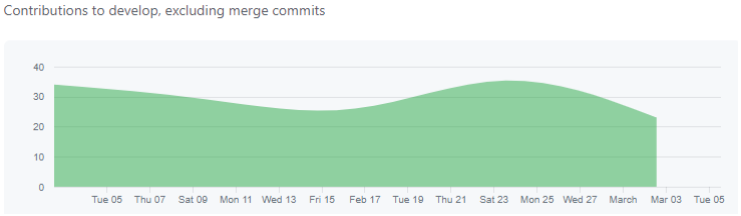

Figura B.8: *MP014 - Android - Progettazione della base tecnologica*

B.4.3 MP002: Budget variance

Passato
L'andamento della budget variance rispetta quello della schedule variance.

B.4.4 MP003: SPICE capability level

Passato 4/6

- **QP001:** Il gruppo è migliorato sulla Pianificazione dall'inizio del progetto. Dato che tutti hanno svolto almeno una volta ogni ruolo, onguno riesce a svolgere il suo compito. Se questo risulta più difficile di quanto pinificato, la persona in difficoltà crea un ticket su Asana o su Slack così gli altri membri del gruppo possono aiutarlo. Il responsabile riesce a controllare l'andamento del lavoro guardando Asana e le build su travis, senza dover aspettare una risposta da un collega;

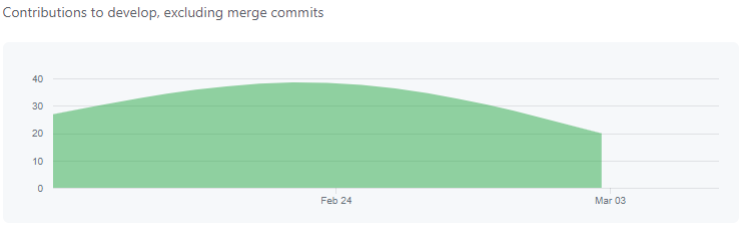

Figura B.9: *MP014 - Skill - Progettazione della base tecnologica*

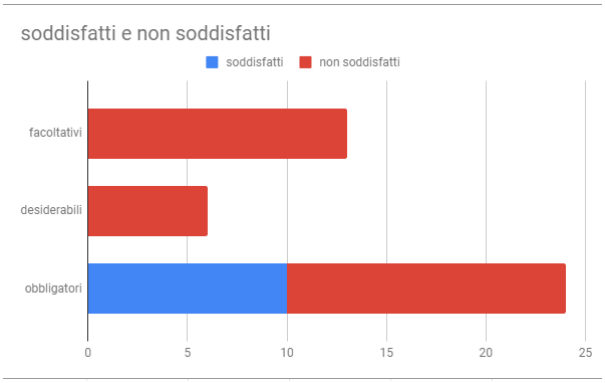

Figura B.10: *MP015 - MP016 Tipologia di requisiti*

- **QP002:** Il miglioramento delle attività di processo risulta ancora qualcosa di complesso da gestire. Questo richiede, nella maggior parte dei casi, una riunione con tutti i membri del gruppo, per prendere una decisione. Questo porta, spesso, alla modifica di un documento e comporta l'apprendimento del nuovo metodo. Detto questo, sono stati fatti alcuni miglioramenti preventivi avvenuti con successo, come per esempio il passaggio a typescript:
  - il team si è accorto che la produzione di codice per la backend risultava ostica usando javascript;
  - il responsabile ha programmato una riunione, nella quale sono stati discussi vantaggi del nuovo linguaggio, facilità per la traduzione (js -> ts), compatibilità con il codice già prodotto;
  - il team ha deciso di utilizzare TypeScript;
  - La traduzione è avvenuta in un paio d'ore e la velocità di produzione delle classi per la Skill è aumentata;
- **QP003:** I rischi sono mitigati dal fatto che il gruppo è composto da 7 membri, quindi, se qualcuno ha delle difficoltà a svolgere il proprio

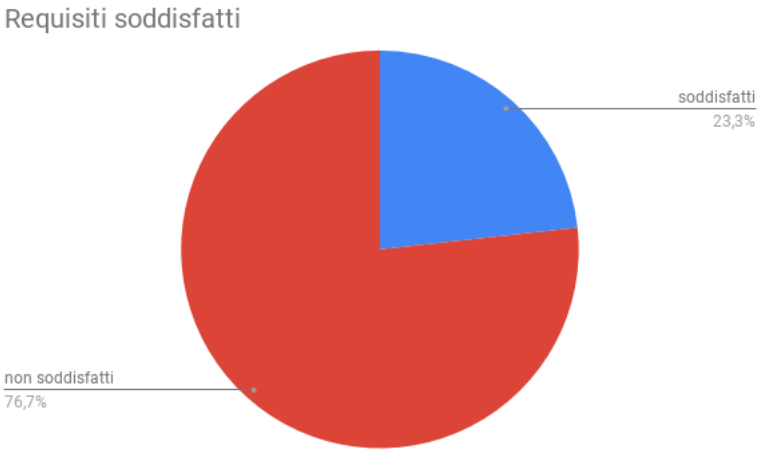

Figura B.11: *MP015 - MP016 Differenza soddisfatti e non soddisfatti*

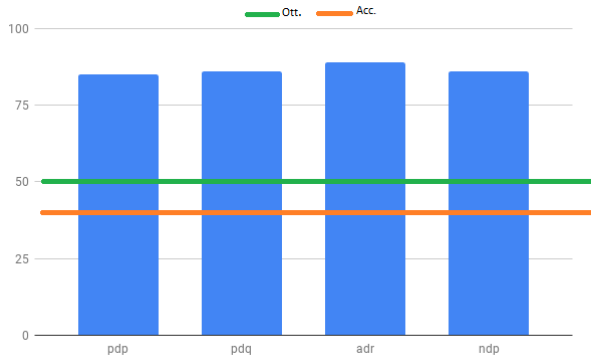

Figura B.12: *MPR002 Indice di Gulpease$_G$*

lavoro, è facile che un altro membro del gruppo sia disposto ad aiutarlo;

- **QP004:** La verifica del software risulta abbastanza semplice, dato che più membri del gruppo hanno esperienza (scolastica) nella produzione di test di unità e integrazione. Questo è supportato dal tool Travis-CI e dalla abbondante documentazione per i test su Node.js. Risulta però ancora ostico lavorare usando il Test Driven Development;

- **QP005:** I test di sistema, integrazione (lato API Gateway - database) risultano difficili da automatizzare, dato che molti dei tool automatici sono a pagamento. Il compito viene quindi svolto a mano, anche se il gruppo ha deciso di produrre uno script per eseguire test funzionali, entro il periodo di Verifica e Collaudo;

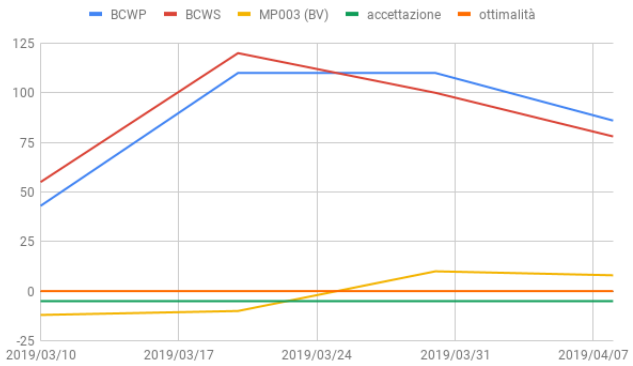

Figura B.13: *MP001 - schedule variance - Progettazione di dettaglio e codifica*

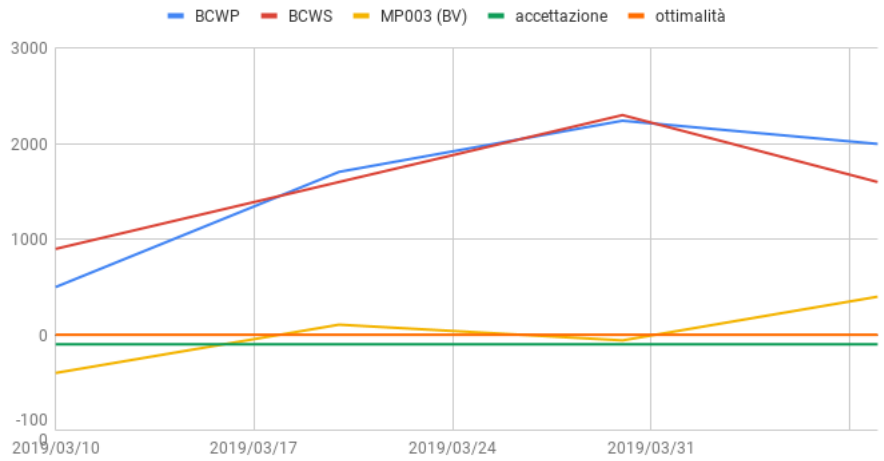

Figura B.14: *MP002 - budget variance - Progettazione di dettaglio e codifica*

- **QP006:** Il versionamento è completamente automatizzato tramite Travis-CI. Per l' applicazione Android è stato semplice, grazie a Gradle; la Skill, essendo scritta in typescript, ha bisogno di essere compilata in javascript, prima di essere pubblicata su AWS Lambda.
  Questo genera molti file .js che sporcano l'ambiente di lavoro. Per risolvere il problema, il gruppo ha prodotto degli script che si occupano di fare pulizia prima di trasferire i file su Github, e per fare il deploy automatico su AWS Lambda.

Di seguito viene riportata la tabella con il nome di ogni processo associato al suo codice per facilitare la lettura del grafico:

- QP001: Pianificazione delle attività di progetto, valutazione e controllo

dei processi §2.2.1;

- QP002: Miglioramento continuo delle attività di processo §2.2.2;
- QP003: Analisi e prevenzione dei rischi §2.2.3;
- QP004: Verifica del software §2.2.4;
- QP005: Gestione dei test §2.2.5;
- QP006: Versionamento e build §2.2.6.

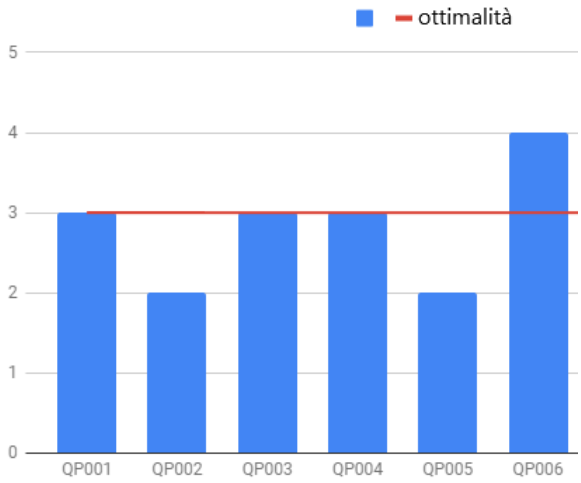

Figura B.15: *MP003 - ISO/IEC 15504 - Progettazione di dettaglio e codifica*

B.4.5 MP005: Occorrenza rischi non previsti

Passato
Rischi non previsti: 1

- **JavaScript:** grazie allo sviluppo del POC durante il periodo di Progettazione della base tecnologica, il gruppo si è accorto di come JavaScript fosse un ostacolo nella produzione della Skill, data la mancanza delle caratteristiche base dei linguaggi di programmazione (principalmente interfacce e tipi di ritorno). Per questo motivo è stato fatto il passaggio a TypeScript all'inizio del periodo di Progettazione di dettaglio e codifica.

B.4.6 MP006: Indisponibilità dei servizi

Passato

Indisponibilità dei servizi: 0

Durante il periodo di progettazione di dettaglio e codifica, PragmaDB ha avuto qualche giorno di downtime per non sforare con il piano gratuito di Amazon AWS. Questo non ha creato problemi, perché avevamo pieno controllo sul server su cui è hostato.

B.4.7 MP007 MP008 MP009 MP010 MP011

Passato

Queste metriche sono state usate per calcolare la manuntenibilità del codice, grazie alla piattaforma di test statici SonarCloud. Questo permette di visualizzare la qualità del software in un solo grafico ricco di informazioni.
Il seguente grafico presenta nelle ascisse le linee di codice, nelle ordinate il tempo previsto di risoluzione del problema, la grandezza del cerchio indica la quantità di linee di codice e il colore del cerchio corrisponde alla gravità del problema.
Tutti i problemi non in verde sono stati risolti, in quanto indicano un possibile errore grave durante l'esecuzione del codice. Quelli in verde verranno risolti durante il periodo di Verifica e Collaudo.

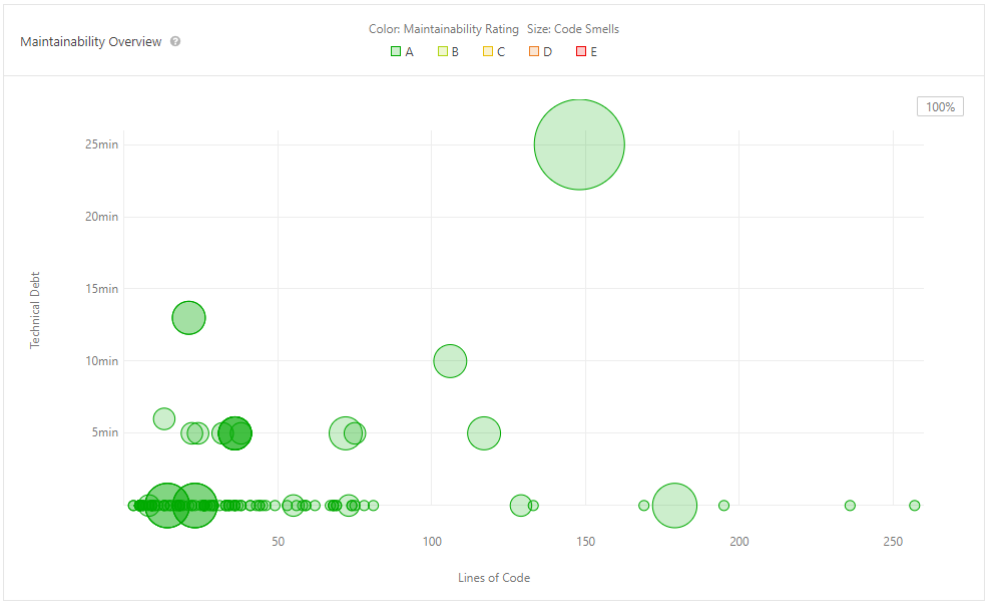

Figura B.16: *MP007 MP008 MP009 MP010 MP011 - manuntenibilità del codice app Android - Progettazione di dettaglio e codifica*

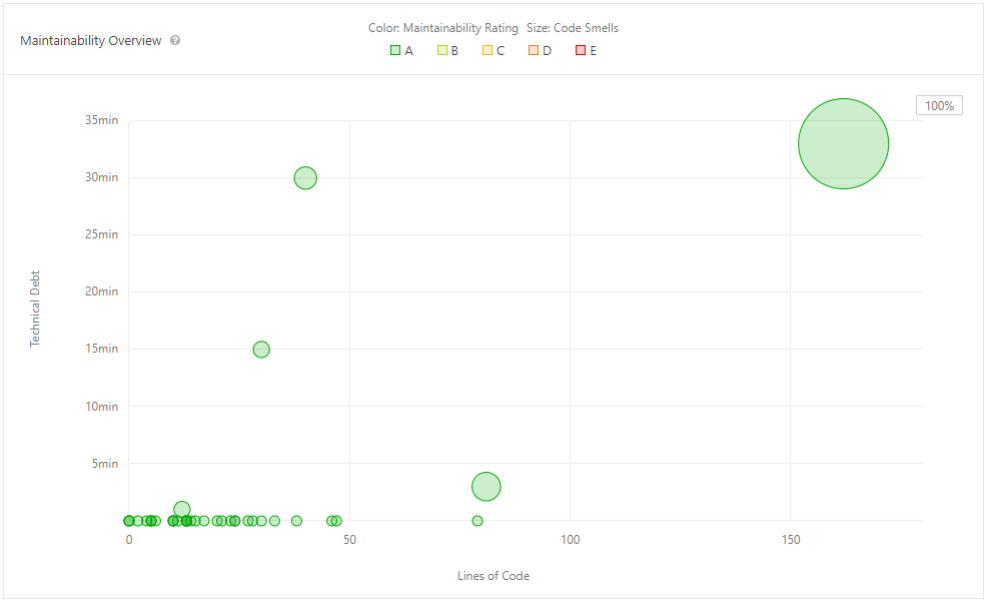

Figura B.17: *MP007 MP008 MP009 MP010 MP011 - manuntenibilità del codice Skill - Progettazione di dettaglio e codifica*

B.4.8 MP011: Tempo medio per risolvere un errore

Passato
tempo medio per risolvere un errore: 33.16 minuti.
La risoluzione degli errori nella Skill risulta molto lunga, in quanto bisogna fare ogni volta il deploy su AWS Lambda per testare le funzioni della skill (richiede da 40 secondi a 2 minuti). Per questo motivo, il team pone molta cura nella redazione dei test di unità e integrazione (che possono essere eseguiti senza deploy).

B.4.9 MP012: Efficienza della progettazione dei test

Passato
tempo medio per sviluppare un test: 35 minuti.
La progettazione di ogni test velocizza la seguente, in quanto hanno molti punti in comune tra di loro. Questo permette di velocizzare complessivamente la stesura dei test.

Figura B.18: *MP013 - Skill - Progettazione di dettaglio e codifica*

B.4.10 MP013: Percentuale build superate

Passato
Viene fatta distinzione tra Android e Skill, in quanto vengono contenute in repository diversi.
Le build dell'applicazione Android sono state per la maggior parte positive.
Le build della Skill (progetto node.js) risultano con molti errori. Questo è dovuto al fatto che sono stati sviluppati test di unità e integrazione prima di produrre il codice. Per questo motivo questa metrica è considerata come passata.

B.4.11 MP014: Media commit giornaliera

Passato
Il seguente grafico contiene il numero totale di commit giornalieri. Questi risultano poco omogenei, in quanto questo periodo è stato caratterizzato da cicli formati da progettazione di dettaglio (giornate con pochi commit) e codifica (molti commit).

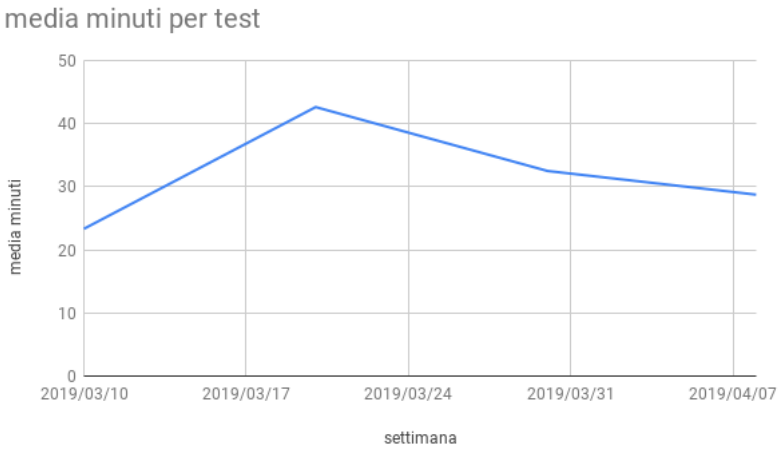

Figura B.19: *MP013 - media minuti per test - Progettazione di dettaglio e codifica*

B.4.12 MP015, MP016: Percentuale requisiti soddisfatti

Passato

Sono stati soddisfatti tutti i requisiti che il gruppo aveva preventivato per il periodo di progettazione di dettaglio e codifica, con l'eccezione del requisito riguardante la sveglia, che attualmente non è possibile implementare usando le API fornite da Alexa Skill Kit (dopo una riunione con la proponente, e con la sua approvazione, questo requisito è stato spostato in "facoltativo - non accettato"). Il gruppo ha stimato che i requisiti obbligatori non soddisfatti richiederanno poco lavoro, in quanto presentano aspetti simili ad altri requisiti (sotto il punto di vista della progettazione in dettaglio e codifica). I requisiti desiderabili e facoltativi sono per la maggior parte riguardanti aspetti secondari del prodotto (invece che blocchi), quindi compresi nelle attività di verifica e collaudo.
Come richiesto dal committente, molti dei requisiti sono stati scomposti in requisiti più specifici.
requisiti facoltativi sono stati non accettati, dato che il team li ha trovati difficili.

B.4.13 MPR001 Ortografia

Passato

Gli errori grammaticali trovati durante una ispezione finale sono stati meno

Figura B.20: *MP013 - Android - Progettazione di dettaglio e codifica*

di 2 per documento.

B.4.14 MPR002 Indice di Gulpease

Passato
Il seguente grafico è stato generato attraverso un bot che calcola giornalmente l'indice di gulpease dei documenti.
Si può notare come l'indice di gulpease è sempre stato sopra il limite di accettazione e ottimalità.

B.5 Verifica e collaudo

Questa sezione verrà compilata alla fine del periodo di Verifica e collaudo.

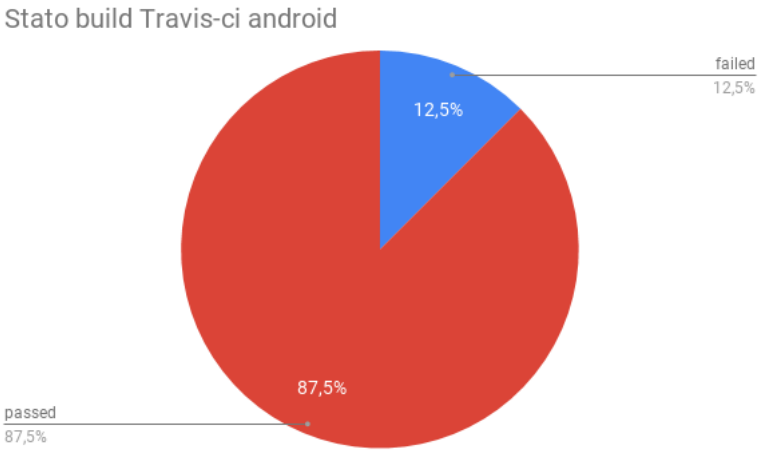

Figura B.21: *MP013 - Skill - Progettazione di dettaglio e codifica*

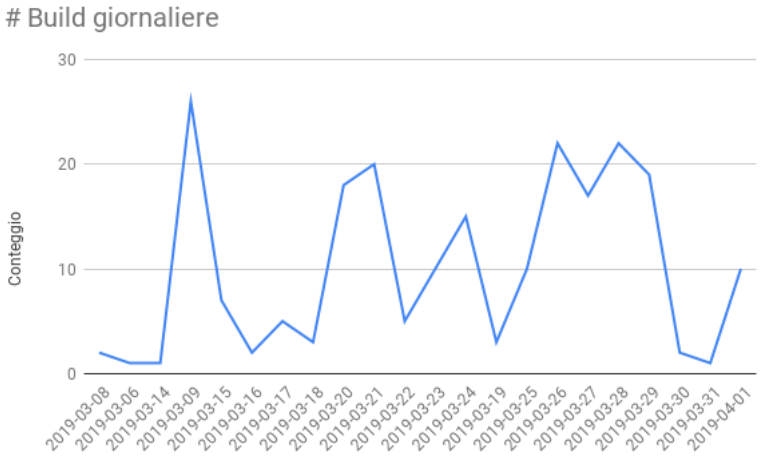

Figura B.22: *MP014 numero commit giornalieri - Progettazione di dettaglio e codifica*

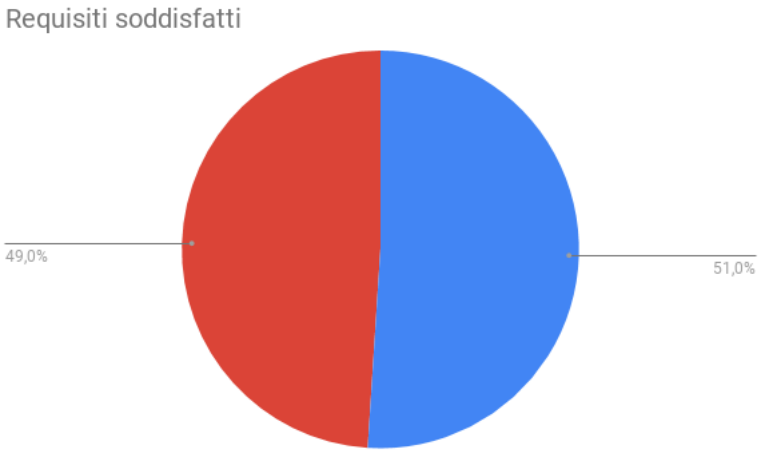

Figura B.23: *MP015 - MP016 Tipologia di requisiti - Progettazione di dettaglio e codifica*

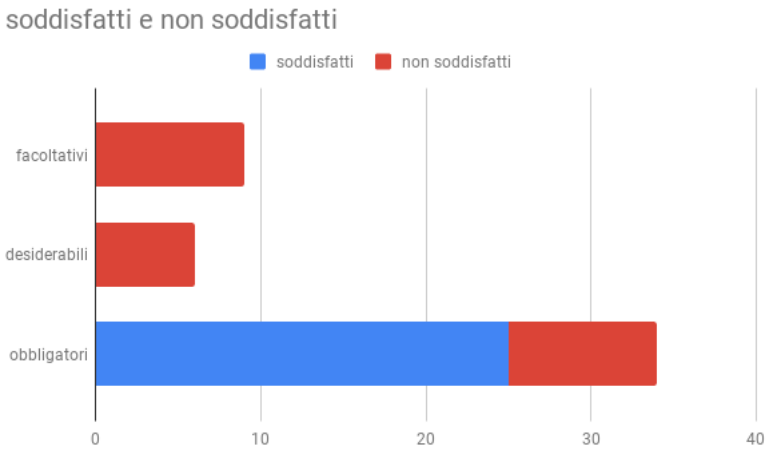

Figura B.24: *MP015 - MP016 soddisfatti / non soddisfatti - Progettazione di dettaglio e codifica*

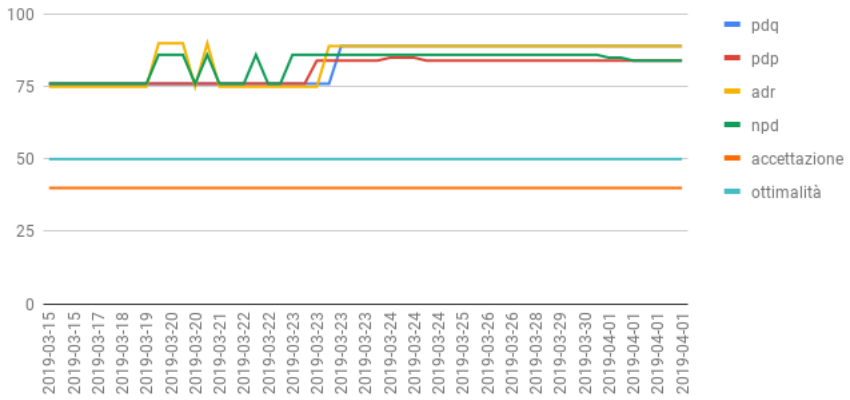

Figura B.25: *MPR002 Indice di Gulpease$_G$ - Progettazione di dettaglio e codifica*