1. Introduction
    1. Describes the problem statement, illustrates why this is a problem and describes the contribution the thesis makes in solving this problem. Optionally, it can give a short description (1-3 sentences each) of the remaining chapters. Good introductions are concise, typically no longer than 4 pages.
        1. Description of code categories
            1. Manually-maintained
                1. Production
                2. Test
            2. Generated
                1. Production
                2. Test
        2. Why is generated code a problem for static code analysis
            1. Irrelevant in terms of performance/security
        3. How does excluding generated code improve the analysis
        4. Why is an automatic approach to detect generated code releveant
    2. The introduction reveals the full (but summarized) results of your work. This appears counter-intuitive: does this not break the tension, like revealing the name of the murderer on the first page of a thriller? Yes, it does. That is the whole point. A thesis, and thus its architecture, aims primarily to inform, not entertain.
        1. Found many generator patterns
        2. By manually looking at including classes very good detection results
            1. Generated/Maintained code ratio
            2. Detected/Not detected ratio
        3. Creation of a Generator-Pattern Repository
2. Terms and Definitions
    1. *Terms and Definitions*: Defines the fundamental concepts your thesis builds on. Your thesis implements a new type of parser generator and uses the term *non-terminal symbol* a lot? Here is where you define what you mean by it. The key to this chapter is to keep it very, very short. Whenever you can, don’t reinvent a description for an established concept, but reference a text book or paper instead.
        1. Generator String

      2. Suffix Tree
      3. Clone
      4. CloneChunk
      5. CloneClass
      6. CloneResult
   2. If a reader cares enough about your topic to read your thesis, he probably knows the terms anyway. The chapter thus mostly serves to disambiguate overloaded terms and as a collection of pointers to more detailed sources for novice readers.
3. Related Work
   1. *Related Work*: Collects descriptions of existing work that is related to your work. Related, in this sense, means *aims to solve the same problem* or *uses the same approach* to solve a different problem. This chapter typically reads like a structured list. Each list item summarizes a piece of work (typically a research paper) briefly and explains the relation to your work. This last part is absolutely crucial: the reader should not have to figure out the relation himself. Is your piece better from some perspective? More generalizable? More performant? Simpler? It is ok if it is not, but I want you to tell me.
      1. E. Ukkonen – Suffix Tree Clone Detection
      2. Jonathan Bernwieser – Automatic Categorization of Source Code in Open-Source Software
4. Approach
   1. *Approach*: Outlines the main thing your thesis does. Your thesis describes a novel algorith for *X*? Your main contribution is a case study that replicates *Y*? Describe it here.
      1. Find generator strings by detecting clones in comments
      2. Use Teamscale as scanner to get the AST & Tokens
      3. Filter the tokens to extract the comments
      4. Normalize comments for use in suffix tree clone detection
      5. Build suffix tree
      6. Find clones
      7. Filter possibly generated clone results
      8. Generate links to the files containing the generator strings
      9. Generation of a Generator-Pattern Repository
         1. Pattern

         2. Generator
         3. Language
         4. Relevanz
5. Evaluation
   1. Evaluation: Describes why your approach really solves the problem it claims to solve. You implemented a novel algorithm for X? This chapter describes how you ran it on a dataset and reports the results you measured. You replicated a study? This chapter gives the results and your interpretations.
   2. The Appraoch and Evaluation chapters contain the meat of your thesis. Often, they make up half or more of the pages of the entire document.
      1. Research Questions
         1. RQ1: Evaluation of the completeness and accuracy of the produced heuristics against a reference data set
         2. RQ2: Automatic classification of source code in a huge collection of open source systems & evaluation of the ratio of generated code
      2. Study Objects
         1. RQ1: Qualitas Corpus
            1. Characteristics
            2. Why used
         2. RQ2: Git crawler projects
            1. Characteristics
            2. Why used
      3. Study design
         1. G: Set of generated code
            1. DG: Set of detected generated code
            2. NG: Set of not detected generated code
         2. M: Set of manually maintained code
            1. DM: Set of detected manually maintained code
            2. NG: Set of not detected manually maintained code
         3. |G|/|M| = Ratio of generated to manually maintained code
         4. |NG|/|DG| = Ratio of not detected generated code to detected generated code
         5. |NG ∩ DM| = Ratio of not detected generated code to detected manually

    maintained code

4. Procedure
    1. Use Teamscale as scanner to get the AST & Tokens
        1. How the AST looks like
        2. How the tokens look like
    2. Filter the tokens to extract the comments
        1. Show unfiltered
        2. Show filtered
    3. Normalize comments for use in suffix tree clone detection
        1. Comment to CloneChunk conversion
        2. Characteristics of a CloneChunk
        3. Sentinels
    4. Build suffix tree
    5. Find clones
        1. Conversion of clone classes to clone results
    6. Filter possibly generated clone results
        1. Filter pattern
    7. Generate links to the files containing the generator strings
        1. Get filesystem path from teamscale server
    8. Generation of a Generator-Pattern Repository
        1. Pattern
        2. Generator
        3. Language
        4. Relevanz
5. Results
    1. RQ1:
        1. G: = ?
            1. DG: = ?
            2. NG: = ?
        2. M: = ?
            1. DM: = ?

                    2. NG: = ?
                3. |G|/|M| = ?
                4. |NG|/|DG| = ?
                5. |NG ∩ DM| = ?
            2. RQ2:
                1. Test on a huge set of open-source projects
                2. Ratio of code categories etc.
        6. Discussion
            1. Completeness and accuracy
            2. Relevance of each result
        7. Threats to Validity
            1. Wrong filtering and thus loosing generator patterns
            2. Draw-off between minimum clone length and unnecessary generated data
6. Future Work
    1. *Future Work*: In science folklore, the merit of a research question is compounded by the number of interesting follow-up research questions it raises. So to show the merit of the problem you worked on, you list these questions here. If you don't care about research folklore (I did not as a student), this chapter is still useful: whenever you stumble across something that you should do if you had unlimited time, but cannot do since you don't, you describe it here. Typical candidates are *evaluation on more study objects*, investigation of potential *threats to validity*, … The point here is to inform the reader (and your supervisor) that you were aware of these limitatons. Limit this chapter to very few pages. Two is entirely fine, even for a Master's thesis.
        1. Integration in Teamscale
        2. Evaluation on more study objects to find more generator patterns
        3. Test of the system on different clone lengths
7. Conclusion
    1. *Conclusion*: Short summary of the contribution and its implications. The goal is to drive home the result of your thesis. Do not repeat all the stuff you have written in other parts of the thesis in detail. Again, limit this chapter to very few pages. The shorter, the easier it is to keep consistent with the parts it summarizes.