Summary of the Paper

Goal:

Reveal the relationship between the **developer-related characteristics** and **Technical Debt** in order to serve the software development and software quality control. The result will be helpful to management level, personal level as well as research level.

Data sources:

Details of **commits** from OSS [2.1] by developers with a variety of skills, seniority and involvement, etc. [2.1.3]. Projects are in Java ™.

Approaches and Tools:

Static code Analyzer [2.3], which, in this paper, is SonarQube, used to get TD from the original data set [2.3].

Grouping and computation of data, including decreased TD and increased TD, Induced TD, Frequency, Developer Seniority, Interval [3.2].

Gini Index and **Spearman's correlation**. Gini index is used to solve RQ1 while 2, 3, 4 is using Spearman's correlation.

Results:

1. Concerning RQ1, the answer is no. Some are more like uniformly but others, not. TD is **not uniformly changed** by all developers in OSS.
2. TD is negatively related to **commit frequency.**
3. TD is negatively related to **seniority of developers**.
4. TD is positively related to commit interval.

Mitigation to Threats to Validity:

1. Use SCA to calculate TD to upgrade accuracy.
2. Use unified set of rules in SCA to avoid bias and error.
3. Use widely-known and still-in-develop projects as data source to make sure it is typical as OSS.

My ideas

1. First, concerning correlational analysis, in this paper mainly uses Spearman's to induct the result of positive or negative. However, Spearman's is only for linear correlational analysis. Maybe this is enough to get what this paper wants but going further by using other correlational analysis may reveal deeper relationship which needs regression analysis. For example, to what level the manager should control the quality of commits which may also be applied to financial prediction. If the relation is not linear, the result in the paper may have limited value. Besides, I am also curious about what may be inducted if using other correlation coefficient, such as Pearson's.
2. What if using $\frac{commit\_date(c_j) - First\_commit\_date(d_i)}{number\_of\_commits(d_i)}$ as the formula for seniority? A developer/commit may have the first date one year earlier than the second but still get a great seniority value with current formula.
3. This may be a little out of topic, but what do you think of Coding AI, Pooyan? While a full-functioned coding AI may be impossible for now, it may still be a way to solve the problem of TD. A partially-functioned coding AI, working like "detect a functional part like a SCA tool and replace it with current 'good' code." This may need not only mining from repositories but also from code itself. Anyway, just take this as a flashing idea.
4. As "main module" is referred in the selection of systems, but whether commits analyzed is all from that module or not is not mentioned. Is the analysis in this paper the same as the one in QRS2017, carefully check architecture of each system? If not, this may also be a point to go deeper.
5. As mentioned in 2.3, the rules in computing TD is defined by SonarQube, but technical debts are not limited to this list. For example, documentation is part of TD and in QRS2017 paper you gave me, documentation commits are detected. If an further analysis is applied on those aspects, more characteristics may be revealed.
6. Dynamic code analysis, use others coefficient to solve RQ1.