Deliverable 1

Process Control Chart

GABRIELE QUATRANA 0306403

Sommario

- Introduzione
- Progettazione
- Process Control Chart
- Risultati
- Analisi
- Conclusioni
- Riferimenti

Introduzione

- L'obiettivo di questo deliverable è quello di misurare la stabilità di un attributo di progetto attraverso un **Process Control Chart**.
- Il **Process Control Chart** è un grafico utilizzato per studiare l'evoluzione temporale e la stabilità di un processo di sviluppo qualsiasi.
- Per costruire un **PCC** bisogna:
  - Selezionare l'asse Y: attributo da monitorare.
  - Selezionare l'asse X: metrica temporale considerata (giorni, mesi, anni, release, …)
- Una volta definito l'attributo di interesse, si valuta il suo comportamento rispetto all'andamento medio:
  - Si definiscono i limiti superiore e inferiore del grafico.
  - Si individuano gli **outlier**, ovvero i valori che sorpassano i limiti definiti in precedenza e che rappresentano un comportamento anomalo del processo.

Introduzione

- L'attività di **Software Analytics**, svolta tramite **PCC**, permette di controllare i processi in corso per trovare e correggere eventuali problemi che si verificano:
  - Possiamo individuare l'istante in cui si è verificato un problema.
  - Possiamo studiare l'andamento del processo nei periodi precedenti o successivi a tale anomalia.
  - Possiamo determinare le cause del problema individuato.
- Il progetto preso in considerazione per lo sviluppo del **Process Control Chart** è **QPID**:
  - La metrica temporale considerata sono i **mesi**.
  - L'attributo da analizzare considerato è il **numero di «Fixed New Feature»**.

Progettazione

- Per raccogliere i dati necessari alla costruzione del grafico, è stato sviluppato un software apposito in linguaggio Java.
- Il programma si occupa di:
  - Clonare il progetto da un repository **GitHub**.
  - Estrarre i commit dal progetto clonato.
  - Estrarre i ticket di tipo «Fixed New Feature» da **Jira**.
  - Eseguire il mapping tra commit e ticket.
  - Analizzare i ticket rimasti.
  - Salvare i dati ottenuti su un file *CSV*.
- Infine, il file *CSV* è stato importato in un foglio *Excel* per realizzare il **Process Control Chart**.

Progettazione – Git

- Per l'interazione con il progetto *Git* è stata utilizzata la libreria *JGit*:
  - Il programma clona la repository tramite il comando *clone()*:

```
private static void cloneProject(String projName) throws GitAPIException {
    if (!Files.exists(Paths.get(repoDir))) {
        String url = "https://github.com/apache/" + projName.toLowerCase();
        Git git = Git.cloneRepository().setURI(url).setDirectory(new File(repoDir)).call();
        git.close();
    }
}
```

  - Dopodiché, estrae tutti i commit dal repository clonato:

```
public static List<RevCommit> getAllCommits(Path repoPath) throws IOException, GitAPIException {
    List<RevCommit> commits = new ArrayList<>();
    try (Git git = Git.open(repoPath.toFile())) {
        Iterable<RevCommit> logs = git.log().all().call();
        for (RevCommit commit : logs) {
            commits.add(commit);
        }
    }

    return commits;
}
```

Progettazione – Jira

- Attraverso le **Rest API** messe a disposizione da *Jira*, vengono estratti tutti ticket di tipo «Fixed New Feature»:

```
// Rest API Query to get all the Fixed New Features
String url = "https://issues.apache.org/jira/rest/api/2/search?jql=project=%22" + projName
        + "%22AND%22issueType%22=%22New%20Feature%22AND(%22status%22=%22closed%22OR"
        + "%22status%22=%22resolved%22)AND%22resolution%22=%22fixed%22&fields=key,resolutiondate,versions,created&startAt="
        + i.toString() + "&maxResults=" + j.toString();
```

- Per ognuno dei ticket ottenuti, viene istanziato un nuovo oggetto *Ticket*, che mantiene l'ID e la data di creazione del ticket:

```
// Create the Ticket Object from JSON and add it to the list
for (; i < total && i < j; i++) {
    String date = issues.getJSONObject(i % 1000).getJSONObject("fields").get("resolutiondate").toString()
            .substring(0, 7);
    String key = issues.getJSONObject(i % 1000).get("key").toString();
    Ticket entry = new Ticket(key, date);
    tickets.add(entry);
}
```

Progettazione – Git & Jira

- Per effettuare il mapping tra i ticket *Jira* e i commit *Git*, si eliminano tutti i ticket che non hanno commit associati:

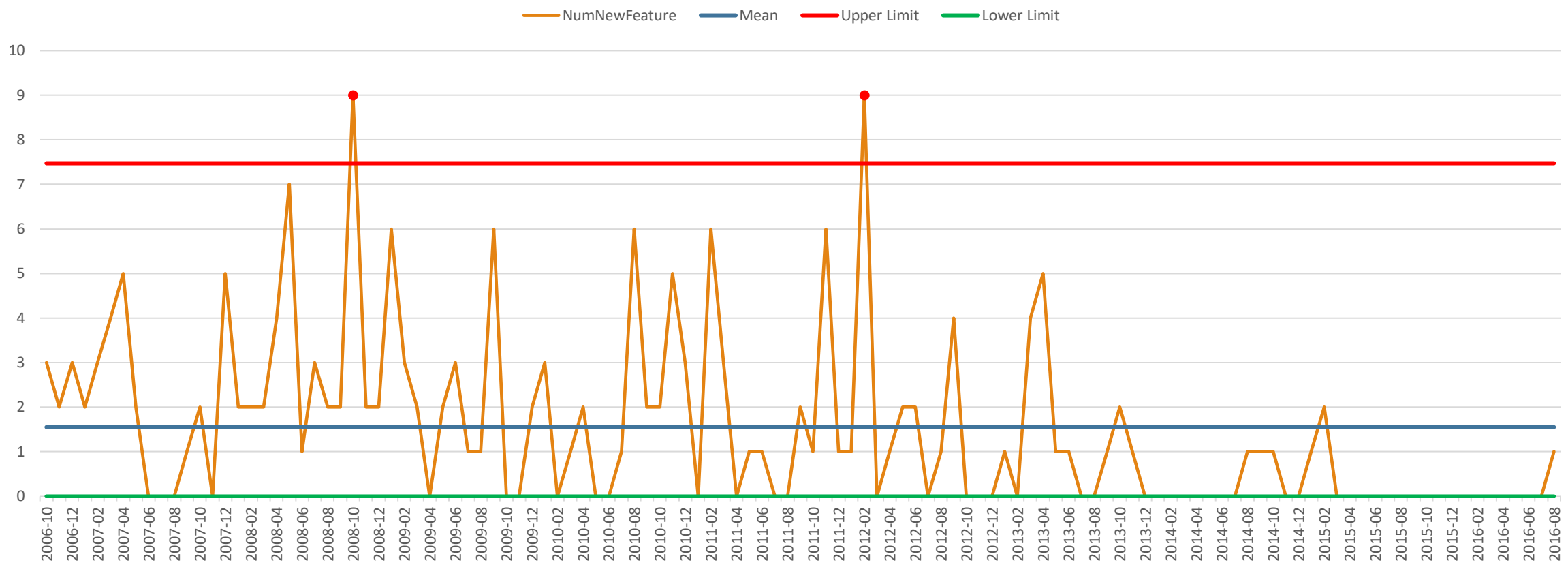
```
for (RevCommit commit : commits) {
    String message = commit.getFullMessage();
    if (message.contains(ticketID +",") || message.contains(ticketID + " ") || message.contains(ticketID + ":")
            || message.contains(ticketID + ".") || message.contains(ticketID + "\r") || message.contains(ticketID + "_")
            || message.contains(ticketID + "-") || message.contains(ticketID + "\n") || message.contains(ticketID + ")")
            || message.contains(ticketID + "/") || message.endsWith(ticketID) || message.contains(ticketID + "])) {

        goodTickets.add(ticket);
    }
}
```

- In seguito, il programma costruisce una lista che contiene i mesi compresi tra la data del primo ticket e dell'ultimo ticket.
- Per ogni mese, viene contato il numero di volte che questo appare nella lista dei ticket.
- Infine, i risultati ottenuti vengono salvati su un file *CSV*.

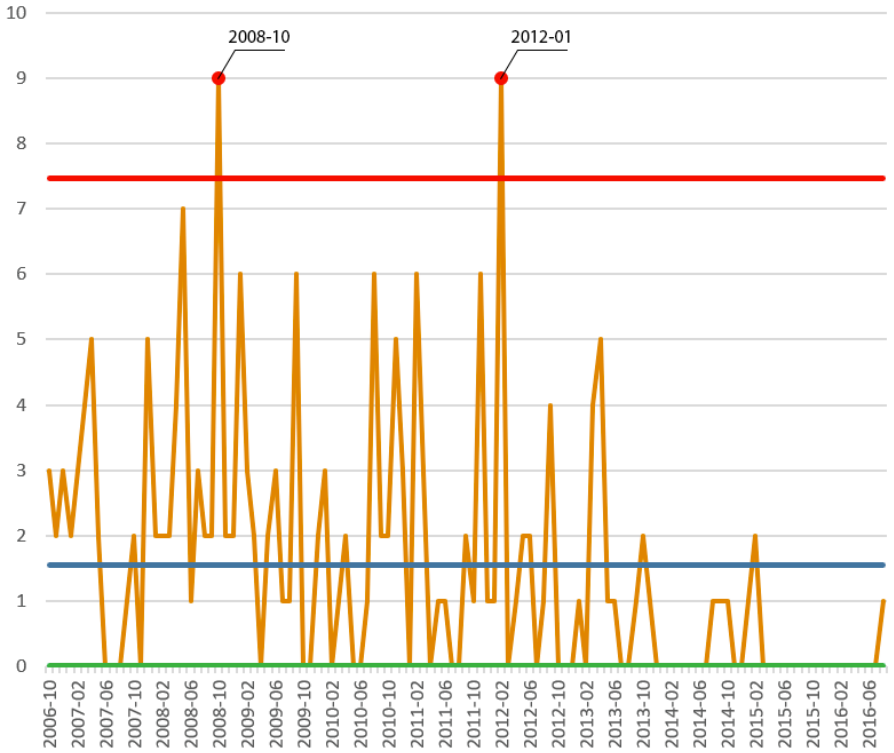
Progettazione – Grafico

- Il file *CSV* ottenuto è stato importato in un foglio *Excel* per poter costruire il grafico.
- Sono stati calcolati i seguenti dati:
  - Numero medio di «Fixed New Feature» tra tutti i mesi considerati: ***MEAN***.
  - Deviazione standard dei dati ottenuti: ***STDV.***
  - Limite superiore: ***Upper Limit = MEAN + 3*STDV.***
  - Limite inferiore: ***Lower Limit = max{0; MEAN – 3*STDV}.***
- I valori calcolati in precedenza sono stati inseriti nel grafico per visualizzare graficamente l'andamento dei vari punti.

Process Control Chart

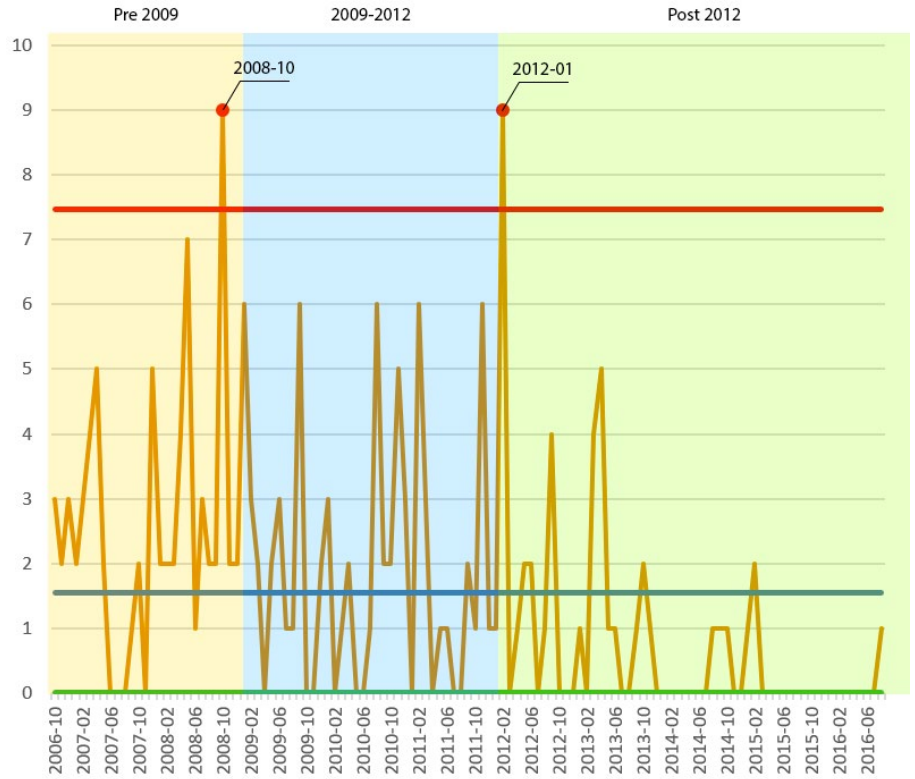


Process Control Chart che rappresenta il numero di «Fixed New Feature» per mese del progetto QPID



Risultati

- 185 ticket di tipo «Fixed New Feature».
- Periodo di osservazione di 119 mesi (2006-10/2016-08).
- Si può notare che il grafico è generalmente stabile:
  - Solo **2** punti oltrepassano il limite superiore del grafico.
  - **2008-10**: superiore di circa 1,5 rispetto al limite.
  - **2012-01**: superiore di circa 1,5 rispetto al limite.



Analisi

- Analizzando il grafico ottenuto, si può suddividere il processo di sviluppo in tre periodi:
  - Prima del 2009:
    - Periodo in cui si presenta il primo **outlier**.
    - L'andamento generale di discosta molto dalla media.
  - Tra 2009 e 2012:
    - Periodo più stabile dei tre.
    - Le oscillazioni tendono a essere più vicine alla media.
    - Nessun valore esce dai limiti.
  - Dopo il 2012:
    - Periodo in cui si presenta il secondo **outlier**.
    - In un primo momento l'andamento sembra essere abbastanza regolare.
    - Nella parte finale del periodo, la curva tende a stabilizzarsi verso il limite inferiore.

Conclusioni

- Nel periodo **pre-2009** si può osservare un andamento abbastanza instabile con un'oscillazione elevata nell'ultima parte:
  - Essendo il periodo di inizializzazione del progetto, sono state sviluppate un numero maggiore di nuove feature.
- Nel periodo **tra 2009 e 2012** si può osservare un andamento molto regolare:
  - L'applicazione è entrate in una fase di sviluppo costante in cui le feature principali sono state implementate.
  - Altre feature vengono implementate sporadicamente.
- Nel periodo **post-2012** si può osservare una diminuzione dell'attività di sviluppo:
  - Nella parte iniziale del periodo osserviamo un valore anomalo che potrebbe essere dato dal rilascio imminente di una nuova versione del software.
  - Nella seconda parte riscontriamo una diminuzione sostanziale dell'attività di sviluppo.
  - Questo potrebbe essere il risultato della cessazione dello sviluppo del progetto o della sostituzione dell' **Issue Tracking System**.

Riferimenti

- GitHub: https://github.com/gabrielequatrana/Deliverable1
- Travis CI: https://app.travis-ci.com/gabrielequatrana/Deliverable1
- SonarCloud: https://sonarcloud.io/dashboard?id=gabrielequatrana_Deliverable1