

# Fundamentals of Software Systems (FSS)

## Software Evolution – Part 02 Assignment

### Submission Guidelines

To correctly complete this assignment you must:

- Carry out the assignment in a team of 2 to 4 students.
- Carry out the assignment with your team only. You are allowed to discuss solutions with other teams, but each team should come up its own personal solution. A strict plagiarism policy is going to be applied to all the artifacts submitted for evaluation.
- As your submission, upload the filled Jupyter Notebook (including outputs) together with the d3 visualization web pages (i.e. upload everything you downloaded including the filled Jupyter Notebook plus your `output.json`)
- The files must be uploaded to OLAT as a single ZIP ( `.zip` ) file by Dec 12, 2022 @ 23:55.

### Group Members

- Firstname, Lastname, Immatriculation Number
- **TO BE FILLED**

### Task Context

In this assignment we will be analyzing the *elasticsearch* project. All following tasks should be done with the subset of commits from tag `v1.0.0` to tag `v1.1.0`.

Website: <https://github.com/elastic/elasticsearch> Repository: <https://github.com/elastic/elasticsearch.git>

In [1]:

```
from enum import Enum

class Modification(Enum):
    ADDED = "Lines added"
    REMOVED = "Lines removed"
    TOTAL = "Lines added + lines removed"
    DIFF = "Lines added - lines removed"
```

### Task 1: Author analysis

In the following, please consider only `java` files.

The first task is to get an overview of the author ownership of the *elasticsearch* project. In particular, we want to understand who are the main authors in the system between the two considered tags, the authors distribution among files and the files distribution among authors. To this aim, perform the following:

- create a dictionary (or a list of tuples) with the pairs author => number of modified files

- create a dictionary (or a list of tuples) with the pairs file => number of authors who modified the file
- visualize the distribution of authors among files: the visualization should have on the x axis the number of authors per file (from 1 to max), and on the y axis the number of files with the given number of authors (so for example the first bar represent the number of files with single author)
- visualize the distribution of files among authors: the visualization should have on the x axis the number of files per author (from 1 to max), and on the y axis the number of authors who modified the given number of files (so for example the first bar represent the minor contributors, i.e., the number of authors who changed only 1 file)

Comment the two distribution visualizations.

Now, let's look at the following 3 packages in more detail:

1. `src/main/java/org/elasticsearch/search`
2. `src/main/java/org/elasticsearch/index`
3. `src/main/java/org/elasticsearch/action`

Create a function that, given the path of a package and a modification type (see class `Modification` above), returns a dictionary of authors => number, where the number counts the total lines added or removed or added+removed or added-removed (depending on the given `Modification` parameter), for the given package. To compute the value at the package level, you should aggregate the data per file.

Using the function defined above, visualize the author contributions (lines added + lines removed). The visualization should have the author on the x axis, and the total lines on the y axis. Sort the visualization in decreasing amount of contributions, i.e., the main author should be the first.

Compare the visualization for the 3 packages and comment.

In [ ]:

## Task 2: Knowledge loss

We now want to analyze the knowledge loss when the main contributor of the analyzed project would leave. For this we will use the circle packaging layout introduced in the "Code as a Crime Scene" book. It should show how much of each file was written by the main contributor of *elasticsearch* (according to the analysis above using `Modification.TOTAL`) and indicate which areas would be affected most when this contributor leaves the project. This assignment includes the necessary `knowledge_loss.html` file as well as the `d3` folder with the d3 dependencies. Your task is to create the `output.json` file according to the specification below. This file can then be visualized with the files provided.

For showing the visualization, once you have the output as `output.json` you should

- make sure to have the `knowledge_loss.html` file in the same folder
- start a local HTTP server in the same folder (e.g. with python `python3 -m http.server`, serving necessary for d3)
- open the served `knowledge_loss.html` and look at the visualization

For testing, you can use the provided `output.json` and should see a circle packaging layout with two circles, one big red, and one small white-red.

For the package you identify as the worst in terms of knowledge loss, investigate the author contributions using the function defined in the previous exercise and comment how the situation is, e.g. how big the gap between the main author and the second biggest contributor for the selected package is.

## Output Format for Visualization

Example:

- `root` is always the root of the tree
- `size` should be the total number of lines of contribution
- `weight` can be set to the same as `size`
- `ownership` should be set to the percentage of contributions from the main author (e.g. 0.98 for 98% if contributions coming from the main author)

```
{
  "name": "root",
  "children": [
    {
      "name": "test",
      "children": [
        {
          "name": "benchmarking",
          "children": [
            {
              "author_color": "red",
              "size": "4005",
              "name": "t6726-patmat-analysis.scala",
              "weight": 1.0,
              "ownership": 0.9,
              "children": []
            },
            {
              "author_color": "red",
              "size": "55",
              "name": "TreeSetIterator.scala",
              "weight": 0.88,
              "ownership": 0.2,
              "children": []
            }
          ]
        }
      ]
    }
  ]
}
```

## JSON Export

For exporting the data to JSON you can use the following snippet:

```
import json

with open("output.json", "w") as file:
    json.dump(tree, file, indent=4)
```

In [ ]:

## Task 3: Code Churn Analysis

The third and last task is to analyze the code churn of the *elasticsearch* project. For this analysis we look at the code churn, meaning the daily change in the total number of lines of the project. Visualize the code churn over time bucketing the data by day. Remember that you'll need to fill the gaps for days when there are no commits. Chose a filling strategy and justify it.

Look at the churn trend over time and identify two outliers. For each of them:

- identify if it was caused by a single or multiple commits (since you are bucketing the data by day)
- find the hash of the involved commit(s)
- find the involved files
- look at the actual diff

Based on the above, discuss if the outlier is a false positive or should be a reason for concern.

In [ ]: