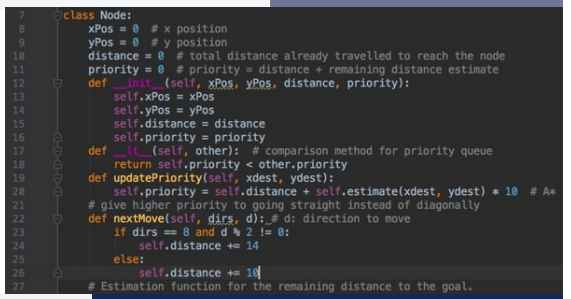
Philipps Universität Marburg

Seminar
Machine Learning in Software Engineering

Prof. Dr. Gabriele Taentzer
Philipps-Universität Marburg

Summer 2023



Philipps Universität Marburg

Introduction to the Subject

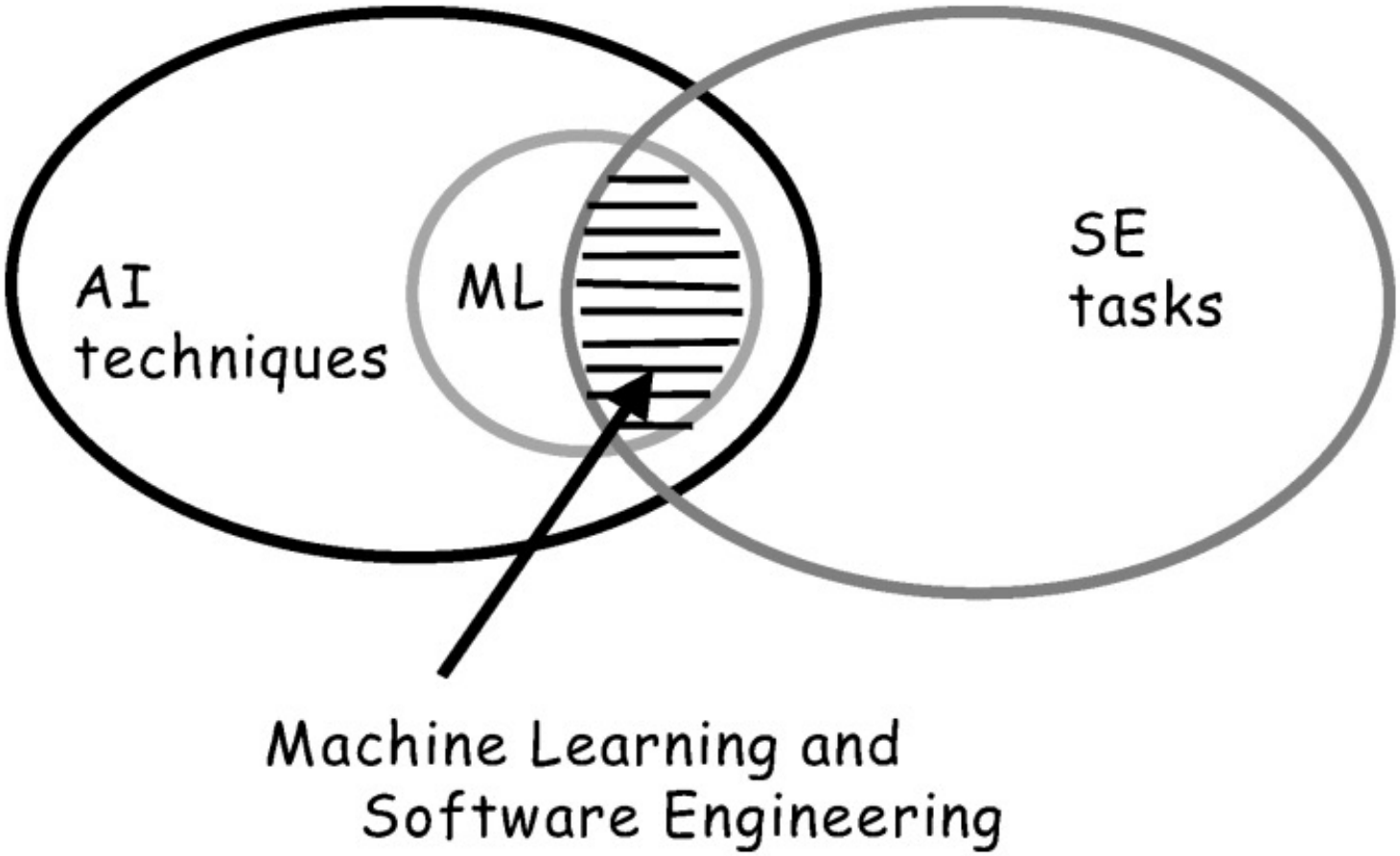
```python
class Node:
    xPos = 0  # x position
    yPos = 0  # y position
    distance = 0  # total distance already travelled to reach the node
    priority = 0  # priority = distance + remaining distance estimate
    def __init__(self, xPos, yPos, distance, priority):
        self.xPos = xPos
        self.yPos = yPos
        self.distance = distance
        self.priority = priority
    def __lt__(self, other):  # comparison method for priority queue
        return self.priority < other.priority
    def updatePriority(self, xdest, ydest):
        self.priority = self.distance + self.estimate(xdest, ydest) * 10  # A*
    # give higher priority to going straight instead of diagonally
    def nextMove(self, dirs, d): # d: direction to move
        if dirs == 8 and d % 2 != 0:
            self.distance += 14
        else:
            self.distance += 10
    # Estimation function for the remaining distance to the goal.
```

Machine learning

Machine learning is useful for

- poorly understood problem domains where little knowledge exists for humans to develop effective algorithms
- domains where there are large databases containing valuable implicit regularities to be discovered
- Domains where programs must adapt to changing conditions

Machine learning in software engineering



Machine learning in software engineering

- Prediction and estimation.

E.g.: software quality, software size, software development cost

- Property and model discovery

E.g. discovery of useful information about software entities, discovery of loop invariants, process model mining

- Transformation

E.g. serial programs into functionally identical parallel programs, improving the modularity of software

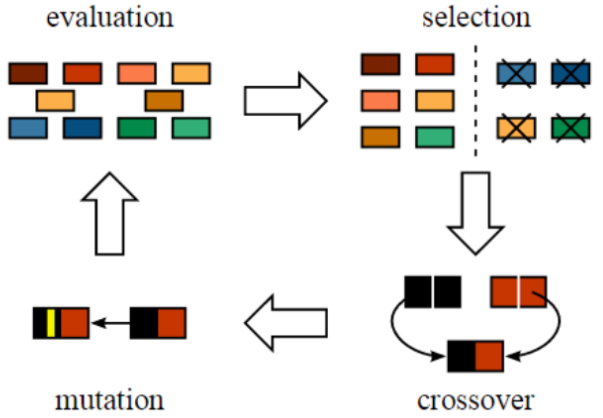
- Generation and synthesis

E.g. **test case generation**, generating project management schedules

- Improvement

E.g. **automatic program repair**

Genetic algorithms



- Given: An optimization problem
  - It may be constrained.
  - It may have several objectives.
- Basic workflow:
  - Start with an initial population
  - While the termination condition is not satisfied
    - Apply some genetic operations (such as mutation and crossover) to individuals of the current population
    - Select the fittest individuals for the next iteration
- Suitable method if
  - There is no domain knowledge.
  - There is no training data.

1. Combining Multiple Coverage Criteria in Search-Based Unit Test Generation

This example shows how EvoSuite covers the method set of the class `ArrayIntList`: the method is called, but statement coverage is not achieved.

```
public class ArrayIntList
       extends RandomAccessIntList
       implements IntList, Serializable {
 public int set(int index, int element) {
   checkRange(index);
   incrModCount();
   int oldval = _data[index];
   _data[index] = element;
   return oldval;
 }
}
```

(a) Source code excerpt.

```
@Test
public void test9() throws Throwable {
 ArrayIntList arrayIntList0 = new ArrayIntList();
 // Undeclared exception!
 try {
   int int0 = arrayIntList0.set(200, 200);
   fail("Expecting IndexOutOfBoundsException");
 } catch(IndexOutOfBoundsException e) {
   // Should be at least 0 and less than 0, found 200
 }
}
```

(b) Test case generated by EvoSuite.

- Implementation of multiple coverage criteria as fitness functions for a search-based test suite optimisation.
- An empirical study of the effects of multiple-criterion optimisation on the effectiveness, convergence, and size of a test suite.

2. Seeding strategies in search-based unit test generation

- When generating unit tests for object-oriented software, there often arises the need to create specific string or numeric values to pass in as parameters. This is called seeding.
- If there is existing knowledge about the class under test in terms of sample values, then these can be used instead of randomly generated values.
- This may lead to an improvement of the overall performance of the test generation, which is typically measured in terms of the achieved code coverage.
- However, it is not clear what influence seeding has on the achievable results and what are the best seeding strategies.

Seeding constants

```
if(x == 27 && y > 250) {
  // ...
}
```

Dynamic seeding

```
if(str1.equals(str2 + "bar")) {
  // ...
}
```

3. An Industrial Evaluation of Unit Test Generation: Finding Real Faults in a Financial Application

```
1  ...
2  if(!StringUtils.isEmpty(objx.getObj().getLocale())){
3    //Faulty Statement
4    Double interest = Double.valueOf(PropertiesReader.
       getProperty("interest.rate.A_" + objx.getObj().
       getLocale() + ""));
5  }
6  ...
7}
```

Code excerpt

Generated test case

```
1public void test4() throws Throwable {
2   ...
3   FaultyClass var0 = new FaultyClass();
4   Objectx var1 = new Objectx();
5   Object var2 = new Object();
6   var2.setLocale("hi");
7   var1.setObj(var2);
8   var0.faultyMethod(var1);
9   ...
10}
```

- How effective are automatically generated unit tests in terms of finding real faults?
- What categories of faults are harder to detect using the current automated test generation tools?
- What major barriers do developers see when adopting automatic test generation tools?

4. ARJA: Automated Repair of Java Programs via Multi-Objective Genetic Programming

- Code is represented as abstract syntax tree (AST)
- ARJA takes a buggy program and a test suite as input (at least one test fails)
- Potentially buggy statements are localized.
- AST is changed with mutation and crossover.
- The fitness evaluation checks how well the modified code passes the tests.
- The search space is reduced by change rules.

No.	Rule	rationale
1	Do not delete a variable declaration statement (VDS).	Deleting a VDS is usually very disruptive to a program, and keeping a redundant VDS usually does not influence the correctness of a program.

```
1092 final StrTokenizer cloned = (StrTokenizer)
         super.clone();
1093 if (cloned.chars != null) {
1094   cloned.chars = cloned.chars.clone();
1095 }
1096 cloned.reset();
1097 return cloned;
```

Code excerpt

5. Shaping Program Repair Space with Existing Patches and Similar Code

```
1 +if(target != null && target.getType()==Token.STRING){
2 -if(target != null){
3    className = target.getString();
4  }
```

Listing 1: The faulty code snippet from *Closure-57*

```
1 if(last != null && last.getType() == Token.STRING){
2   String propName = last.getString();
3   return (propName.equals(methodName));
4 }
```

Listing 2: A similar code snippet to the faulty one

- An automated program repair approach based on the intersection of two search spaces: the search space from existing patches and the search space from similar code.
- A method to obtain a search space from existing patches, based on an abstract space definition on AST types.
- A method to obtain a search space from similar code based on code differencing.
- An experiment on Defects4J that shows the effectiveness of our approach.

6. Automatic Repair of Real Bugs in Java: a Large-Scale Experiment on the Defects4j Dataset

- Defects4J is a large, peer-reviewed, structured dataset of real-world Java bugs.
- Each bug in Defects4J comes with a test suite and at least one failing test case that triggers the bug.
- This paper reports on an experiment to explore the effectiveness of automatic test-suite based repair on Defects4J.
- Three tools for automatic program repair are compared: JGenProg, jKali and Nopol

References

1. Rojas, José Miguel, et al. "Combining multiple coverage criteria in search-based unit test generation." *Search-Based Software Engineering: 7th International Symposium, SSBSE 2015, Bergamo, Italy, September 5-7, 2015, Proceedings 7*. Springer International Publishing, 2015.
2. Rojas, José Miguel, Gordon Fraser, and Andrea Arcuri. "Seeding strategies in search-based unit test generation." *Software Testing, Verification and Reliability* 26.5 (2016): 366-401.
3. Almasi, M. Moein, et al. "An industrial evaluation of unit test generation: Finding real faults in a financial application." *2017 IEEE/ACM 39th International Conference on Software Engineering: Software Engineering in Practice Track (ICSE-SEIP)*. IEEE, 2017.
4. Yuan, Yuan, and Wolfgang Banzhaf. "Arja: Automated repair of java programs via multi-objective genetic programming." *IEEE Transactions on software engineering* 46.10 (2018): 1040-1067.
5. Jiang, Jiajun, et al. "Shaping program repair space with existing patches and similar code." *Proceedings of the 27th ACM SIGSOFT international symposium on software testing and analysis*. 2018.
6. Martinez, Matias, et al. "Automatic repair of real bugs in java: A large-scale experiment on the defects4j dataset." *Empirical Software Engineering* 22 (2017): 1936-1964.

Links

- Test Case Generation
  - EvoSuite: https://www.evosuite.org/
  - Randoop: https://randoop.github.io/randoop/
- Automatic Program Repair
  - Arja : https://github.com/yyxhdy/arja
  - SimFix: https://github.com/xgdsmileboy/SimFix
  - ASTOR: https://github.com/SpoonLabs/astor
- Evaluation data (software projects)
  - Defect4j: https://github.com/rjust/defects4j

Philipps Universität Marburg

Organisation

```
class Node:
    xPos = 0  # x position
    yPos = 0  # y position
    distance = 0  # total distance already travelled to reach the node
    priority = 0  # priority = distance + remaining distance estimate
    def __init__(self, xPos, yPos, distance, priority):
        self.xPos = xPos
        self.yPos = yPos
        self.distance = distance
        self.priority = priority
    def __lt__(self, other):  # comparison method for priority queue
        return self.priority < other.priority
    def updatePriority(self, xdest, ydest):
        self.priority = self.distance + self.estimate(xdest, ydest) * 10  # A*
    # give higher priority to going straight instead of diagonally
    def nextMove(self, dirs, d): # d: direction to move
        if dirs == 8 and d % 2 != 0:
            self.distance += 14
        else:
            self.distance += 10
    # Estimation function for the remaining distance to the goal.
```

Time schedule

- Topic selection until 23.04.2023
- Topic allocation: 24.04.2023
  - Teams of 1 – 3 persons (depending on the topic)
  - Individual discussion of the work per team
    - Please ask me for an appointment.
- Submission of preliminary final version: 30.06.2023
- Block seminar by **arrangement**, between 3.07.2023 and 13.07.2023
- Final submission: 20.07.2023

Seminar registration

- After the topic assignment
- If you want to work on the assigned topic

- Register by 30.04.2023 at the latest

- To register write an email
  - To: pruefungsbuero@mathematik.uni-marburg.de
  - CC: taentzer@mathematik.uni-marburg.de;
  - Content:
    Registration for the seminar “Machine Learning in Software Engineering”
    Name: *Your surname*
    First name: *Your first name*
    Matriculation number: *Your matriculation number*
    Study programme: *Your study programme*

Paper

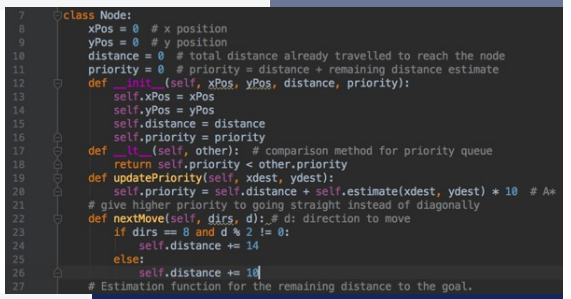
- In LaTeX (text typesetting system)
  - Template on ILIAS
- Length: approx. 3000 words (approx. 6 pages) per group member
  - Additionally: cover page, bibliography, appendix
  - If applicable, (electronic) appendix with code examples, etc.
- Literature:
  - At least one long or two short intensively edited scientific sources per group member.

Presentation

- Block seminar
- Lecture:
  - apprx. 20 minutes per group member
  - Plus 5 minutes per group (for joint introduction/conclusion/etc.)
- After the lecture approx. 10 minutes discussion
- Active participation in discussions
- Compulsory participation

Philipps Universität Marburg

Tips for literature research



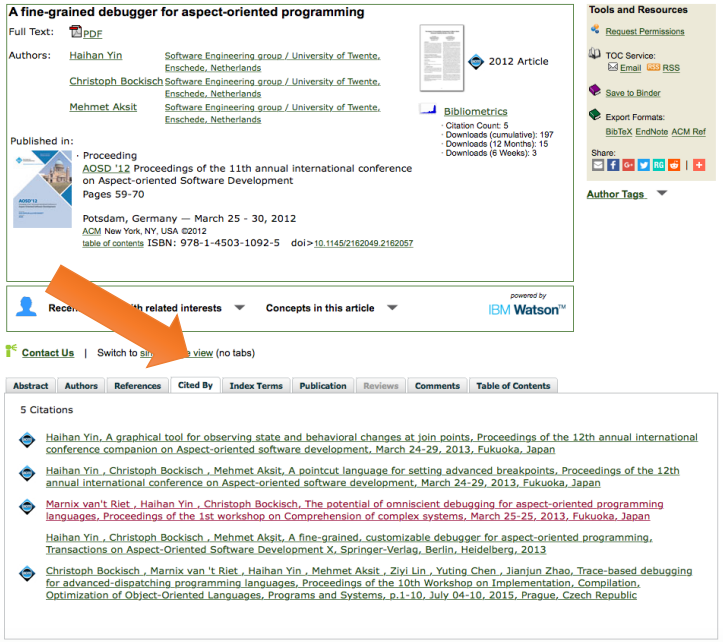
```
class Node:
    xPos = 0  # x position
    yPos = 0  # y position
    distance = 0  # total distance already travelled to reach the node
    priority = 0  # priority = distance + remaining distance estimate
    def __init__(self, xPos, yPos, distance, priority):
        self.xPos = xPos
        self.yPos = yPos
        self.distance = distance
        self.priority = priority
    def __lt__(self, other):  # comparison method for priority queue
        return self.priority < other.priority
    def updatePriority(self, xdest, ydest):
        self.priority = self.distance + self.estimate(xdest, ydest) * 10  # A*
    # give higher priority to going straight instead of diagonally
    def nextMove(self, dirs, d): # d: direction to move
        if dirs == 8 and d % 2 != 0:
            self.distance += 14
        else:
            self.distance += 10
    # Estimation function for the remaining distance to the goal.
```

Literature research

- General directiories:
  - Google Scholar: https://scholar.google.de
  - Microsoft Academic: https://academic.microsoft.com
  - CiteSeerX: http://citeseerx.ist.psu.edu/index
- Typical publishers in computer science
  - ACM: http://dl.acm.org
  - Springer: http://link.springer.com
  - IEEE: http://ieeexplore.ieee.org/Xplore/home.jsp
  - Elsevier: https://www.elsevier.com/advanced-search

Literature research

1. Search by keywords
2. Refine search based on found paper
   - Referenced papers (typically the referenced paper is described and put in relation to the current one)
   - Referenced papers (search directory)
3. About topic-specific conferences / journals
   - Where are the papers found so far published?
   - For which main topic are there conferences?
   - (Search in their proceedings/website, sorted by publication year)
   - General SE conferences / journals
     - ICSE, SPLASH, ECOOP, SE
     - TSE, IEEE Software, TOSEM
   - Ask me for specific conferences / journals



Literature research

- Use Wikipedia?
  - Yes. Why not?
- **Important**:
  - Wikipedia articles (and other online sources) have no assured quality control, so be careful!
- Often there are references to primary literature. Read these as well and, if necessary, prefer to cite primary literature.
- You can cite any source (up to "personal correspondence with ..."), but be clear about this and evaluate sources critically!
- **Important**:
  - Try to find academic sources first
  - For more practical topics, white papers/ documentation/ tutorials/ etc. might be easier to find

Access to literature

- In the network of the University of Marburg you have access to most of the publishers' offers.
  - Also via VPN from home
- For some publications access is restricted, then, e.g.:
  - Google the title of the paper (often there is a "preprint" version on the author's homepage).
  - Write to the authors
  - Ask your supervisor

Managing literature

- Please prepare your paper with LaTeX and BibTex