LINGI2252 – Instructions for second report – Automated Code Analysis

After your "first contact" with the code, which was essentially based on a manual code assessment, for this second report you can use a variety of tools to perform a more automated assessment, and to dive deeper in your discovery of the system. Evidently, you can take into account all remarks on your first report to do a more in-depth study. Also, don't worry if you would come across some issues that seem to contradict what you wrote in your first report. It is normal that as you acquire a better vision on the overall quality of the analyzed system, you may want to revise some earlier conclusions.

First of all, for this second report, you should apply some metrics to analyze certain quality attributes of the code analyzed. You should focus on quality attributes like readability, reusability, adaptability, comprehensibly, and of course maintainability in particular. Choose your metrics carefully, apply them, present their results, and carefully analyze the obtained results to assess where and how the quality of the code could be improved. You may also suggest some further restructuring based on your analysis.

Here are some questions and tips that can be used as a source of inspiration to guide you when performing such **metrics** for this report:

- What metrics can you use for finding bad smells in the code?
  (Choosing for instance from Moose's metrics set, or from the small metrics framework you made in the exercise sessions, or maybe some metrics you implemented yourself.)
- What would be appropriate thresholds for those metrics and why?
- Using these metrics, can you find examples of good/bad design heuristics
  (such as coupling, cohesion, etc.)
- What quality aspects are particularly good or bad in the code? Examples: Efficiency, Completeness, Structuredness, Conciseness, Portability, Legibility, Reusability, …
- What would be the impact on quality of the code improvements you proposed in the first report?
  (for example: better reusability at the cost of efficiency.)
- Is the aforementioned impact measurable, for instance using the metrics you proposed above?
- Would you suggest further code restructurings based on your analysis of metrics? Which ones?

After having performed these metrics and identified further potential code refactorings to improve the code quality, you are asked to further verify or complete your findings by using advanced code inspection, querying, visualisation or other tools like Moose, Mondrian and Soul. (Feel free to use any additional tools as well.) Describe carefully how you applied and analyzed the results of these visualization and code querying tools to further assess the quality of the system you investigated. You need to describe:

- How you used and analyzed the results of applying a **metrics visualization** tool.
  - A tool like MOOSE or Mondrian to "see" the metrics.
  - (How) did the tool help you to find "surprising" things in the code?
  - How did you use manual code inspection (or code querying) to explain or confirm what you discovered?
  - What can you say about the code quality based on this analysis?
- How you used and analysed the results of applying a **code query** tool.
  - How you used code querying (SOUL) to analyze the code in detail.
  - Maybe you wrote some dedicated metric queries in SOUL?
  - How you could assess the code quality based upon this analysis?

For all the tools you used, take care to include a critical discussion of the quality of the tools you used. Although you are encouraged to use tools, beware that tools (metrics, visualisations, queries) may provide wrong results, either because the tool you used is still a prototype, or because you used the tool in a wrong way, or simply because you misinterpreted the results produced by the tool. So, before drawing any final conclusions from the results obtained, please verify carefully that the results and their interpretation make sense (do the values make sense, does the picture look weird?). Manually verify your findings against the code in case of doubt.

For this second report, remember to take into account the detailed feedback we gave on the previous report, and in particular make sure to:

- **Put the findings of your current (second) report in the context of the previous report** (are the results you found now confirming or contradicting what you said previously?);
- Include sufficient concrete examples (to **illustrate** and justify your observations and conclusions);
- Use pictures wisely (illustrations serve a **purpose**, don't use them as page-fillers);
- **Justify** why and how you come to certain conclusions (don't just say that something is good or bad, say **why** that is the case);
- Take into account the **particularities** of the case you are analysing (most good practices and design heuristics are only rules of thumb; they may not be perfectly respected in the case you are analysing, try to understand why);
- Always **verify** the results of a tool or metric against the code so that you really **understand** what is going on;
- Produce a report of good quality (well-structured, correct English, nice layout).

We expect the report to count maximum 8 pages of running text and be no longer than 12 pages in total, including pictures. It should be formatted in A4 with a normal margin, single spacing between lines and using 11pt Times and should be submitted to iCampus, in PDF format only, before the deadline.

Good luck.

Kim Mens and Sergio Castro