Deliverable 1
Process Control Chart

- Jacopo Fabi 0293870

Introduzione

- L'obiettivo del progetto è quello di realizzare un Process Control Chart per uno specifico attributo di processo, relativamente al progetto Falcon.
- Il Process Control Chart è uno strumento statistico che permette di monitorare ed analizzare l'evoluzione temporale e la stabilità di un qualsiasi processo di sviluppo rispetto ad uno specifico attributo
  - Sull'asse X troviamo la metrica temporale considerata (#Revisione, Mese, Anno …)
  - Sull'asse Y troviamo il valore dell'attributo da monitorare

- L'attributo selezionato è il numero di ticket Jira fixati
- Per valutare la stabilità del processo rispetto al numero di ticket Jira fixati, si va ad analizzare il comportamento nel tempo rispetto all'andamento medio
  - Si definiscono Upper Limit e Lower Limit
  - Si individuano gli outlier, ovvero i valori che escono da tali limiti

Progettazione

- Per analizzare il processo, è stato sviluppato un programma Java che si occupa di:
  1. Ottenere i ticket di tipo ‘fix bug’ da Jira
  2. Ottenere i commit da GitHub
  3. Mappare Commit Git e Ticket Jira
  4. Analizzare i ticket e salvare il risultato del report su un file .csv

Progettazione – Jira Ticket

- Tramite le REST API di Jira, si recuperano tutti i ticket relativi a bug fixati:

```
String url = "https://issues.apache.org/jira/rest/api/2/search?jql=project=%22" + projName
+ "%22AND%22issueType%22=%22Bug%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();
```

- Ogni ticket corrisponde ad un oggetto JiraTicket, identificato dai campi key e date, quest'ultimo fondamentale per individuare i bug fixati in ogni mese di vita del progetto:

```
for (; i < total && i < j; i++) {
    String date = issues.getJSONObject(i % 1000).getJSONObject("fields").get("resolutiondate").toString();
    String key = issues.getJSONObject(i % 1000).get("key").toString();
    JiraTicket entry = new JiraTicket(key, date);
    fixedBugs.add(entry);
}
```

- Viene restituito l'array contenente tutti i JiraTicket trovati

```
return fixedBugs;
```

Progettazione – Git Commit

- In aggiunta alle richieste, si realizza il mapping Jira-Git, sfruttando la libreria JGit per il recupero da Github della lista dei commit:
  - Si apre in locale la repository tramite i comandi *clone* o *checkout*

```
if (!Files.exists(Paths.get(repoDir))) {
    this.git = Git.cloneRepository()
    .setURI(gitUrl + projectName + ".git")
    .setDirectory(new File(repoDir))
    .call();
} else {
    try (Git gitRepo = Git.open(new File( repoDir + "/.git"))){
        this.git = gitRepo;
        gitRepo.checkout().setName(this.getDefaultBranch()).call();
        gitRepo.pull().call();
    }
}
```

  - Si ottiene la lista dei commit effettuati sul master tramite il comando *log*, andando ad istanziare un oggetto GitCommit per ogni commit individuato, recuperando la lista di tutti i commit trovati

```
this.commits = new ArrayList<>();
    Iterable<RevCommit> commitsLog = null;

    this.git.checkout().setName(this.getDefaultBranch()).call();
    commitsLog = git.log().call();

    for (RevCommit commit : commitsLog) {
        this.commits.add(new GitCommit(commit.getId(), new Date(commit.getCommitTime() * 1000L),
commit.getFullMessage()));
    }
    return this.commits;
```

Progettazione – Merging Git w/ Jira

- Si effettua il mapping di ogni JiraTicket con i relativi GitCommit identificati:
  - Si imposta sul JiraTicket la stessa data ottenuta dal GitCommit, rendendo coerenti le informazioni

```
public static void bindJiraToGit(List<JiraTicket> tickets) {
    GitAPI git = new GitAPI(GIT_PROJECT_NAME, "output/");
    git.init();
    List<GitCommit> commits = git.getCommits();

    for (GitCommit c : commits) {
        for (JiraTicket t : tickets) {
            if (c.getComment().contains(t.getKey()) && (t.getDate().compareTo(c.getDate()) < 0)) {
                t.setDate(c.getDate());
            }
        }
    }
}
```

Progettazione – Analisi Ticket e Report

- Recuperiamo tutti i mesi dove ci sono dei JiraTicket validi:

```
List<String> months = new ArrayList<>();
for (JiraTicket t : tickets) {
    if (!months.contains(t.getMonth())) {
    months.add(t.getMonth());
    }
}
```

- Per ogni mese selezionato, si contano le occorrenze all'interno della lista di JiraTicket, in modo da ottenere il numero di 'bug fix' per ogni mese di vita del progetto:

```
for (String m : months) {
    int count = 0;
    for (JiraTicket t : fixedBugs) {
        if (t.getMonth().equals(m)) {
        count++;
    }
}
```

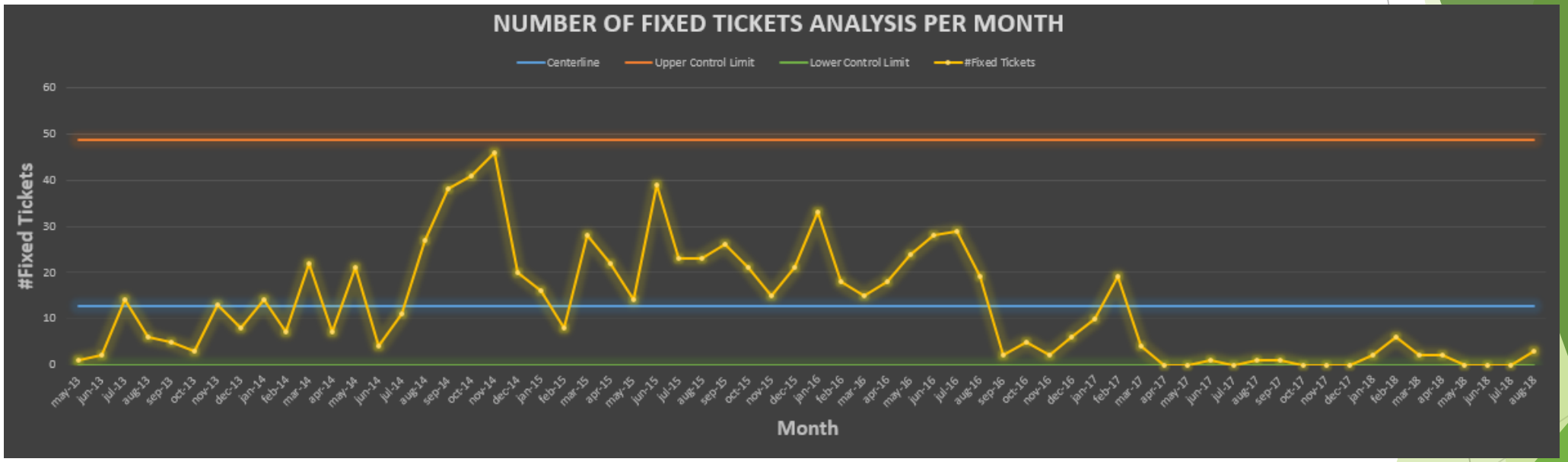
- Il report generato viene scritto su un file chiamato 'FALCONprocessControlData.csv':

	A	B	C
1	Month	#Bug Fixed	
2	2017-6	1	
3	2015-8	23	
4	2015-7	23	
5	2013-9	5	

Progettazione – Process Control Chart

- I risultati salvati sul file '.csv' sono stati riportati su un foglio Excel così da poter calcolare tutti i valori necessari alla realizzazione del Process Control Chart:
  - **Media** di bug fixed in ogni mese E
  - Deviazione standard del campione σ
  - **Lower limit:** $\max\{0; E - 3\sigma\}$
  - **Upper limit:** $E + 3\sigma$
- Per realizzare il grafico, inoltre, è necessario definire:
  - Asse X: mesi
  - Asse Y: numero di bug fixed

Process Control Chart



Risultati

- Periodo di osservazione: 64 mesi
- # di Ticket ‘bug fix’: 816

- Tramite il grafico controlliamo la stabilità del ‘numero di tickets risolti’ per ogni mese di sviluppo del progetto.
- L’intervallo temporale di analisi va da Maggio 2013 ad Agosto 2018, momento in cui il progetto Falcon è stato archiviato.

- Nessun punto è al di fuori del Process Control Chart, di conseguenza possiamo dire che, in questo determinato intervallo temporale, nessun mese ha un ‘numero di tickets risolti’ particolarmente significativo.