Radboud University

Dr. Daniel Strüber
Dr. Carlos Diego Damasceno

Radboud University
Institute for Computing
and Information Sciences

Autumn term 2022/23
Published: 26.09.2022
To be submitted:
03.10.2022 13:30

Software Product Lines

Assignment 4

Task 1: Code tangling and scattering

a) What is code tangling? Does the following example contain it?
b) What is code scattering? Does the following example contain it?
c) What are the possible consequences of code tangling and scattering?
d) Where could code tangling and scattering appear in a chat application?

```
class Graph {
  List<Node> nodes = new ArrayList<Node>();
  List<Edge> edges = new ArrayList<Edge>();
  boolean locked = false;
  void addEdge(Node source, Node target, boolean directed) {
    if (locked) return;
    locked = true;
    Edge edge = new Edge(source, target);
    edges.add(edge);
    Logger.log("Edge added: " + edge);
    locked = false;
    if (!directed)
      addEdge(target, source, true);
  }
  void addNode(String name, Color color) {
    if (locked) return;
    locked = true;
    Node node = new Node(name);
    node.setColor(color);
    nodes.add(node);
    Logger.log("Node added: " + node);
    locked = false;
  }
}
```

Task 2: Usage scenarios

At this point, we have considered the following variability mechanisms: runtime parameters, design patterns, version control and build systems, components, services, white- and black-box frameworks.

For each of the following scenarios, discuss which variability mechanisms are well-suited to support them, and which not suited at all. Discuss why.

a) The most important requirement is runtime performance
b) The system will be developed with many programmers
c) There are many potential customers that all have different requirements
d) There are plans to buy functionality from a third-party vendor
e) There are many fine-grained extensions

Task 3: Black-box framework implementation

a) Evolve your chat product line towards a black-box framework implementation. Implement each of the features considered so far (colors, authentication, 2x encryption, logging, 2 UIs) as plug-ins.*

   Manually test some products of the resulting product line. To this end, implement a class with a main method that launches your chat system with a selection of plug-ins. You can hard-code the plug-in selection into the main method; it's not required to implement a generic plug-in loader mechanism.

   Hint: Create at least one interface that provides all required hotspots and can be extended by the plug-ins. It might make sense to actually create several interfaces, which allows to implement different plug-in types.

   * *Modified version for smaller groups:*
   3-person groups may omit support for logging
   2-person groups may omit support for logging and authentication

b) Critically reflect on what you did in subtask a):
   - How much effort was required to create the plug-ins?
   - How often did you have to change the framework source-code or plug-in interfaces because the existing solution did not fit?

Hand in a zip file with:

- a) your source code, including the class for launching and testing your system
- b) a brief textual report (ca. 2 paragraphs, PDF or txt)