SAARLAND UNIVERSITY
COMPUTER SCIENCE

Prof. Dr. Sven Apel
Yannick Lehmen
Maurice Vincon

# Software Engineering WS 2023

# Exam Preparation Sheet (Suggested Solutions)

***Note:*** **This task sheet was created by tutors. The tasks are neither relevant nor irrelevant for the exam, the suggested solutions are neither correct nor incorrect.**

## 1 Join Points

a) Mark all join points in the following code example.
b) List all join points from the following code example that are selected by the following pointcuts:

1. call(void Locker.lock())
2. call(* Locker.*(..))
3. get(* Foo.locker)
4. execution(void Foo.log(String))
5. execution(void Foo.*(String))
6. call(void *(*))
7. call(* *(..))

You can assume that any function for which the returned value is never stored has return type void.

```
public class Foo{
  private Locker locker = new Locker();

  void main(String param) {
        int lockID = locker.lock();
    try{
    int i = param.length()%2;
        firstOperation(parameter);
        log("First Operation executed with parameter " + param);
        secondOperation(parameter, i);
        log("Second Operation executed with parameters " + param +
         ", " + i);
    } finally{
        locker.unlock(lockID);
    }
  }
}
```

**Suggested solution:**

a)
- line 3: constructor call
- line 3: field access (set locker)
- line 5: method execution
- line 6: field access (get locker)
- line 6: method call
- line 8: method call
- line 9: method call
- line 10: method call
- line 11: method call
- line 12: method call
- line 14: field access (get locker)
- line 14: method call

b)
- -
- line 6: method call, line 13: method call
- line 6: field access (get locker), line 13: field access (get locker)
- -
- line 5: method execution
- line 9, 10, 12, 14: method call
- line 3: constructor call, line 6, 8, 9, 10, 11, 12, 14: method call

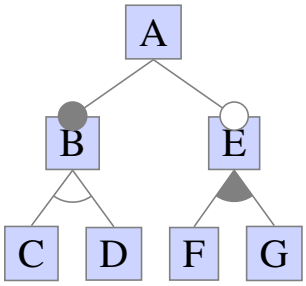
## 2 Coverage Criteria

For the given feature diagram:

a) give a set of configurations that achieves 2-wise feature coverage

b) give a set of configurations that achieves 2-wise interaction coverage

c) give a set of configurations that achieves 3-wise feature coverage




**Suggested solution:**

a) $\{A, B, C, E, F, G\}, \{A, B, D, E, F, G\}$

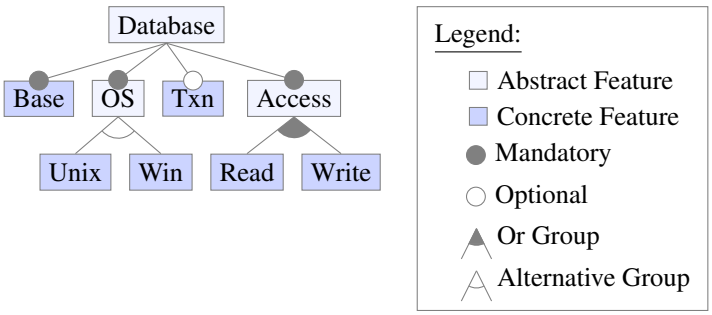
b) $\{A, B, C\}, \{A, B, D\}, \{A, B, C, E, F, G\}, \{A, B, D, E, G\}, \{A, B, D, E, F\}$

c) $\{A, B, C, E, F, G\}, \{A, B, D, E, F, G\}$

# 3 CART

| Configuration | Database | Base | OS | Access | Unix | Win | Txn | Read | Write | Runtime (in s) |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 3 |
| 2 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 6 |
| 3 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 6 |
| 4 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 11 |
| 5 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 11 |
| 6 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 11 |
| 7 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 15 |

1. Create the CART of the data in Table 1. Use the formulas from the lecture and provide the samples and their means for every branch. If two split options are equally good, we use the alphabetic ordering of feature names as tie breaker.
2. Predict the Runtime (in s) for the following configurations (given in set notation):
    1. { Database, Base, OS, Access, Unix, Write }
    2. { Database, Base, OS, Access, Win, Txn, Write }
3. For each of the following partial configurations find a valid configuration that is in line with the partial configuration and has the lowest predicted runtime:
    1. { (Win, 1), (Txn, 0) }
    2. { (Unix, 1), (Access, 1), (Txn, 1)}




**Suggested solution:**

1. possible split of X with 8 datapoints on: Unix

   left mean: 4.0

   right mean: 12.0

   left squared error loss: 18.0

   right squared error loss: 12.0

   sum: 30.0

   possible split of X with 8 datapoints on: Win

   **This splits the datapoints in the same way as Unix and will thus result in the same means and squared error losses (just with the left and the right branch swapped).**

   sum: 30.0

   possible split of X with 8 datapoints on: Txn

   left mean: 8.0

right mean: 8.0

**The squared error losses do not have to be computed since there is no knowledge gain for the prediction through this split at this point in time.**

left squared error loss: 78.0

right squared error loss: 80.0

sum: 158.0

possible split of X with 8 datapoints on: Read

left mean: $\frac{1+3+2*6+2*11+15}{7} = \frac{53}{7} = 7 + \frac{4}{7} = 7.\overline{571428}$

right mean: 11.0

left squared error loss: $(\frac{53}{7}-1)^2+(\frac{53}{7}-3)^2+(\frac{53}{7}-6)^2+2*(11-\frac{53}{7})^2+(15-\frac{53}{7})^2 = \frac{46^2+32^2+2*11^2+2*24^2+52^2}{49} = \frac{7238}{49} = \frac{1034}{7} = 147 + \frac{5}{7} = 147.\overline{714285}$

right squared error loss: 0.0

sum: $147.\overline{714285}$

possible split of X with 8 datapoints on: Write

left mean: 9.8

right mean: 5.0

left squared error loss: 58.8

right squared error loss: 56.0

sum: 114.8

**split by feature: Unix**

possible split of XL with 4 datapoints on: Txn

left mean: 4.5

right mean: 3.5

left squared error loss: 4.5

right squared error loss: 12.5

sum: 17.0

possible split of XL with 4 datapoints on: Write

left mean: 6.0

right mean: 2.0

left squared error loss: 0.0

right squared error loss: 2.0

sum: 2.0

**split by feature: Write**

**Only one** possible split of XLR with 2 datapoints on: Txn

**split by feature: Txn**

possible split of XR with 4 datapoints on: Txn

left mean: 15.0

right mean: 11.0

left squared error loss: 0.0

right squared error loss: 0.0

sum: 0.0

**We won't find a better splitter here, which means that only a splitter, which is as good as Txn and comes before Txn in the alphabetical order could still be picked instead.**

possible split of XR with 4 datapoints on: Read

left mean: $\frac{11+11+15}{3} = 12 + \frac{1}{3} = 12.\overline{3}$

right mean: 11.0

left squared error loss: $2 * (\frac{4}{3})^2 + (\frac{8}{3})^2 = \frac{2*16+64}{9} = \frac{96}{9} = \frac{32}{3} = 10 + \frac{2}{3} = 10.\overline{6}$

right squared error loss: 0.0

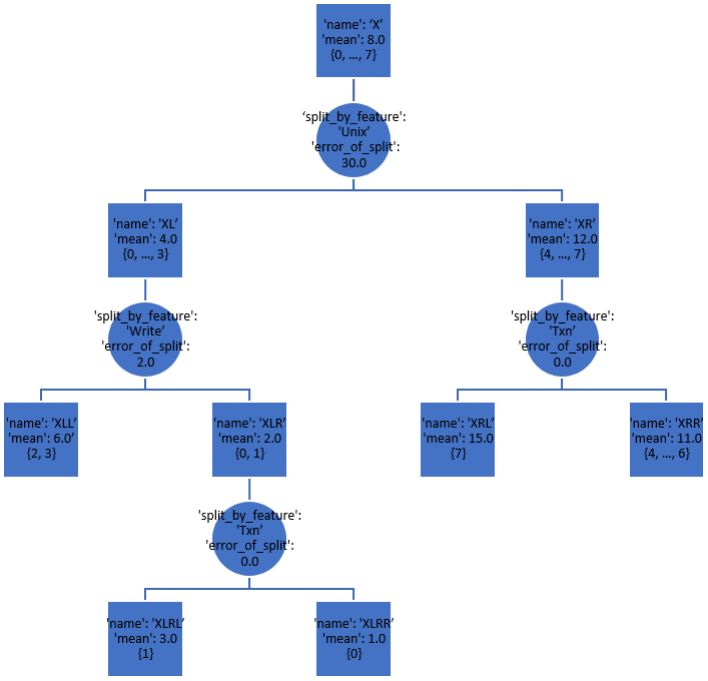
sum: $10.\overline{6}$

possible split of XR with 4 datapoints on: Write

**This splits the datapoints in a way such that the numbers for the computations are the same as for Read, because Configuration 5 and 6 have both the Runtime 11. Thus this results in the same means and squared error losses.**

sum: $10.\overline{6}$

**split by feature: Txn**




2. Predict the Runtime (in s) for the following configurations (given in set notation):
    1. {Database, Base, OS, Access, Unix, Write} $\rightarrow 6s$
    2. {Database, Base, OS, Access, Win, Txn, Write} $\rightarrow 15s$

3. For each of the following partial configurations find a valid configuration that is in line with the partial configuration and has the lowest predicted runtime:
    1. {Database, Base, OS, Win, Access, Read)} $\rightarrow 11s$
    2. {Database, Base, OS, Unix, Access, Read, Txn} $\rightarrow 3s$

# 4 Coarsening/Refinement

Prove the cases 8 and 14 from the *Coarsening and Refinement* slide in the lecture slides. Use the notation that is used to prove the cases 1-4 in the lecture.

$$\text{if } e_1 \text{ then choice } \langle \phi, e_2, e_3 \rangle \text{ else } e_4 \quad \equiv \quad \text{choice } \langle \phi, \text{ if } e_1 \text{ then } e_2 \text{ else } e_4, \text{if } e_1 \text{ then } e_3 \text{ else } e_4 \rangle \tag{8}$$

$$\text{choice } \langle \phi, \text{choice } \langle \neg\phi, e_2, e_1 \rangle, e_1 \rangle \quad \equiv \quad e_1 \tag{14}$$

**Suggested solution:**

**Proof:** We have to proof that for any $e, e' \in$ *VExpr* it holds that $e \equiv e'$ if and only if, for any given configuration $\psi \in \mathbb{B}_F$ with $e$ and $e'$ closed under $\psi$, $[\![e]\!]_\psi = [\![e']\!]_\psi$ holds. The proof is by structural induction over the terms $e, e' \in$ *VExpr*.

*Induction hypothesis*: If $e$ and $e'$ both contain no choice expressions, then $e \equiv e'$ if and only if $e = e'$ and therefore also $\forall\, \psi \in \mathbb{B}_F.\ [\![e]\!]_\psi = [\![e']\!]_\psi$, as none of the transformation rules are applicable.

*Induction step*: $e_1, e_2, e_3, e_4 \in$ *VExpr*, closed$(e_1, \psi)$, closed$(e_2, \psi)$, closed$(e_3, \psi)$, closed$(e_4, \psi)$

**Case 8:** $e = \text{if } e_1 \text{ then choice } \langle \phi, e_2, e_3 \rangle \text{ else } e_4 \quad e' = \text{choice } \langle \phi, \text{ if } e_1 \text{ then } e_2 \text{ else } e_4, \text{if } e_1 \text{ then } e_3 \text{ else } e_4 \rangle$

$$[\![e]\!]_\psi = [\![\text{if } e_1 \text{ then choice } \langle \phi, e_2, e_3 \rangle \text{ else } e_4]\!]_\psi = \text{if } [\![e_1]\!]_\psi \text{ then } [\![\text{choice } \langle \phi, e_2, e_3 \rangle]\!]_\psi \text{ else } [\![e_4]\!]_\psi$$
$$[\![e']\!]_\psi = [\![\text{choice } \langle \phi, \text{ if } e_1 \text{ then } e_2 \text{ else } e_4, \text{if } e_1 \text{ then } e_3 \text{ else } e_4 \rangle]\!]_\psi$$

if $\psi \Rightarrow \Phi$:

$$\begin{aligned} \text{if } [\![e_1]\!]_\psi \text{ then } [\![\text{choice } \langle \phi, e_2, e_3 \rangle]\!]_\psi \text{ else } [\![e_4]\!]_\psi &= \text{if } [\![e_1]\!]_\psi \text{ then } [\![e_2]\!]_\psi \text{ else } [\![e_4]\!]_\psi \\ &= [\![\text{if } e_1 \text{ then } e_2 \text{ else } e_4]\!]_\psi \\ &= [\![\text{choice } \langle \phi, \text{if } e_1 \text{ then } e_2 \text{ else } e_4, \text{if } e_1 \text{ then } e_3 \text{ else } e_4 \rangle]\!]_\psi \end{aligned}$$

if $\psi \Rightarrow \neg\Phi$ :

$$\begin{aligned} \text{if } [\![e_1]\!]_\psi \text{ then } [\![\text{choice } \langle \phi, e_2, e_3 \rangle]\!]_\psi \text{ else } [\![e_4]\!]_\psi &= \text{if } [\![e_1]\!]_\psi \text{ then } [\![e_3]\!]_\psi \text{ else } [\![e_4]\!]_\psi \\ &= [\![\text{if } e_1 \text{ then } e_3 \text{ else } e_4]\!]_\psi \\ &= [\![\text{choice } \langle \phi, \text{if } e_1 \text{ then } e_2 \text{ else } e_4, \text{if } e_1 \text{ then } e_3 \text{ else } e_4 \rangle]\!]_\psi \end{aligned}$$

$$\Rightarrow [\![e]\!]_\psi = [\![e']\!]_\psi \overset{\text{(I.H.)}}{\Longrightarrow} e \equiv e'$$

**Case 14:** $e = \text{choice } \langle \phi, \text{choice } \langle \neg\phi, e_2, e_1 \rangle, e_1 \rangle \quad e' = e_1$

if $\psi \Rightarrow \Phi$ :

$$[\![e]\!]_\psi = [\![\text{choice } \langle \phi, \text{choice } \langle \neg\phi, e_2, e_1 \rangle, e_1 \rangle]\!]_\psi = [\![\text{choice } \langle \neg\phi, e_2, e_1 \rangle]\!]_\psi = [\![e_1]\!]_\psi = [\![e']\!]_\psi$$

if $\psi \Rightarrow \neg\Phi$ :

$$[\![e]\!]_\psi = [\![\text{choice } \langle \phi, \text{choice } \langle \neg\phi, e_2, e_1 \rangle, e_1 \rangle]\!]_\psi = [\![e_1]\!]_\psi = [\![e']\!]_\psi$$

$$\Rightarrow [\![e]\!]_\psi = [\![e']\!]_\psi \overset{\text{(I.H.)}}{\Longrightarrow} e \equiv e'$$

□

# 5 Type Derivations

Provide the type derivation for the following expression:
$true \mid \emptyset \vdash \mathsf{let}\ x = \mathsf{choice}\ \langle \mathsf{A}, 42, \mathsf{true} \rangle\ \mathsf{in}\ x : (\mathsf{Num}, \mathsf{A}), (\mathsf{Bool}, \neg \mathsf{A})$

**Suggested solution:**

$$\text{T-Let}\ \dfrac{\text{T-Choice}\ \dfrac{\text{T-Num}\ \dfrac{42 \in \mathbb{Z}}{\mathsf{A} \mid \emptyset \vdash 42 : (\mathsf{Num}, \mathsf{A})} \qquad \text{T-True}\ \dfrac{}{\neg \mathsf{A} \mid \emptyset \vdash \mathsf{false} : (\mathsf{Bool}, \neg \mathsf{A})}}{true \mid \emptyset \vdash \mathsf{choice}\ \langle \mathsf{A}, 42, \mathsf{true} \rangle : (\mathsf{Num}, \mathsf{A}), (\mathsf{Bool}, \neg \mathsf{A})} \qquad x \notin \mathrm{dom}(\emptyset) \qquad \text{T-Id}\ \dfrac{(x, (\mathsf{Num}, \mathsf{A}), (\mathsf{Bool}, \neg \mathsf{A})) \in \Gamma \qquad true \Rightarrow \mathsf{A} \vee \neg \mathsf{A}}{true \mid (x, (\mathsf{Num}, \mathsf{A}), (\mathsf{Bool}, \neg \mathsf{A})) \vdash x : (\mathsf{Num}, \mathsf{A}), (\mathsf{Bool}, \neg \mathsf{A})}}{true \mid \emptyset \vdash \mathsf{let}\ x = \mathsf{choice}\ \langle \mathsf{A}, 42, \mathsf{true} \rangle\ \mathsf{in}\ x : (\mathsf{Num}, \mathsf{A}), (\mathsf{Bool}, \neg \mathsf{A})}$$

# 6 Code Scattering/Tangling

1. Define the terms *Code Scattering* and *Code Tangling*.

2. Calculate the scattering degree SD for the features *Color* and *Encryption* from the code on the bottom of the page. Then calculate the tangling degree TD of the modules containing those features. Note that every line of code that is only executed if `Conf.ENCRYPTION = true` belongs to the feature *Encryption*. Every line of code that is only executed if `Conf.COLORED = true` belongs to the feature *Color*. The variable `Conf.COLOR` describes the used color if the feature *Color* is activated.

```java
import java.util.ArrayList;
import java.util.List;

public class ChatApplication {
  Client client;
  List<Message> messages =
      new ArrayList<>();

  void newMessage(String content) {
    Message message = new Message(content);
    if (Conf.ENCRYPTION) {
      message.encryptMessage();
    }
    messages.add(message);

  }

  void createClient() {
    if (Conf.COLORED) {
      client = new Client(Conf.COLOR);
    } else {
      client = new Client();
    }
  }
}
```

```java
public class Client {
  Color color;

  public Client(Color color) {
    this.color = color;
  }

  public Client() {
  }

  void displayMessage(Message message) {
    String messageToDisplay =
        message.getContent();
    if (Conf.ENCRYPTION) {
      messageToDisplay =
          decryptMessage(messageToDisplay);
    }
    System.out.println(messageToDisplay);
  }

  String decryptMessage(String message) {
    return new StringBuilder(message)
        .reverse().toString();
  }
}
```

```java
public class Message {
  String content;

  public Message(String content) {
    this.content = content;
  }

  void encryptMessage() {
    content = new StringBuilder(content)
        .reverse().toString();
  }

  public String getContent() {
    return content;
  }

  @Override
  public String toString() {
    return "Message{" +
        "content='" + content + '\'' +
        '}';
  }
}
```

```java
public class Conf {
  public static boolean ENCRYPTION = true;
  public static boolean COLORED = true;
  public static Color COLOR = Color.BLACK;
}
```

**Suggested solution:**

1. *Code Scattering* defines how distributed code belonging to one feature is across multiple modules. *Code Tangling* occurs when code belonging to different features is present in one module.

2. SD(*Color*) = 3, SD(*Encryption*) = 4
   TD(ChatApplication) = 2, TD(Client) = 2, TD(Message) = 1, TD(Conf) = 2

# 7 Merge Conflicts

In the figure below, the revisions B and C have both evolved separately from revision A. Work on the following tasks:

a) Perform an unstructured merge of revisions B and C and mark all merge conflicts.

b) Perform a structured merge of revisions B and C and mark all merge errors.

(a) Revision A

```
class Foo {
        int a = 2;

        void foo() {
                int b = 2;
                int c = a +
                ↪  b;
                return 4 / c;
        }
}
```

(b) Revision B

```
class Foo {
        int a = 2;

        void foo() {
                int b = 2;
                int d = a;
                int c = a -
                ↪  b;
                return c * d;
        }
}
```

(c) Revision C

```
class Foo {
        int f = 2;

        void foo() {
                int b = 2;
                int c = f +
                ↪  b;
                return 4 / c;
        }
}
```

**Suggested solution:**

(a) unstructured

```
class Foo {
        int f = 2;

        void foo() {
                int b = 2;
                int d = a;
                <<<<<<< RevB.java
                int c = a - b;
                =======
                int c = f + b;
                >>>>>>> RevC.java
                return c * d;
        }
}
```

(b) structured

```
class Foo {
        int f = 2;

        void foo() {
                int b = 2;
                int d = a; // merge error
                int c = f - b;
                return c * d;
        }
}
```