TDA 594/DIT 593 - Assignment 2 - Domain Engineering and Analysis

Due Date: Sunday, November 21, 23:59
Submission: Via Canvas (one ZIP file per team)

Overview

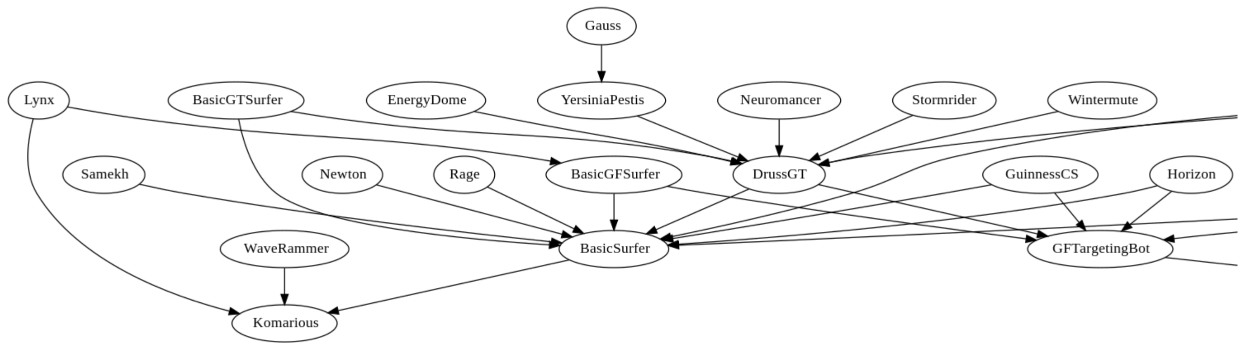
You are leading the product management team of a company that develops the control software for mobile service robots. The company has been developing this kind of software for a long time, for many different customers. It recently also acquired smaller startups that have made these robots even more intelligent, relying on various AI techniques (e.g., reinforcement learning or search-based identification of optimal strategies), since some of their customers participated in RoboCup competitions.

You have recently worked on developing more configurable robots, offering your customers easier ways to customize them. You also experimented with engineering domain-specific languages that your customers can use to specify the behavior of your robots. All these developments have led to the situation that your company has a large number of control software projects with many commonalities, but which differ in various respects. Those that your company developed have substantial overlap in code; those from the acquired startups have an overlap at the feature level, but the code looks very different. Unfortunately, the maintenance effort has grown out of hand.

Through interacting with the customers of your company, you also notice an increasing demand for customization and quickly tailoring the robot to their needs. Customizing a robot from your large portfolio of robots is rather difficult for you, and customers have difficulty setting up the robots. You see potential in developing a product line for this market.

You have decided to systematically explore your large portfolio of robot control systems, identifying the existing features, then scoping them, and later integrating those features into a platform. ***In short, you have decided to create a product line representing mobile robots***. This should significantly reduce maintenance effort and the time-to-market of new robot variants, while also allowing your customers to tailor the robots to their needs.

For this course, we don’t have the portfolio of commercial robot-control software, but we have RoboCode, whose many open-source bots nicely reflect the situation in this imaginary company. There is diversity in the documentation of the bots, ranging from very detailed to high-level and scattered. Some bots have been derived from others, leading to code overlap, while others overlap only at the feature level, having significantly different codebases. The figure below shows some core robots from which many robots have been derived (the arrow points to the source of derivation). ***Therefore, your task in this assignment is to build a feature model representing mobile robots, based on the existing open-source RoboCode robots.***



Your Tasks

1. Analyze this domain by reading through descriptions of open-source robots.
    a. The primary source of information is the RoboWiki: http://robowiki.net/wiki/Open_Source_Bots
    b. You can find the individual bot code repositories and look for additional information on the Internet.
2. Identify features that are mandatory and that are optional for a robot product line. You can identify features that pertain to functional aspects (e.g., how the robot moves or a specific strategy to combine movement and targeting) and non-functionalities aspects (e.g., accuracy, speed). Also keep in mind that features may not directly correspond to externally-visible functionality, such as debugging or logging strategies. In addition to the lectures and resources listed on Canvas, the following papers may offer additional hints:
    a. D. Nesic, J. Krueger, S. Stanciulescu, T. Berger, "Principles of Feature Modeling," http://www.cse.chalmers.se/~bergert/paper/2019-fse-fm_principles.pdf
    b. T. Berger, D. Lettner, J. Rubin, P. Grünbacher, A. Silva, M. Becker, M. Chechik, K. Czarnecki, "What is a Feature? A Qualitative Study of Features in Industrial Software Product Lines," https://gsd.uwaterloo.ca/sites/default/files/berger-2015-featurestudy.pdf
    c. This part can be nicely parallelized among the team members.
        i. For selecting the variants, different strategies are possible, you could follow the derivation graph from the core robots, select bots from the respective RoboWiki page randomly, or combine both strategies. Try to look at a diverse group of robots to capture the widest array of features and variation points[1].
3. Organize the features in a feature model. If you identified dependencies from the descriptions, then also model them.
4. Translate your feature model into a propositional formula. List all of the original bots you examined together with their features (so, each bot is now a valid configuration of the feature model). Document two features in more detail.
    a. Some items you may wish to consider in describing a feature include:
        i. Feature Name

[1] The full version of the picture above can be found at https://chalmers.instructure.com/courses/16077/files/folder/Assignments?preview=1757333

        ii. Description of the feature and its corresponding requirements
        iii. Relationship to other features, especially hierarchy, order, and grouping
        iv. Potential interactions with other features
        v. External dependencies, such as required hardware resources (if any)
        vi. Interested stakeholders or customers
        vii. Configuration knowledge, such as "activated by default"
        viii. Configuration questions asked during the requirements analysis step
        ix. Constraints, such as "requires feature X and excludes feature Y"
        x. Behavioral specifications, including invariants and pre- and post conditions
        xi. Known effects on non-functional properties, such as "improves performance and increases energy consumption"
        xii. Rationale for including a feature in the scope of the product line
        xiii. Additional attributes, such as numbers and textual parameters, used for further customization during product generation
    b. You do not need to include all of these fields, but should try to use these items to structure your description. Do not make up information that you do not have.
5. Translate your propositional formula into the DIMACS format discussed in class. Use a SAT solver (see hints below) to generate one valid configuration to ensure that your propositional formula, CNF conversion, and DIMACs translation are correct.
    a. If the generated configuration matches one of the bots you analyzed, note which one. If not, note that as well.
    b. If you are unable to generate a valid configuration, identify and fix issues in your formula or CNF translation.

Hints

There is no specific requirement on completeness, and there is also no single correct answer. Try to be thorough and detailed. Ideally, your feature model would capture all current RoboCode bots accurately. However, we do not expect you to read the descriptions of all 200+ bots and form a perfect model. In general, your model should be detailed, logical, and apply reasonable constraints. Specifically, you should aim for a model with at least 25 features.

We recommend trying to capture a relevant sample of the bots and forming a model from that sample, then picking a small second set of bots to validate the model and ensure that it reflects those bots as well. To help get you started, we have provided the code for three bots (BasicSurfer, BasicGFSurfer, and GFTargetingBot) on Canvas:
https://chalmers.instructure.com/courses/16077/files/folder/Assignments?preview=1757335

Examine the code of these bots and look at their description on the RoboCode wiki. We recommend that you, as a group, examine these bots to form the start of your feature model before attempting to split the work of examining additional bots. This will allow you to form a common starting point and beginning of a feature model that can be expanded.

From this point you can split up the work in any way you choose, but we recommend one or both of the following:

- Each group member looks at around five additional bots and develops their own feature model, building on the common model formed from the first three bots. The group then works together to merge the individual models into a final combined model (merging overlapping features).
- Each group member could take responsibility for a subset of the high-level features from the initial model. This group member can then examine additional models and refine that feature and further options that emerge when that feature is selected, and suggest additional features that were not in the original model. In this case, each member should look at a large number of bots, but is primarily focused on refinements and expansion of the chosen features.

To create the feature model, **you are required to use the tool FeatureIDE** (which integrates into Eclipse). FeatureIDE is available from https://featureide.github.io/. This page also includes installation instructions and tutorials. **We recommend the use of FeatureIDE 3.7**. You must use Java 8 or newer (we have tested with Java 8 and 15). We have tested with Eclipse 09-2020 and 12-2018. Therefore, we recommend the combination: (FeatureIDE 3.7, Java 15, Eclipse 09-2020). If you encounter issues, report them to your supervisor.

When organizing features in the feature model, you might recognize overlaps in the features from different team members. For instance, you might have two different feature names, but when discussing them in the team, you notice that these refer to the same functionality or are only slightly different. In the former case, you should choose the more intuitive name; in the latter case, you also introduce a sub-feature capturing the difference.

Be careful when designing the constraints. Elements such as XOR result in very large propositional formulae when converted to conjunctive normal form. Include constraints if they are necessary and make intuitive sense. However, make sure they are actually needed to create valid products.

You are free to use any SAT solver for this assignment. However, we recommend using the well-known C solver PicoSAT (http://fmv.jku.at/picosat/), which has bindings for languages like Python (https://github.com/ContinuumIO/pycosat) and Golang (https://github.com/wkschwartz/pigosat). Note which tool you used in your deliverable, if you did not use PicoSAT. Specifically, we recommend the use of the Python binding linked previously, as it is relatively simple to invoke.

Deliverable

Submit one ZIP file to Canvas with:

- Document (in PDF form) reporting about the results:
    - Document and justify the strategy that you followed for selecting bots to examine variants and also the strategy for identifying features.

    - Provide the feature model (include a high-resolution image), briefly explain the hierarchy of the model and the cross-tree constraints that you derived.
    - Provide the propositional formula and the mapping of features to integer IDs used by the DIMACS format
    - Provide the list of original bots, mapped to configurations of your feature model
    - Provide the configuration generated by the SAT solver, and the mapping of that configuration to a bot you have examined (if such a mapping exists).
- The source file (or full-resolution image) of the feature model you created.
- The DIMACs file you created from your propositional formula.

Grading Guidelines

Note, these guidelines are intended to give some guidance, but are not exhaustive. Each supervisor will assign a grade based on the correctness and quality of your work.

Grade	Guidelines
5	• Thorough and well-documented feature model, documenting at least 25 features. • Detailed explanations of the identified features. • Consistent and logical use of constraints (model constructs and cross-tree constraints). • Explanations of each constraint present and detailed. • Detailed justification for design decisions made. • Detailed mapping of bots to your model. • Explanation of any model changes made after using the SAT solver to identify a configuration. • Well-formatted feature model, propositional formula, and DIMACS file.
4	• Thorough and reasonably documented feature model, documenting at least 25 features. • Reasonably consistent and logical use of constraints (model constructs and cross-tree constraints). Some attempt at explaining the constraints. • Explanations of the identified features present. • Justification present for design decisions made. • Detailed mapping of bots to your model. • Explanation of any model changes made after using the SAT solver to identify a configuration. • Well-formatted feature model, propositional formula, and DIMACS file.
3	• Reasonably thorough feature model with basic understandable documentation, documenting at least 20 features. • Constraints (model constructs and cross-tree constraints) present and reasonably logical. Some attempt at explaining these constraints. • Justification present for design decisions made. • Some mapping of bots to your model. • Well-formatted feature model, propositional formula, and DIMACS file.

U	● No, or limited, documentation for your feature model. Too few features identified. ● Constraints (model constructs and cross-tree constraints) either not present or not reasonable. ● No justification present for design decisions. ● No or limited mapping of bots to your model. ● Missing or significant formatting issues in feature model, propositional formula, and DIMACS file.