Research Objectives

Goals in sequential order

Main Aim (narrow scope) = Refactoring Process

1. Detection of code smells to determine flaws in code.
   - Report on internal problems detected
2. Address smells with refactoring und thus improve internal state
   - Report on the success through measurment

Secondary Aim: Discussion part

3. Find appropriate refactoring measures, insights and suggestions of smart factories
4. Detect potential threats to Validity
5. Report on unsuspected Results

Note: Aim is not to reworking the entire Application

Reworking the entire application is not in the scope of this paper. Therefore it will be determined which internal problems (smells) will be refactored beforehand.

Hypotheses

1. Improving Internal State

Statement

Refactoring leads to an improvement in the quality of the code.

Motivation

- Refactoring is known to increase the quality attributes of a project
- Refactoring is my means of choice in increasing the code quality
- I want to show a link between refactoring steps and the improvement of internal state

Variables

- Independent Variable: Structure of the code
- Dependent Variable: Quality Metrics

Testing the Hypothesis

- Measuring the code quality with predefined metrics
- Check whether refactoring the smells makes the metrics better and thus improves quality
- Check the strength of my hypothese according to the amount of increase

Shortcomings

- Sample size of n = 1
- However, the strength of improvement is of relevance
- Also, it is the insights that are generated by testing this hypotheses that add value to this paper (e.g. suggestions for CPS specific systems)

2. Alteration in External Behavior

Statement

The degree of fulfilled test requirements changes visibly based on which code smell is being refactored.

Motivation

- Changes in external behavior must be avoided during Refactoring in order to reduce bugs from being introduced
- Knowing which Code Smells show the most risks in introducing bugs in our cyber physical system is valuable.
- This is especially true when not having a comprehensive test suite.

Variables

- Independent Variable: Code Smell (e.g. Code duplication)
- Dependent Variable: How many Test Requirement are passed & Extent of Debugging

Testing the Hypothesis

- Testing whether requirements are fulfilled after each Code Smell that is refactored
- Start running tests early on in the first refactoring steps to avoid needing to revert to previous state (due to too much alteration in external behavior)

Shortcomings

- Again sample size of n = 1
- One Code smell might introduce no bugs in our project, but does the opposite in another
- The indications can then be examined in future works

3. Smart Factory Suggestions

NOTE: This hypothesis i am at least happy about, as I find it difficult to falsify.

Statement

Refactoring cyper physical systems differs from refactoring other software systems.

Area of Concentration

Relevance of certain quality attribute, prone to a code smell, importance of refactoring itself

Motivation

- In our paper, we want to find peculiarities of cps systems
- In the discussion section of the paper compare my insights with other sources (i.e. “Do you smell it too” paper)
- My paper as a contribution to find similarities too see where cps systems differ from other software projects
- These findings can be further proven / disproven in future works