The Academic Plan

- **Methodology ✢**[7 DAYS]
  - **Step 1: demonstrate what Jigsaw does and how to use it, to the extent that is pertinent for this study**
    1. Select 10 Java methods that use logging calls, as a test set ✓
    2. Map source code of logged Java methods in the test set to AST structure via the Eclipse JDT framework ✓
    3. Use Jigsaw to determine the correspondences between the ASTs in a pairwise manner comparison ✓
    4. Check that the Jigsaw similarity measure makes intuitive sense ✓
    5. Visualize the comparison results using Gephi (a graphing tool) ✓
  - **Step 2: construct an anti-unifier (structural generalization) from two given logged Java methods with a special attention to logging calls ✓**
    1. Developing an Algorithm that:
       * maps the source code of two logged Java methods to the AST structure via the Eclipse JDT framework ✓
       * creates an extension of AST structures, called AUAST, to allow the application of higher-order anti-unification modulo theories ✓
       * determines potential candidate structural correspondences between AUAST nodes using the Jigsaw framework ✓
       * applies some constraints to prevent the anti-unification of logging calls with anything else ✓
       * develops a greedy selection algorithm to approximate the best anti-unifier to our problem by determining the best structural correspondences for each node
       * develops a measure of similarity between two AUASTs ✓
       * constructs an anti-unifier ✓
    2. Implement our approach as an Eclipse plug-in, building atop Jigsaw ✓
    3. Test our approach and tool on logged Java methods of the test set ✓
    4. Measure the test coverage ✢[1 DAYS]
  - **Step 3: anti-unify a set of AUASTs of logged Java methods**
    1. Develop a modified version of a hierarchical agglomerative clustering algorithm suited to our application ✓
    2. Implement our approach as an Eclipse plug-in, building atop the Step 2.2 extension of Jigsaw ✓
    3. Test our approach and tool on logged Java methods of the test set ✓
  - **Step 4: conduct an empirical study to characterize where logging calls are used in the source code ✢**[7 DAYS]
    * Select three open-source software systems that make use of logging ✓

    * Extract all logged Java methods from the source code of these systems and anti-unify them to determine the patterns on a per-system method-granularity basis via the tool developed in Step 3.2 ✛[5 DAYS]

- **Writing the thesis** ✛[21 DAYS]
  - Writing chapters of thesis, including:
    * CHAPTER 1: Introduces the problem and why it matters, outlines the idea of our proposed solution, and some key related work to argue that the problem has not been solved and the idea of solution is novel ✛[3 DAYS]
    * CHAPTER 2: motivates the problem of understanding where to use logging calls in the source code through an example ✛[2 DAYS]
    * CHAPTER 3: provides the background information ✓
    * CHAPTER 4: describes our proposed approach and its implementation as an Eclipse plug-in ✓
    * CHAPTER 5: presents an empirical study conducted to evaluate our approach and its application to characterize logging usage ✛[4 DAYS]
    * CHAPTER 6: discusses the results and findings of my work, threats to its validity, and the remaining issues. ✛[3 DAYS]
    * CHAPTER 7: describes related work to our research problem and how it does not adequately address the problem ✓
    * CHAPTER 8: concludes the dissertation, presents the contributions of our study, and future work ✛[2 DAYS]
    * Appendix ✛[2 DAYS]
    * Corrections and Revisions ✛[5 DAYS]