Project

During the evolution of a system parts of the source code often become isolated from the rest of the system and are not referenced anymore. This can be either the result of refactoring, or simply certain functionality is not needed anymore. If the code is not removed from the system, this code is called *dead code*. Having dead code in the system only makes the maintenance harder since the system seems to be larger than it actually is. As a result dead code should be removed.

The goal of the project is to give you a taste of what it means to build a software analysis tool. Your project will be an analysis tool which will detect dead code in a subject software system. You will have to use and combine as efficiently as possible all the techniques that you will learn during the course: static analysis, dynamic analysis, multi-version analysis, and ecosystem analysis to implement a tool that performs dead code detection.

Evaluation criteria and grading

The project will be graded with a maximum of 6 points. The points will be distributed across the following two criteria:

- **Accuracy** of the detector (4p) which is measured as the number of lines of code that are correctly detected as containing dead code minus the number of lines that are incorrectly detected as containing dead code. Number of lines means number of non-comment lines of code.
- **Architecture** of the implementation (2p). Solution which combine the different types of analysis in inventive ways will receive extra points. Only groups which integrate at least three of the lecture analyses will receive the full grade.

Unless exceptional circumstances arise, all members of a team will receive the same grade.

Timeline

The duration of the project is 4 weeks. During each week you will have to integrate in your project a new subsystem which takes into account a new type of information that you have learned about during the lecture in that week. The lectures and corresponding submodules that your system should integrate are:

1. Static Analyzer
2. Multi-Version Analyzer
3. Dynamic Analyzer
4. Ecosystem Analyzer

During the four weeks of the project you will work on a case study that you choose or we propose. At the end of the project we will give you a project that you have not seen before and you will have to analyze it and present the results.

The Software Evolution Cup

The team that provides the tool with the best accuracy will be declared the winner of the Software Evolution Cup. For the cup, the architecture does not matter but like in the real world, the only thing that matters is the result.

Publishing your Project

Each team has to create a project on SqueakSource. Send us an email with the name of the project as soon as you have created it!

API

In your project you must have a class that is called *DeadCodeDetector* which has two methods that we are interested in:

- *getTopTenSuspects* returns a list with the fully qualified names of the top ten suspects ranked by the number of lines of code they cover.
- *getAllSuspects* the second returns a list of the fully qualified names for all the artifacts that are potentially dead code.

The following code snippet presents the way the API of your class will progress through the four weeks:

```
| detector |

"after the first week"
detector := DeadCodeDetector new.
detector loadModelFromFile: 'Framv4.mse' date: (Date today).

"after the second week..."
detector addVersionModelFromFile: 'Framv3.mse' date: (Date fromString: '01−01−2009');
    addVersionModelFromFile: 'Framv2.mse' date: (Date fromString: '01−01−2008');

"after the third week..."
detector runTestsFromPackage: 'FramPack'. "for Smalltalk people"
detector loadDynamicInfoFromFile: ''. "for Java people"

"after the fourth week..."
detector addExternalSystemModelFromFile: 'Client1.mse'.
detector addExternalSystemModelFromFile: 'Client2.mse'.

"getting the dead code"
detector getTopTenSuspects.
```

Each of the subsystems will take into account different sources of information:

- Static Analysis: a FAMIX model presented to you as an MSE file.
- Multi-Version Analyzer: multiple FAMIX versions of the system as MSE files
- Dynamic Analyzer: the source code of the system together with a collection of unit tests
- Ecosystem Analyzer: FAMIX models of several other projects that are part of the ecosystem, some of which might be using your project

Final Report

The members of the group must provide a brief description (not longer than a page) in which they enumerate:

- Design decisions.
- Enhancements introduced in the tool.
- Limitations.