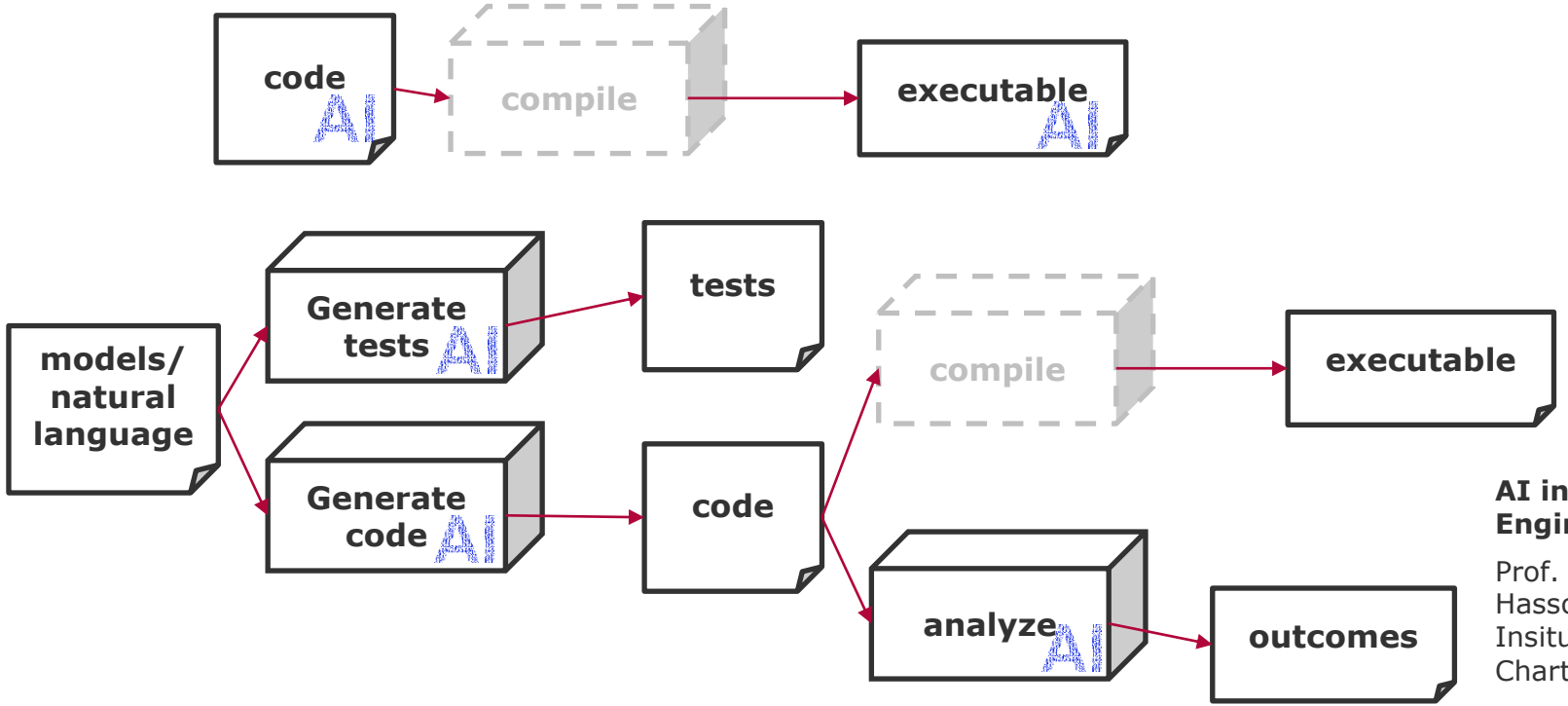
AI in Software Engineering

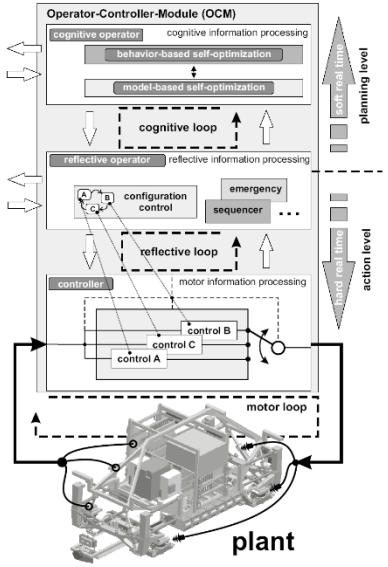
Prof. Dr. Holger Giese and Christian Medeiros Adriano
Head of the System Analysis and Modeling Group
Hasso Plattner Institute at the University of Potsdam

Software Engineering for AI vs. AI in Software Engineering



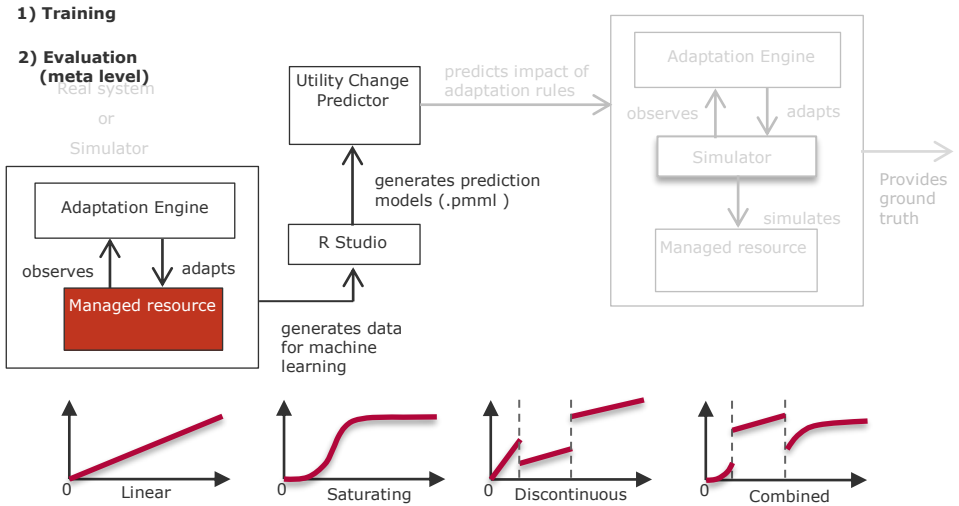
Software Engineering for AI

We worked on …



Sven Burmester, Holger Giese and Oliver Oberschelp. Hybrid UML Components for the Design of Complex Self-optimizing Mechatronic Systems. In Helder Araujo, Alves Vieira, Jose Braz, Bruno Encarnacao and Marina Carvalho editors, Proc. of 1st International Conference on Informatics in Control, Automation and Robotics (ICINCO 2004), Setubal, Portugal, Pages 222-229, INSTICC Press, August 2004.

Self-Adaptive & Train Goals



Sona Ghahremani, Christian M. Adriano and Holger Giese. Training Prediction Models for Rule-Based Self-Adaptive Systems. In 2018 IEEE International Conference on Autonomic Computing (ICAC), Pages 187-192, 2018. Sona Ghahremani, Holger Giese and Thomas Vogel. Improving Scalability and Reward of Utility-Driven Self-Healing for Large Dynamic Architectures. In ACM Trans. Auton. Adapt. Syst., Vol. 14(3), Association for Computing Machinery, New York, NY, USA, February 2020.

AI in Software Engineering

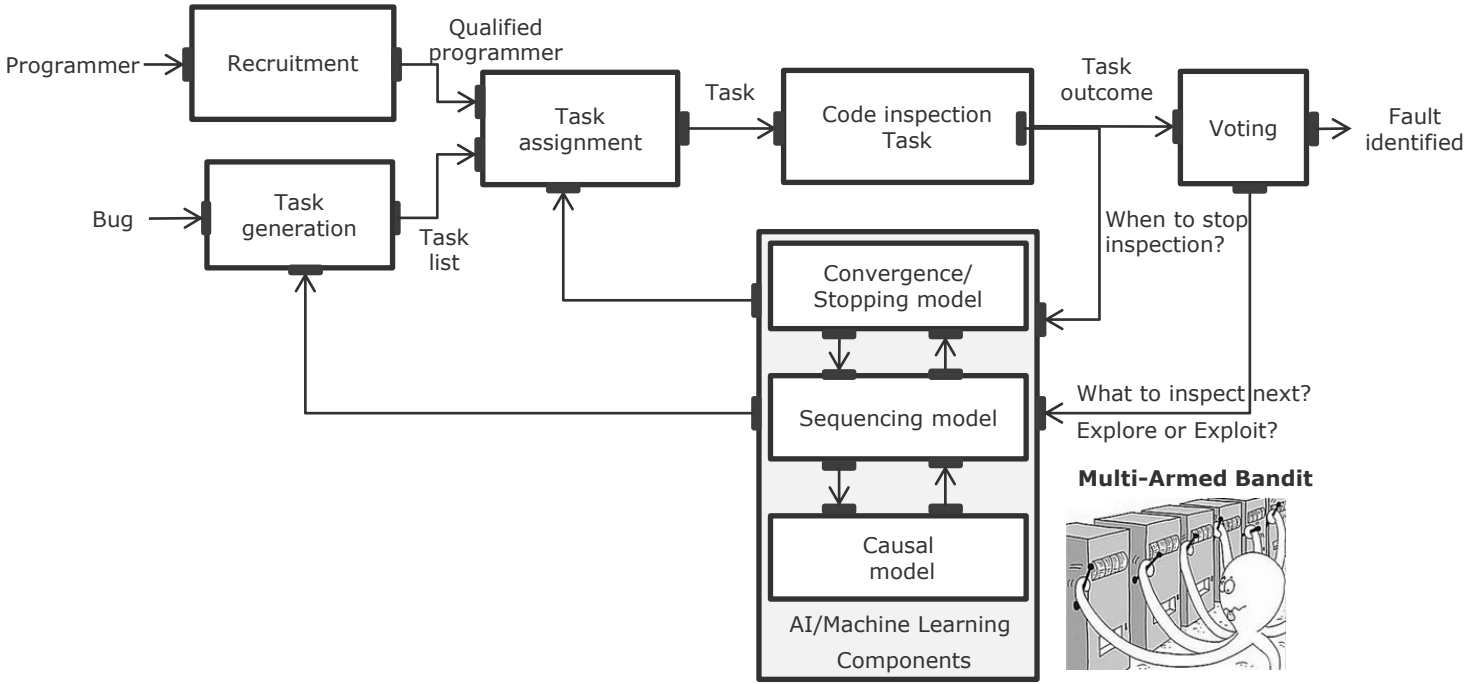
Prof. Holger Giese
Hasso Plattner Insitute

AI for Software Engineering

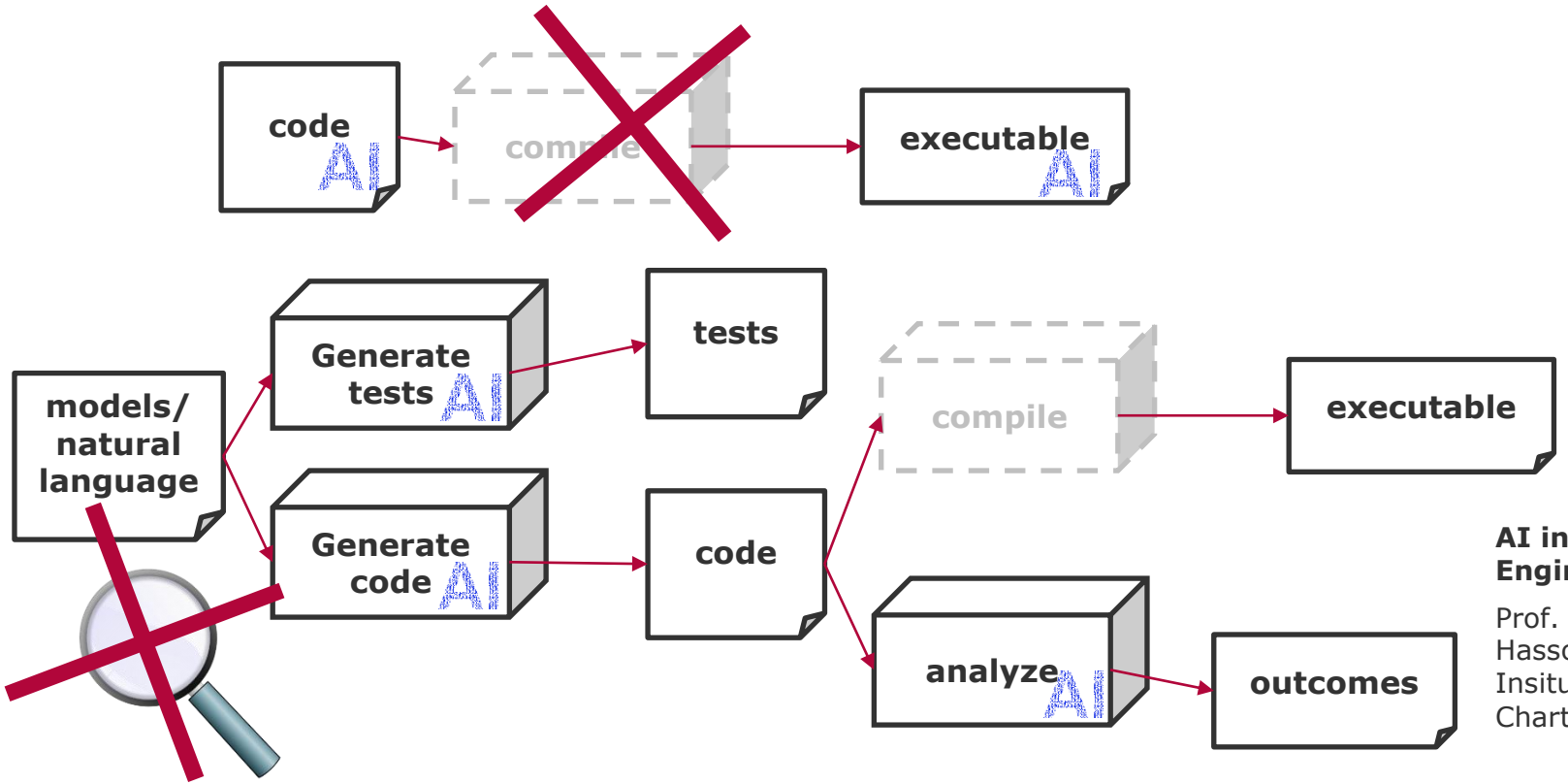
We worked on …

How to optimally allocate code inspection task to minimize cost and time while maximizing the accuracy of software failure diagnostic?

Approach: causal and sequential decision models decide which tasks to generate and who should execute them.



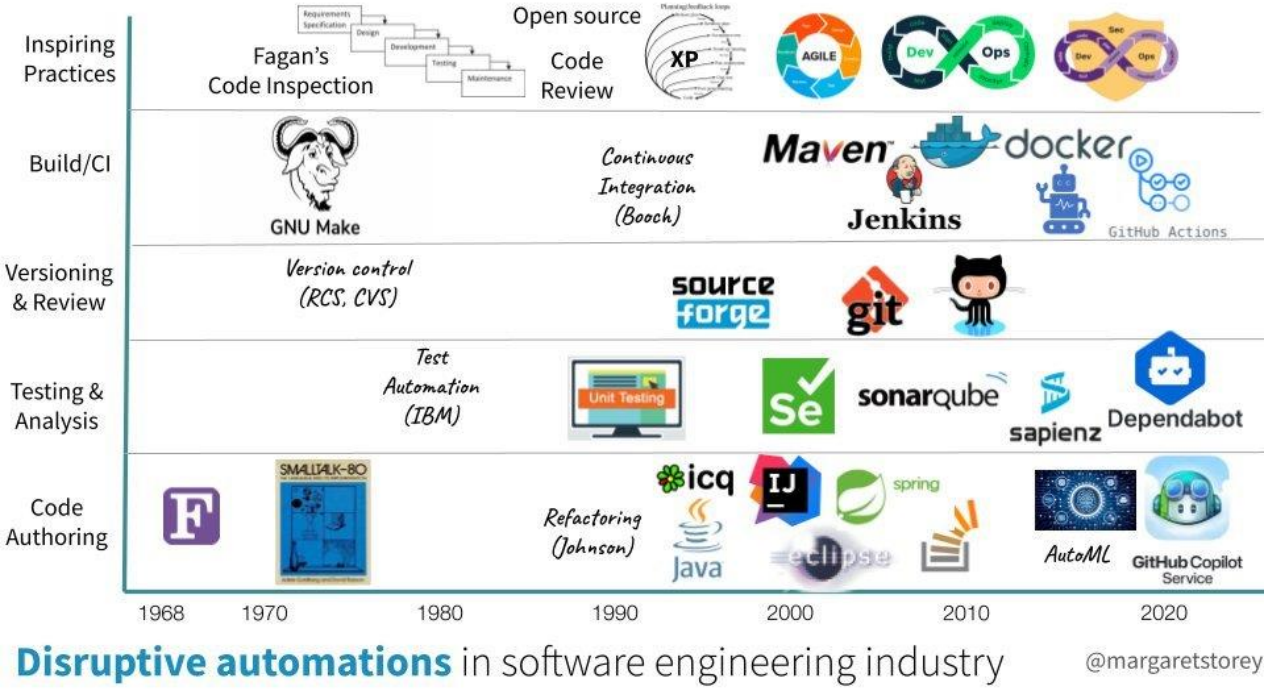
Software Engineering for AI vs. AI in Software Engineering



Agenda

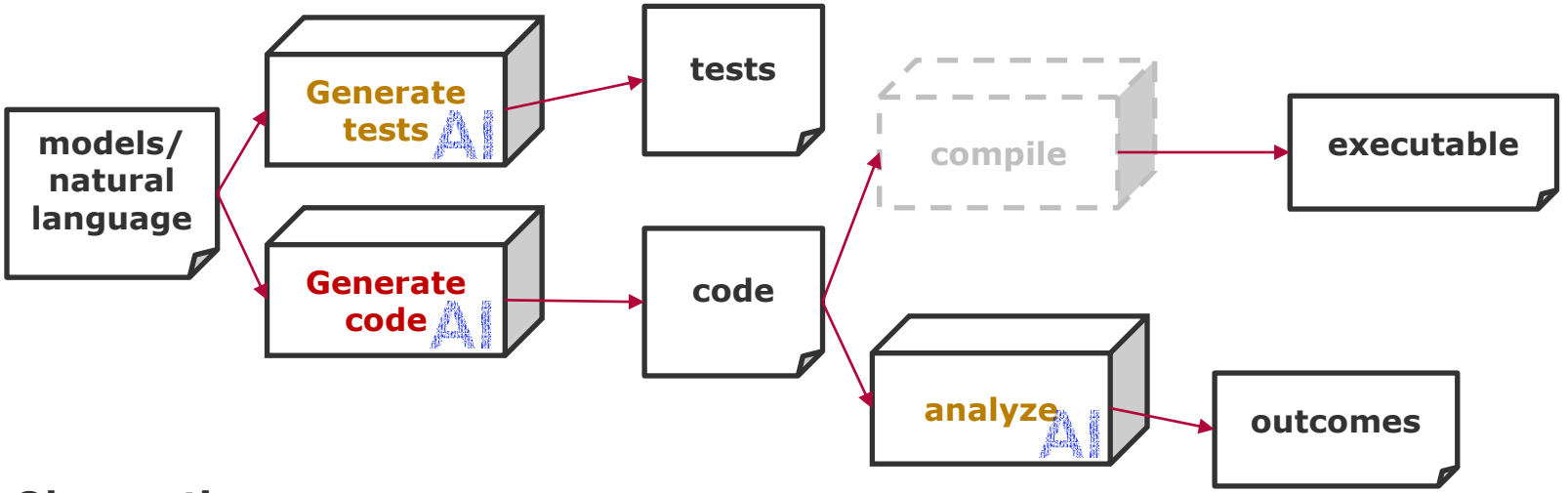
1. Background: Automation in SE
2. Overview: AI in SE
3. Use Case 1: Clone Detection
4. Use Case 2: Code Completion
5. Use Case 3: Code Generation
6. Conclusion & Outlook

1. Background: Automation in SE



https://www.slideshare.net/mastorey/ase-keynote-2022-from-automation-to-empowering-software-developers

Automation in SE



Observations:

- **Automation** that affect the executable is more "dangerous" (generate code, compile, …)
- **Automation** that do not affect the executable is less "dangerous" as it may only affect the executable via human decisions (generate tests, …)

2. Overview: AI in SE

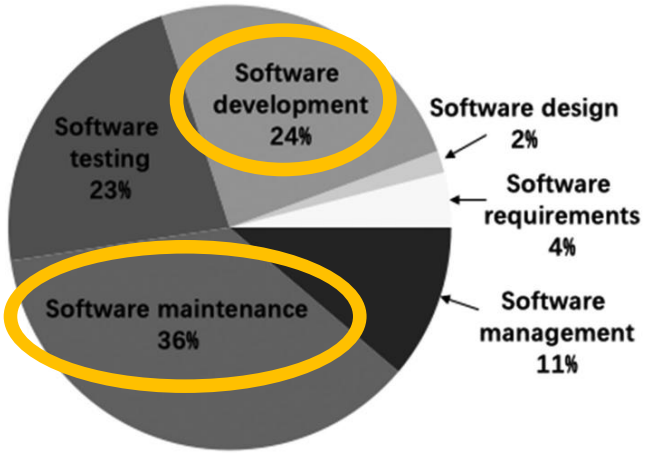


Fig. 2. The distribution of DL techniques in Different SE activities.

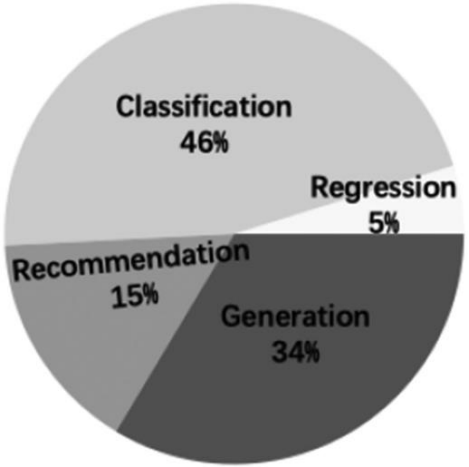


Fig. 3. The classification of primary studies.

[Yang+2922] Yanming Yang, Xin Xia, David Lo and John Grundy. A Survey on Deep Learning for Software Engineering. In ACM Comput. Surv., Vol. 54(10s), Association for Computing Machinery, New York, NY, USA, September 2022

AI in SE: Software Maintenance

Task type	Data type	Dataset	Reference
defect prediction	code-based data	PROMISE dataset	[SP04, IEEE46, IEEE43, IEEE85, IEEE127]
	code-based data	Ivy, jEdit, Log4j, Lucene, PBeans, POI, Synapse, Velocity, Xalan-J, Xerces	[IEEE87, IEEE92]
	code-based data	cleaned NASA , AEEEM datasets	[SP04]
	code-based data	Bugzilla, Columba, Eclipse JDT, Eclipse Platform, Mozilla and PostgreSQL	[IEEE84]
	text-based data	CSIC dataset	[IEEE30]
program repair	code-based data	DeepFix dataset	[AAAI03, IEEE138]
	code-based data	SATE IV	[MITP01]
	code-based data	ETH-Py1502, MSR-VarMisuse	[ICLR02]
	code-based data	SPoC dataset	[ACM11]
	code- and text-based data	Bears, QuixBugs, ManySStuBs4J	[IEEE29]
	code- and text-based data	Bugs.jar	[IEEE29, IEEE46]
	code- and text-based data	Defects4J	[IEEE29, IEEE46, IEEE132]
code clone detection	code-based data	BigCloneBench	[AAAI04, ACM04, IEEE27, MK04, MK05, IEEE110, IEEE38, IEEE51, IEEE55]
	code-based data	OJClone	[AAAI04, MK04, MK05, ACM28, IEEE38]
	code-based data	Google Code Jam	[ACM04, IEEE27]
bug report related	text-based data	Summary Dataset(SDS)	[IEEE137, IEEE39]
	text-based data	Authorship Dataset (ADS)	[IEEE137, IEEE39]
	text-based data	OpenOffice, Eclipse, Net Beans	[IEEE139]
self-admitted technical debt detection	text-based	[14]	[IEEE100]
	text-based data	[1]	[IEEE28]
code review	code-based data	cloudstack, ambari,aurora, drillgit, accumulo and hbase-git.	[AAAI05]
code change	text-based data	QT, OPENSTACK datasets	[IEEE45]
software/code classification	code-based data	[AAAI02]	[AAAI02]

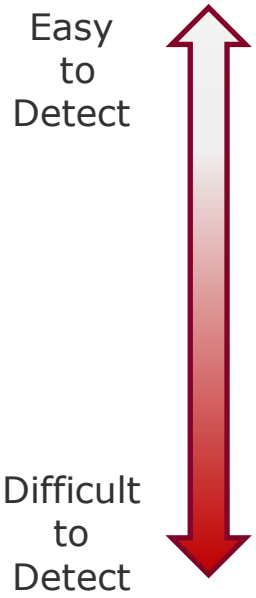
[Yang+2922] Yanming Yang, Xin Xia, David Lo and John Grundy. A Survey on Deep Learning for Software Engineering. In ACM Comput. Surv., Vol. 54(10s), Association for Computing Machinery, New York, NY, USA, September 2022

AI in SE: Software Development

Task type	Data type	common dataset	Reference
Code representation	code-based data	Java-med	[ICLR07]
	code-based data	eth_py150_open	[ACM18, ACM38]
	code-based data	Code.org's Hour of Code (HOC)	[ACM20]
Code generation	code-based data	HearthStone (HS)	[SP09, MITP06]
	code-based data	Karel dataset	[ICLR09, MITP08, ICLR03]
	code-based data	Spider	[MITP06]
	code-based data	DeepCom	[MITP07]
Code comment generation	code-based data	Google Code Jam	[AAAI08]
	text-based data	WMT19	[IEEE36]
	code- and text-baseddata	CODEnn	[IEEE36]
Code search	code- and text-based data	StaQC benchmark	[IEEE02]
	code- and text-based data	CODEnn	[IEEE24]
	code- and text-based data	COsBench	[IEEE24]
Code completion	code-based data	JavaScript (JS)	[MK09]
	code-based data	Python (PY)	[MK09]
Code localization	image-based data	YouTube	[SP08, ACM24, ACM36, IEEE140]
Code summarization	code-based data	NCF representation	[ICLR06]
	code- and text-baseddata	LeClair et al.	[IEEE136]
Method name generation	code- and text-baseddata	MCC corpus	[IEEE59]

[Yang+2922] Yanming Yang, Xin Xia, David Lo and John Grundy. A Survey on Deep Learning for Software Engineering. In ACM Comput. Surv., Vol. 54(10s), Association for Computing Machinery, New York, NY, USA, September 2022

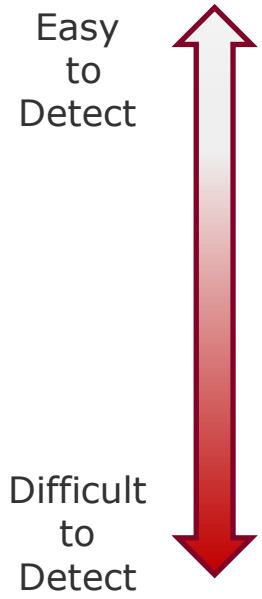
3. Use Case 1: Clone Detection



1. **Type-1** (Textual similarity) = Identical source code fragments (ignore white-space, layout and comments)
2. **Type-2** (Lexical, token-based, similarity) = Identical source code fragments (ignore differences in identifier names)
3. **Type-3** (Syntactic similarity) = Source code fragments that differ at the statement level, e.g., fragments can have statements added, modified and/or removed.
4. **Type-4** (Semantic similarity) = Syntactically distinct source code fragments, **but** that implement the same functionality

[Belon et al. 2007] Bellon, Stefan, et al., 2007. Comparison and Evaluation of Clone Detection Tools. IEEE Transactions on Software Engineering 33, 9, 577-591
[Roy & Cordy 2007] C. K. Roy and J. R. Cordy. 2007. A survey on software clone detection research. Technical Report, Queen's University at Kingston.

Benchmark BigCloneBench



PERCENTAGE OF CLONE TYPES IN BIGCLONEBENCH

Clone Type	Number	Percent(%)
T1	48062	0.56
T2	4614	0.054
VST3	4182	0.053
ST3	16775	0.193
MT3	86109	1.0
WT3/T4	**8416032**	**98.14**

[Svajlenko et al. 2014] Svajlenko, J., 2014 Towards a big data curated benchmark of inter-project code clones, in 2014 IEEE ICSME, pp. 476–480

Rule-Based - SourcererCC

A software S is represented as a collection of code blocks S : {B1, ..., Bn}, where each Bi corresponds to a bag-of-tokens B : {T1..., Tk }, where T is token (method, variable, operator names, etc.)

Assumption: source code follows the Zipf law (similarly as natural language), which preconizes that there are few very popular tokens, and the frequency of tokens decreases very rapidly with popularity rank (Figure-1)

Insight:

- Most code blocks contain one or more of the few very popular tokens (e.g., keywords, counters likes i, j)
- Few code blocks share rare tokens (e.g., identifiers that are domain or project specific).
- Hence, if we sort code blocks by the popularity of tokens in the corpus, the sub-blocks will consist of these rare tokens. This will ensure low probability of different sub-blocks having a similar token.

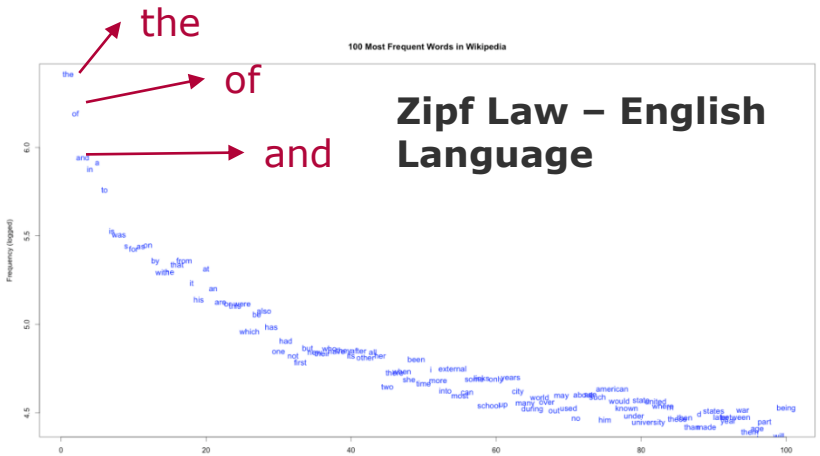


Figure-1 power-law like distribution of token frequency (popularity)

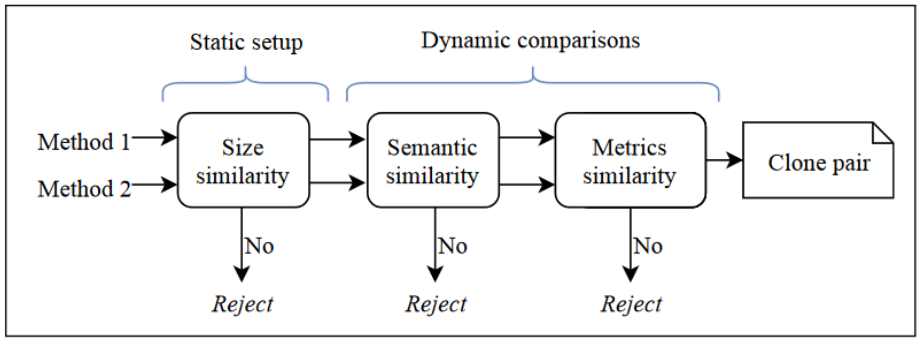
AI in Software Engineering

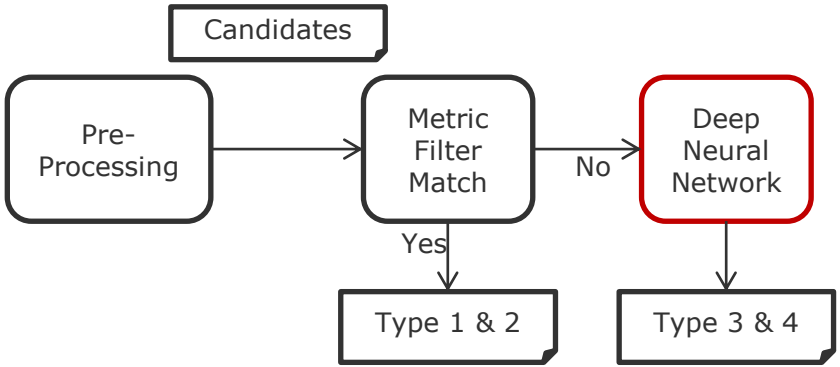
Prof. Holger Giese
Hasso Plattner Insitute

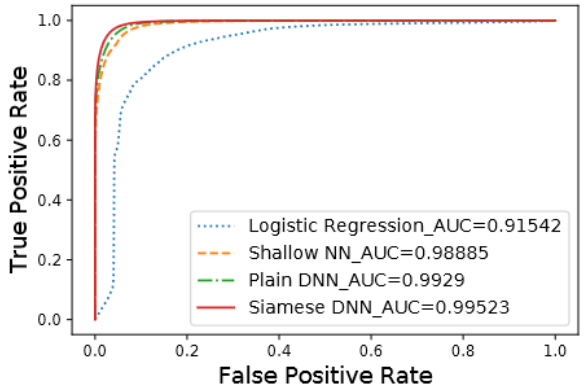
Sajnani, H., et al., 2016, "Sourcerercc: Scaling code clone detection to big-code." *Proceedings of the 38th International Conference on Software Engineering*.
Hindle, A., et al., 2016, "On the naturalness of software." *Communications of the ACM* 59.5: 122-131.
http://wugology.com/zipfs-law/

Hybrid Rule and Learning-based – Oreo

- **Size Similarity:** number of tokens
- **Semantic Similarity**: number of actions tokens (function signature – getByte(), toString(), etc.,) shared by two methods
- **Metrics Similarity**: Halstead effort, Halsted difficulty, Cyclomatic complexity, etc.







Saini, V., et al., 2018, Oreo: Detection of clones in the twilight zone, Proceedings of the 26th ACM

Examples with Oreo

Listing 3: Clone Pair Example: 1

```
1  private void sortByName() {
2      int i, j;
3      String v;
4      for (i = 0; i < count; i++) {
5          ChannelItem ch = chans[i];
6          v = ch.getTag();
7          j = i;
8          while ((j > 0) && (collator.compare(chans[j - 1].getTag(), v) > 0)) {
9              chans[j] = chans[j - 1];
10             j--;
11         }
12         chans[j] = ch;
13     }
14 }
15 ----------------------------------------
16 public void bubblesort(String filenames[]) {
17     for (int i = filenames.length - 1; i > 0; i--){
18         for (int j = 0; j < i; j++) {
19             String temp;
20             if (filenames[j].compareTo(filenames[j + 1]) > 0) {
21                 temp = filenames[j];
22                 filenames[j] = filenames[j + 1];
23                 filenames[j + 1] = temp;
24             }
25         }
26     }
27 }
```

Clones Type-4

Listing 4: Clone Pair Example: 2

```
1  public static String getExtension(final String filename) {
2      if (filename == null || filename.trim().length() == 0 ||
               !filename.contains(".")) return null;
3      int pos = filename.lastIndexOf(".");
4      return filename.substring(pos + 1);
5  }
6  ----------------------------------
7  private static String getFormatByName(String name){
8      if (name != null) {
9          final int j = name.lastIndexOf('.') + 1, k = name.lastIndexOf('/') + 1;
10         if (j > k && j < name.length()) return name.substring(j);
11     }
12     return null;
13 }
```

Clones Type-4

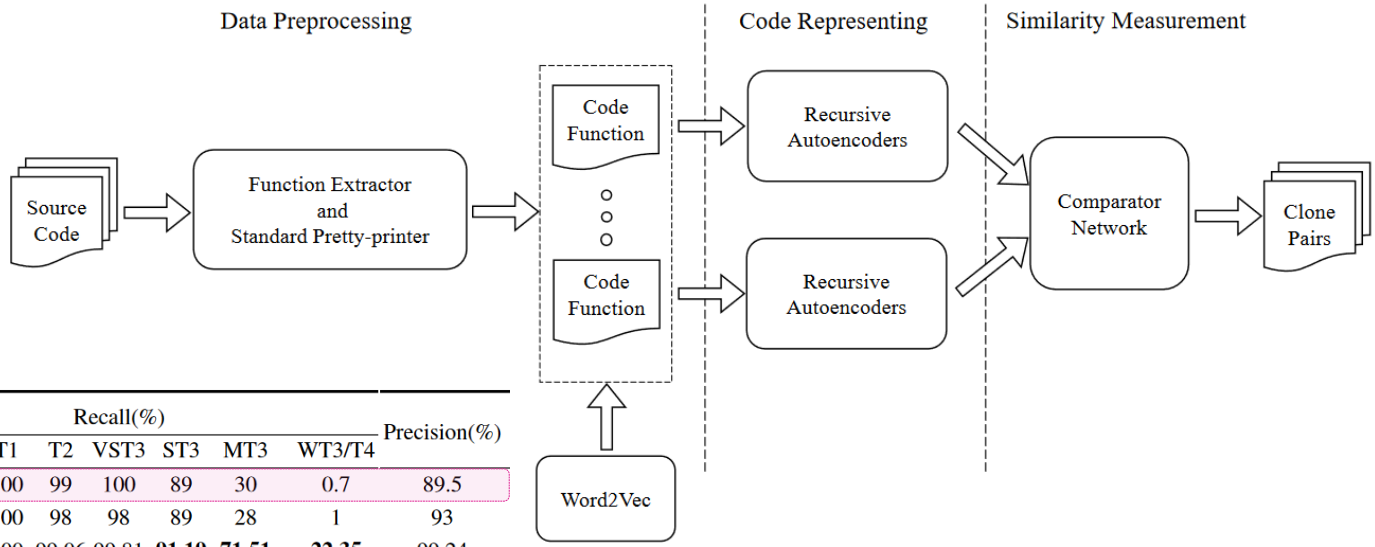
Listing 5: False Positive Example

```
1  public static String getHexString(byte[] bytes) {
2      if (bytes == null) return null;
3      StringBuilder hex = new StringBuilder(2 * bytes.length);
4      for (byte b : bytes) {
5          hex.append(HEX_CHARS[(b & 0xF0) >> 4]).append(HEX_CHARS[(b & 0x0F)]);
6      }
7      return hex.toString();
8  }
9  ----------------------------------------
10 String sequenceUsingFor(int start, int stop) {
11     StringBuilder builder = new StringBuilder();
12     for (int i = start; i <= stop; i++) {
13         if (i > start) builder.append(',');
14         builder.append(i);
15     }
16     return builder.toString();
17 }
```

Clones Type-4

Saini, V., et al., 2018, Oreo: Detection of clones in the twilight zone, Proceedings of the 26th ACM ESC/FCE

Only Learning-based – Sia-RAE



Approaches	Recall(%)						Precision(%)
	T1	T2	VST3	ST3	MT3	WT3/T4	
Oreo	100	99	100	89	30	0.7	89.5
CCLearner	100	98	98	89	28	1	93
Weighted RAE	100	99.06	99.81	**91.19**	**71.51**	**22.35**	99.24
CloneWorks	100	99	98	**93**	3	0	98.7
NiCad	100	99	98	**93**	0.8	0	99
Deckard	60	58	62	31	12	1	34.8
Sia-RAE	99.7	99.2	99	**97.23**	**95.76**	**93.02**	**99.25**

Add embedding!

Feng, C., et al., 2020, Sia-RAE: A Siamese Network based on Recursive AutoEncoder for Effective Clone Detection, in 27th IEEE APSEC

Clone Detection: Discussion

Clone Detection Capabilities

- Rule-based approaches can detect simpler types (T1, T2)
- Learning-based approaches can detect more complex types (T3, T4)
- Deep Learning permits to better detect in particular hard types (ST3, MT3, WT3, WT4)

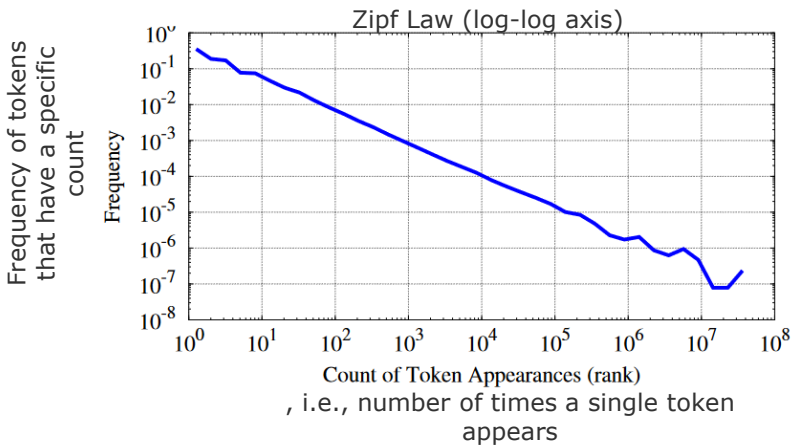
Handling False Positives

- The results must be still manually checked, which can become quite challenging in case of complex/hard cases

Studies about the productivity gains due to better clone detection with AI are missing; likely as the long-term impact on maintenance is very hard to evaluate …

4. Use Case 2: Code Completion

1. **Goal**: given a code context, predict the next token, next line.
2. **Token**: names of methods, variables, operations, types
3. **Facts affecting training** [Allamanis & Sutton 2013]
   - Distribution - Most frequent tokens are few, whereas low frequency are plentiful
   - Power-law - The frequency is a logarithm function of the count of token appearances
   - Context - The more frequent tokens are more context dependent



Allamanis, M., & Sutton, C., 2013, Mining source code repositories at massive scale using language modeling. In IEEE 10th *MRS working conference,* 207-216.

Code Completion: Using Statistics

1. More frequent terms are easier to predict given a context.
2. Code Completion is less certain when predicting domain-specific variable names, because they are less frequent.

Quality depends on More data & Better data

Datasets for Code Completion:

GitHub Python PY150
150K Python files,
113.2M tokens
ETH Py150 Open corpus
7.4M Python file

Github Java Corpus
Projects 11K (training), 4 K(test)
Tokens 1Bi (training), 385M (test)

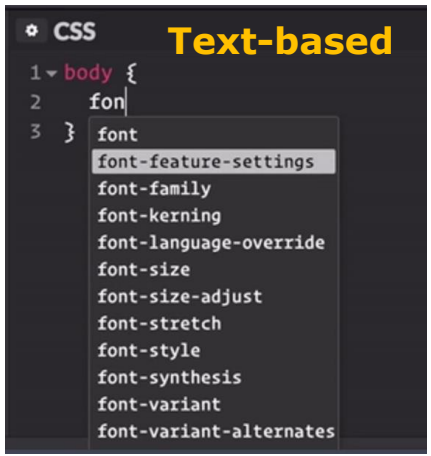
Allamanis, M., & Sutton, C., 2013, Mining source code repositories at massive scale using language modeling. In IEEE 10th *MRS working conference,* 207-216.

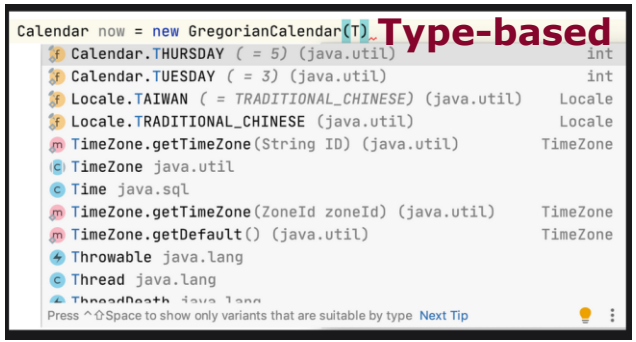
Code Completion:
Traditional vs Deep-Learning

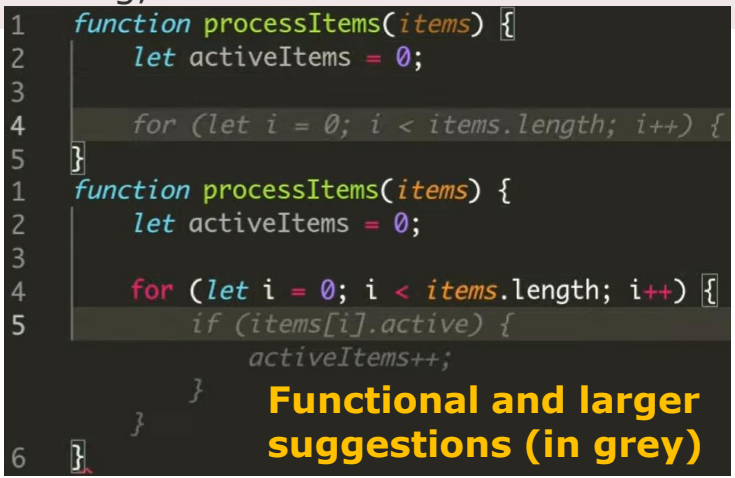
Colored – Typed by user
Grey - Recommendation

Hasso Plattner Institut

	Traditional/Statistical (Built in most IDES)	Deep Learning-Based (CoPilot, Tabnine, Kite)
Input	Previous token	Entire source code and Text comments
Output	Next token or Next line	Naming, code blocks and entire methods

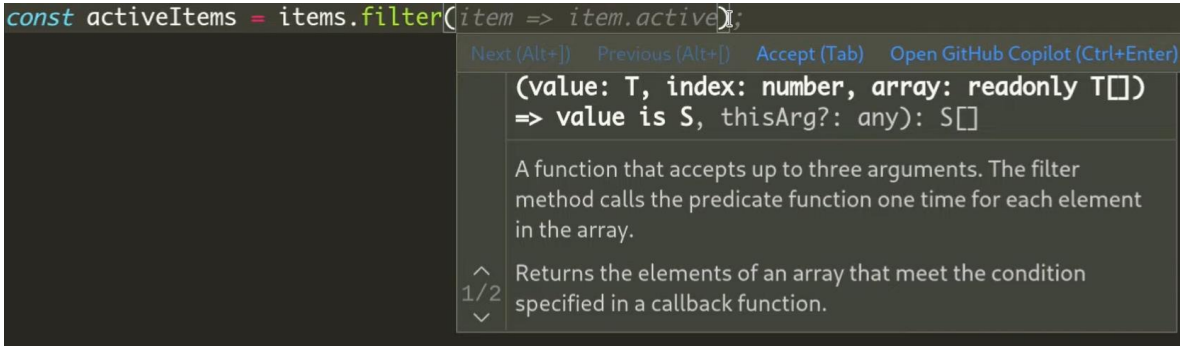






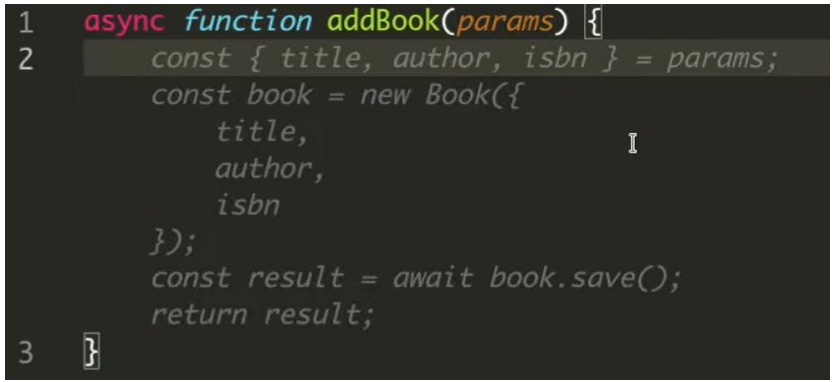
Raychev, V., et al., 2014, Code completion with statistical language models. In *Proceedings of the 35th ACM SIGPLAN,* pp. 419-428.
Tiwari, A., et al., 2022, Survey of Code Completion using DeepLearning.

Code Completion: Deep Learning-Based – CoPilot (1/3)



Colored – Typed by user
Grey - Recommendation

Suggests a whole set of parameters for the function with an explanation.



Suggests the entire body of the function

AI in Software Engineering

Prof. Holger Giese
Hasso Plattner Insitute

GitHub - https://docs.github.com/en/copilot
6 ways GitHub Copilot helps you write better code faster - https://www.youtube.com/watch?v=SXtMnn1v7d8

Code Completion: Deep Learning-Based – CoPilot (2/3)

Hasso Plattner Institut

Step 1: Existing Test Class

Extension: GitHub Copilot | model.test.js

tmp > blah > model.test.js > ...

```
1  import './model.js';
2  import { deepEqual } from 'assert';
3
4  it('should return an empty array by default', () => {
5      const model = new Model();
6
7      deepEqual(model.get(), []);
8  });
9
10 it('should add double the number to the list', () => {
11     const model = new Model();
12
13     model.add(7);
14
15     deepEqual(model.get(), [14]);
16 });
17
```

Step 2: Suggests a whole class based on the file (model.test.js) that existed.

Extension: GitHub Copilot | model.test.js | model.js 1

tmp > blah > model.js > ...

```
1  export default class Model {
2      constructor() {
           this.list = [];
       }

       get() {
           return this.list;
       }

       add(number) {
           this.list.push(number * 2);
       }
   }
```

!!! Knows that the class should add double the number (number * 2)

GitHub - https://docs.github.com/en/copilot

6 ways GitHub Copilot helps you write better code faster - https://www.youtube.com/watch?v=SXtMnn1v7d8

Code Completion – Discussion (1/2)

What do programmers think about it? New Ways of Programming?

"LLM-assisted programming shares some properties of compilation, pair programming, and programming via search and reuse, there are fundamental differences both in the technical possibilities as well as the practical experience.".

[Sarkar et al. 2022] Sarkar, Advait, et al. "What is it like to program with artificial intelligence?." arXiv preprint arXiv:2208.06213 (2022).

Faster: "I think of Copilot as an intelligent autocomplete... I already have the line of code in mind and I just want to see if it can do it, type it out faster than I can."

Reuse: "Copilot feels useful for doing novel tasks that I don't necessarily know how to do. It is easier to jump in and get started with the task".

Nonetheless… "I was about to write the code and I knew what I wanted to write. But now I'm sitting here, seeing if somehow Copilot came up with something better than the person who's been writing Haskell for five years, I don't know why am I giving it the time of day."

[Barke, James & Polikarpova 2022] Barke, Shraddha, Michael B. James, and Nadia Polikarpova. "Grounded copilot: How programmers interact with code-generating models." arXiv preprint arXiv:2206.15000 (2022).

Code Completion - Discussion

Productivity with Code Completion

- Recruited software developers were asked to implement an HTTP server in JavaScript **as quickly as possible**. The treatment group, with access to the AI pair programmer, completed the task 55.8% faster than the control group.

[Peng 2023] Peng, Sida, et al. "The Impact of AI on Developer Productivity: Evidence from GitHub Copilot." arXiv preprint arXiv:2302.06590 (2023).

Quality of Code Completion

- Copilot-generated code is harder to debug
- Programmers validate suggestions by “pattern matching”
- Programmers are reluctant to accept or repair suggestions.

Risks of Code Completion

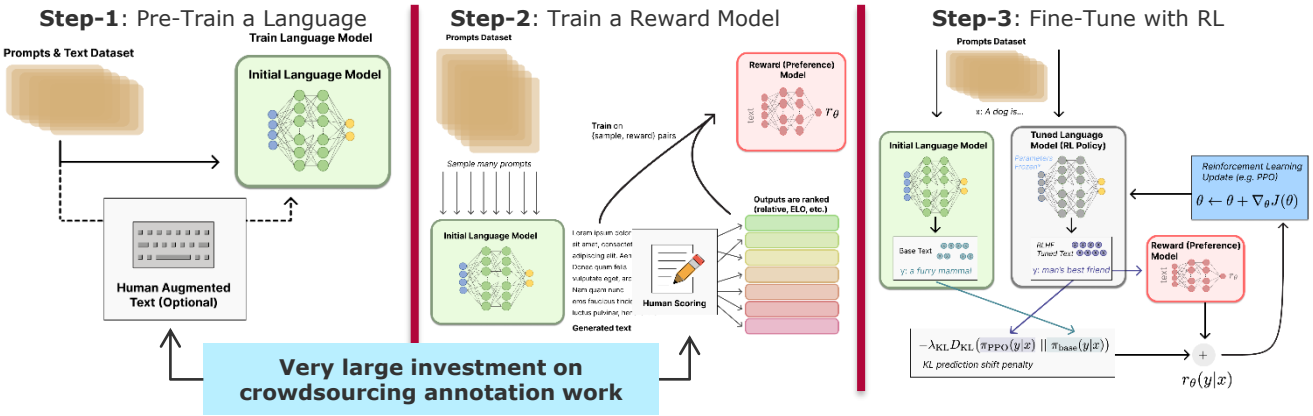
- Programmers suffer from an anchoring bias when looking through multiple suggestions.
- Programmers suffer from cognitive overload due to multi-suggestion pane.

[Barke, James & Polikarpova 2022] Barke, Shraddha, Michael B. James, and Nadia Polikarpova. "Grounded copilot: How programmers interact with code-generating models." arXiv preprint arXiv:2206.15000 (2022).

5. Use Case 3: Code Generation

ChatGPT = InstructGPT + RLHF

1. **ChatGPT** = conversational AI, which is very challenging because human language is ambiguous, and conversations are bounded by contextual information.
2. **Instruct GPT** = language model trained to follow instructions from a prompt
3. **RLHF (Reinforcement Learning from Human Feedback)** = human gives feedback on a generated text in order to align the pre-trained language model with complex human values (what is funny, ethical, safe).

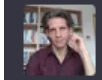


AI in Software Engineering

Prof. Holger Giese
Hasso Plattner Insitute

Lambert et al., 2022, Illustrating Reinforcement Learning from Human Feedback (RLHF), Hugging Face- https://github.com/huggingface/blog/blob/main/rlhf.md
Ouyang , L. et al., 2022, Training language models to follow instructions with human feedback. arXiv preprint arXiv:2203.02155

Code Generation (1/6)



Write a program a that “shuts off the pumps if the water level remains above 100 m for more than 4 seconds.”

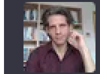
1st version

As an AI language model, I don't have direct access to the physical pumps and water level sensors to implement the program. However, I can provide a basic pseudocode implementation that describes the logic of the program:

```java
set pump_status = "off"
set water_level_timer = 0

while true:
    water_level = read_water_level_sensor()

    if water_level > 100:
        water_level_timer += 1
    else:
        water_level_timer = 0

    if water_level_timer >= 4:
        pump_status = "off"
        turn_off_pumps()
    else:
        pump_status = "on"
        turn_on_pumps()

    wait_for_next_iteration()
```

David Parnas, Jan Madey and G. Asmis. Assessment of safety-critical software in nuclear power plants. In Nuclear Safety, Vol. 32, 1991

Code Generation (2/6)



Write a program a that “shuts off the pumps if the water level remains above 100 m for more than 4 seconds.”

2nd version

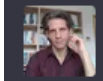
I'm sorry, as an AI language model, I do not have the capability to directly control physical devices or write executable programs that interact with the physical world. However, I can provide a high-level algorithm that could be used to achieve the desired functionality:

1. Continuously monitor the water level sensor.
2. If the water level is above 100 m, start a timer and set a flag indicating that the water level has been above 100 m.
3. If the water level drops below 100 m, reset the timer and clear the flag.
4. If the timer reaches 4 seconds and the flag is still set, shut off the pumps.

This algorithm can be implemented using a variety of programming languages, platforms, and hardware components, depending on the specific requirements of the application.

David Parnas, Jan Madey and G. Asmis, Assessment of safety-critical software in nuclear power plants. In Nuclear Safety, Vol. 32, 1991

Code Generation (3/6)



Write a program a that “shuts off the pumps if the water level remains above 100 m for more than 4 seconds.”

3rd version

As an AI language model, I don't have direct access to the physical pumps and water level sensors to implement the program. However, I can provide a basic pseudocode implementation that describes the logic of the program:

python Copy code

```python
import time

# initialize water level and pump status
water_level = 0
pumps_on = False
pump_on_time = 0

# loop to simulate water level changes
while True:
    # check water level
    if water_level > 100:
        # turn pumps off if they are on
        if pumps_on:
            pumps_on = False
            print("Pumps turned off")
        # record time if pumps are off
        else:
            pump_off_time = time.time()
            # check if water level has been above 100 for more than 4 seconds
            if pump_off_time - pump_on_time > 4:
                print("Water level above 100 for more than 4 seconds. Pumps turn
                break
    else:
        # turn pumps on if they are off
        if not pumps_on:
            pumps_on = True
            pump_on_time = time.time()
            print("Pumps turned on")
    # simulate water level changes
    water_level += 1
    time.sleep(1)
```

David Parnas, Jan Madey and G. Asmis. Assessment of safety-critical software in nuclear power plants. In Nuclear Safety, Vol. 32, 1991

Code Generation (4/6)

java

```
set pump_status = "off"
set water_level_timer = 0

while true:
    water_level = read_water_level_sensor()

    if water_level > 100:
        water_level_timer += 1
    else:
        water_level_timer = 0

    if water_level_timer >= 4:
        pump_status = "off"
        turn_off_pumps()
    else:
        pump_status = "on"
        turn_on_pumps()

    wait_for_next_iteration()
```

1st version

3rd version

python Copy code

```
import time

# initialize water level and pump status
water_level = 0
pumps_on = False
pump_on_time = 0

# loop to simulate water level changes
while True:
    # check water level
    if water_level > 100:
        # turn pumps off if they are on
        if pumps_on:
            pumps_on = False
            print("Pumps turned off")
        # record time if pumps are off
        else:
            pump_off_time = time.time()
            # check if water level has been above 100 for more than 4 seconds
            if pump_off_time - pump_on_time > 4:
                print("Water level above 100 for more than 4 seconds. Pumps turn
                break
    else:
        # turn pumps on if they are off
        if not pumps_on:
            pumps_on = True
            pump_on_time = time.time()
            print("Pumps turned on")
    # simulate water level changes
    water_level += 1
    time.sleep(1)
```


David Parnas, Jan Madey and G. Asmis. Assessment of safety-critical software in nuclear power plants. In Nuclear Safety, Vol. 32, 1991

Code Generation (5/6)

Write a program in Java to shut off the pumps if the water level remains above 100 m for more than 4 seconds.

Sure, here's an example Java program that can shut off the pumps if the water level remains above 100 m for more than 4 seconds:

Note that this code assumes you have a way to read the water level sensor input and control the pump. You may need to modify the code to match your specific hardware and libraries. Additionally, please test and validate any code you use for controlling important equipment like pumps, and follow appropriate safety protocols.

Java code. It compiles and it is well structured.

java Copy code

```java
import java.util.Timer;
import java.util.TimerTask;

public class WaterLevelMonitor {
    private static final int PUMP_PIN = 12;
    private static final int WATER_LEVEL_PIN = 18;
    private static final int WATER_LEVEL_THRESHOLD = 100;
    private static final int SHUTOFF_DELAY = 4000; // in milliseconds

    private Timer shutoffTimer;

    public WaterLevelMonitor() {
        // Set up GPIO pins here if necessary
    }

    public void startMonitoring() {
        shutoffTimer = new Timer();
        shutoffTimer.schedule(new TimerTask() {
            private boolean waterLevelAboveThreshold = false;
            private long startMillis = 0;

            @Override
            public void run() {
                // Check water level
                boolean waterLevelAboveThresholdNow = checkWaterLevel();
                if (waterLevelAboveThresholdNow && !waterLevelAboveThreshold) {
                    // Water level just went above threshold
                    waterLevelAboveThreshold = true;
                    startMillis = System.currentTimeMillis();
                } else if (!waterLevelAboveThresholdNow) {
                    // Water level below threshold
                    waterLevelAboveThreshold = false;
                    startMillis = 0;
                } else if (waterLevelAboveThresholdNow && waterLevelAboveThresh
                        && (System.currentTimeMillis() - startMillis) >= SHUTOF
                    // Water level has been above threshold for more than SHUTO
                    shutoffPump();
                }
            }
        }, 0, 100); // check every 100 milliseconds
    }

    public void stopMonitoring() {
        shutoffTimer.cancel();
    }

    private boolean checkWaterLevel() {
        // Implement code to read water level from sensor
        // Return true if water level is above threshold, false otherwise
        return true;
    }

    private void shutoffPump() {
        // Implement code to shut off pump
    }

    public static void main(String[] args) {
        WaterLevelMonitor monitor = new WaterLevelMonitor();
        monitor.startMonitoring();

        // Keep program running until interrupted
        while (true) {
            try {
                Thread.sleep(1000);
            } catch (InterruptedException e) {
                break;
            }
        }

        monitor.stopMonitoring();
    }
}
```

Parnas, D., et al., Assessment of safety-critical software in nuclear power plants. In Nuclear Safety, Vol. 32, 1991

Code Generation (6/6)

"Shut off the pumps if the water level remains above 100 m for more than 4 seconds."

Ambiguity and natural language ➪ Many possible interpretations …

"Shut off the pumps if the mean water level over the past 4 seconds was above 100 m."

$$\left[\int_{\tau-4}^{\tau} WL(t)dt\right]/4 > 100$$

"Shut off the pumps if the median water level over the past 4 seconds was above 100 m."

$$\left\{MAX_{(\tau-4,\tau)}\left[WL(t)\right] + MIN_{(\tau-4,\tau)}\left[WL(t)\right]\right\}/2 > 100$$

"Shut off the pumps if the minimum water level over the past 4 seconds was above 100 m."

$$MIN_{(\tau-4,\tau)}\left[WL(t)\right] > 100$$

David Parnas, Jan Madey and G. Asmis. Assessment of [illegible] nuclear power plants. In Nuclear Safety, Vol. 32, 1991

Code Generation - Discussion

```java
set pump_status = "off"
set water_level_timer = 0

while true:
    water_level = read_water_level_sensor()
```

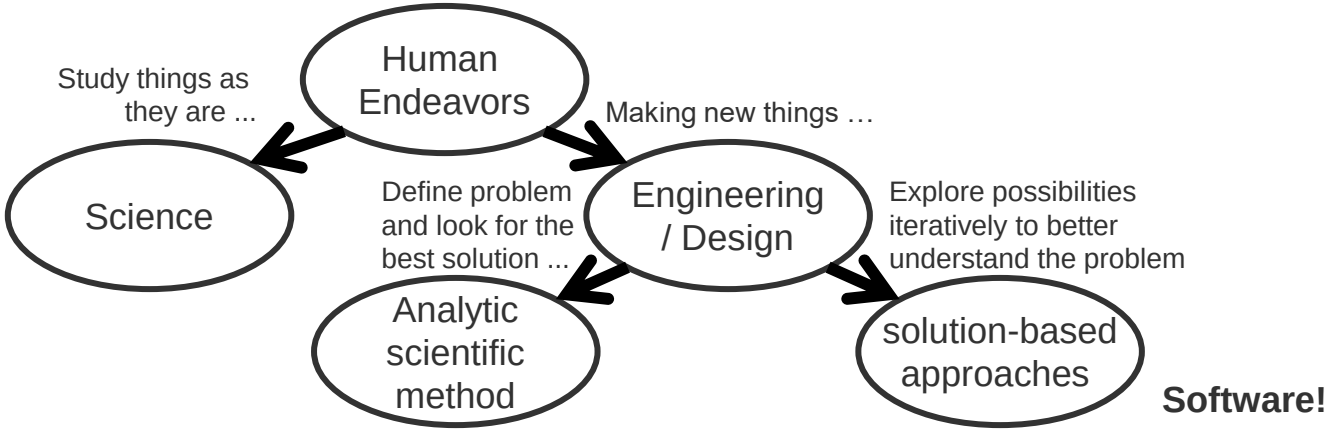
```python
import time

# initialize water level and pump status
water_level = 0
pumps_on = False
pump_on_time = 0

# loop to simulate water level changes
while True:
    # check water level
    if water_level > 100:
        # turn pumps off if they are on
        if pumps_on:
            pumps_on = False
            print("Pumps turned off")
        # record time if pumps are off
```

- Code has still to be adjusted
  - Positioned in the code
  - Replace variables
  - Use the right functions/operations
  - …
- Code may be wrong
  - Pumps are shut off too early in version 2
- Best code offered (1st, 2nd, …) must be identified and adapted/corrected
- **Reuse?** But produces many clones …

6. Conclusion & Outlook

- Automation will substantially improve using AI (resp. has already)
- **Automation** that **does not affect** the executable can therefore **better** support human decisions
  - **Pro:** better clone detection can result in **better** maintenance decisions
  - **Cons:** too much **trust** can also here lead also to wrong decisions
- **Automation** that **affect** the executable raises very subtle problems
  - humans must **evaluate** the outcomes for the specific context (e.g., check that the code really does what is needed (do they know? corner-cases?),
  - humans must **adapt** the outcomes to the specific context (e.g., replace variables (may become quite complex)), and
  - humans may have to **change** the outcomes later or regenerate place them (**maintenance** may become harder? trade-off!).

Outlook – Wicked Problems



- Well-defined problems have specific goals, clearly defined solution paths, and clear expected solutions.

➔ Engineering becomes an optimization problem

- Ill-defined problems are those that do not have clear goals, solution paths, or expected solution.
- Wicked problems are ill-defined problems that are not understood until after the formulation of a solution [Jeffrey2006]

➔ Design becomes an iterative search problem

[Jeffrey2006] Jeffrey Conklin. Dialogue mapping: building shared understanding of wicked problems. Wiley Publishing, Chichester, England, 2006.

Outlook – Empower Developer

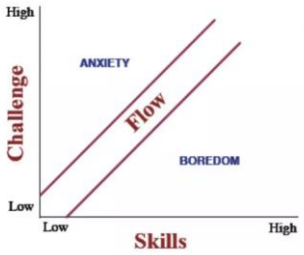
*"**Shared theories** about problem and solution domains that are in the minds of developers"* [Peter Naur, 1985]

*"Every person's **mental model** is incomplete and out of date but we need them all!"* [Jessica Kerr, Explore DDD Conference, 2018]

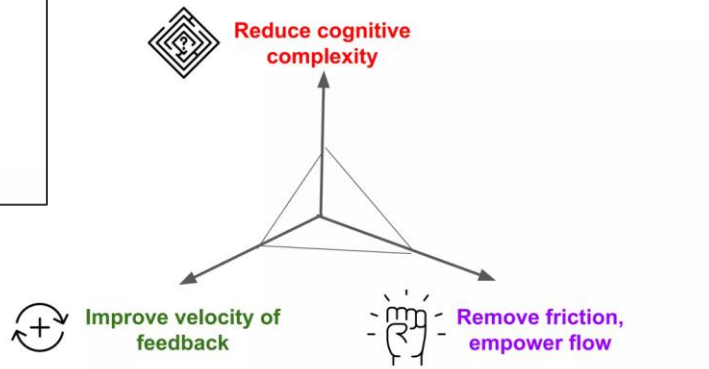
*"**People** are part of the code"* [Jean Yang, QCon Panel, 2021]

Flow is a state of mind, a holistic sensation, that people feel when they act with total involvement [Mihaly Csikszentmihalyi]

Clear goals
Total sense of **involvement**
Loss of **self-consciousness**
Feeling of **control** and being in control
Altered **sense of time**
Above average **skills** and **challenges**



40 | Developer Flow and Fulfillment



50 | The dimensions of developer experience may be in **tension**

https://www.slideshare.net/mastorey/ase-keynote-2022-from-automation-to-empowering-software-developers

Thank you for your attention!

Prof. Dr. Holger Giese and Christian Medeiros Adriano
Head of the System Analysis and Modeling Group
Hasso Plattner Institute at the University of Potsdam

Backup Slides

Prof. Dr. Holger Giese and Christian Medeiros Adriano
Head of the System Analysis and Modeling Group
Hasso Plattner Institute at the University of Potsdam

AI in SE: Software Testing

Task type	Data type	Dataset	Reference
Bug detection	code-based data	Linux, MySQL and Apache HTTPD server	[IEEE68]
	code-based data	[12] includes 986 apps,578 normal apps and 408 vulnerable apps	[EL17]
	text-based data	bug report from Mozilla	[EL12]
Bug localization	code-based data	Defects4J benchmark	[IEEE40]
	text-based data	[13] bug report benchmark	[IEEE40]
	code- and text-based data	AspectJ in Bugzilla, SWT, JDT, Tomcat	[IEEE135]
Vulnerability detection	code-based data	MC&NH dataset	[EL03]
	text-based data	CVE Details websit	[IEEE57]
Test case generation	code-based data	REAPER	[IEEE17]
	image-based data	MNIST, fashion-MNIST	[IEEE93]
	image-based data	CIFAR-10, CIFAR-100	[IEEE93]
Program analysis	text-based data	static analysis alarm data	[IEEE62]
bug classification	code-based data	MozillaProject, Radare2Project	[EL13]

[Yang+2922] Yanming Yang, Xin Xia, David Lo and John Grundy. A Survey on Deep Learning for Software Engineering. In ACM Comput. Surv., Vol. 54(10s), Association for Computing Machinery, New York, NY, USA, September 2022

Code Completion: Deep Learning-Based – CoPilot (3/3)

Hasso Plattner Institut

Or suggests a SQL code

```
1  async function addBook(params) {
2      await query(`INSERT INTO books (title, author, isbn, description, image, price,
       quantity) VALUES ('${params.title}', '${params.author}', '${params.isbn}', '${params.
       description}', '${params.image}', '${params.price}', '${params.quantity}')`);
3  }
```

Hazard!

However, accepting this might create a SQL injection vulnerability. The reason is that content of params is passed "as-is" without any protection*

*Four options of avoiding SQL Injection:

1. Use of Prepared Statements (with Parameterized Queries)
2. Use of Properly Constructed Stored Procedures
3. Allow-list Input Validation
4. Escaping All User Supplied Input

https://cheatsheetseries.owasp.org/cheatsheets/SQL_Injection_Prevention_Cheat_Sheet.html

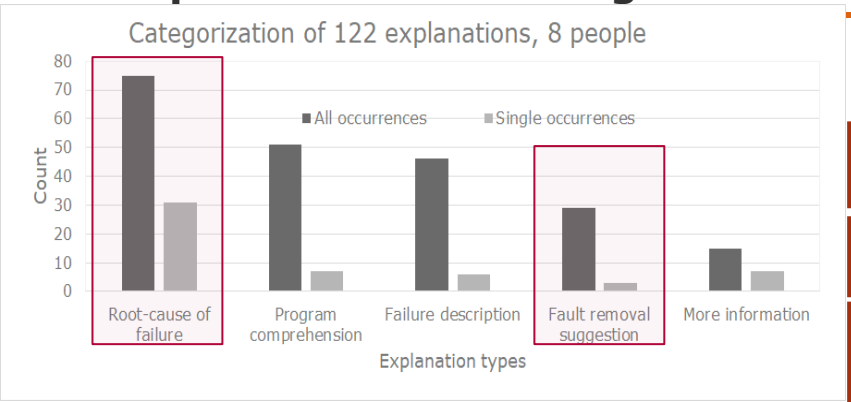
GitHub - https://docs.github.com/en/copilot
Source: 6 ways GitHub Copilot helps you write better code faster - https://www.youtube.com/watch?v=SXtMnn1v7d8

Summary of Approach and Results

Chris' PhD Thesis

Impact of Fault Understanding on bug fixing

Do explanations make bug fixes more accurate?



Yes, but only for explanations that reflect fault understanding

21 programmers, average of 11.7 years of experience

Statistic	No Explanation	Root-Cause	Fault removal
Accuracy	46%	81%	100%
Statistic	No Explanation < Root-cause	Root-Cause < Fault removal	No Explanation < Fault removal
p-value	0.012	0.018	0.00001
Cohen-d	0.74	0.68	1.48
Strength	Large	Large	Large
pos-hoc Power	48%	43%	94%

Summary of Results

1: What are the factors related to the correct identification of a software fault?

Besides programming skill and professional background, the programmer's confidence and her perceived difficulty of the task comprised to the main factors, i.e., the stronger interventional effects measured by the causal models.

2: How many replications are necessary until we correctly recognize a software fault?

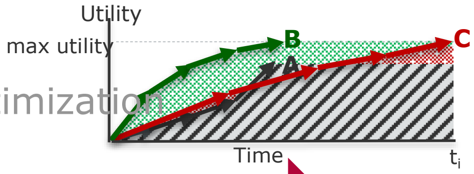
On average 4 times (20% x 20)

3: Are programmers more accurate in their bug fixes if they have access to the explanations?

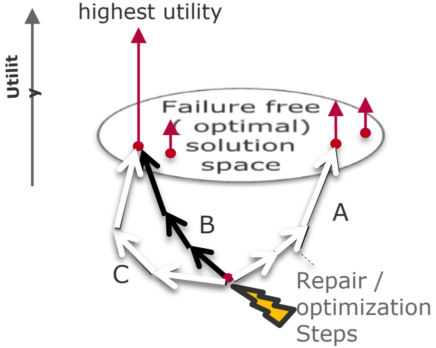
Yes, programmers are more accurate if they have access to two types of explanations: the root-cause and the fault removal suggestion.

Complex Self-Awareness & Train Goals (1/4)

[Ghahremani+2017]
[Ghahremani+2018]



Self-adaptive systems that are rule-based
Architecture-based self-healing and self-optimization



+

	Optimization -based (C)	Rule -based (A)	Utility -driven (B)
Optimal order of repairs	✓	✗	✓
Scalable	✗	✓	✓
Maximum Utility	✓	✗	✓
Expressiveness	+	–	+/–

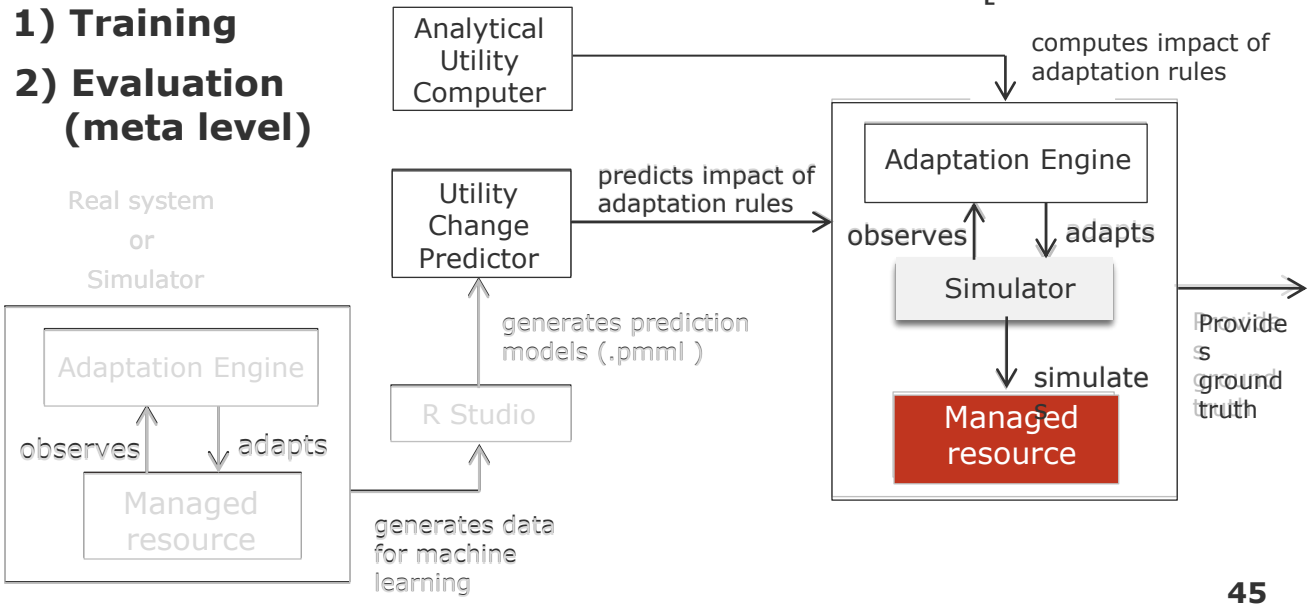
Required: Function computing the impact on the utility for each possible rule application
Open Question: Can we learn these functions offline (training)?

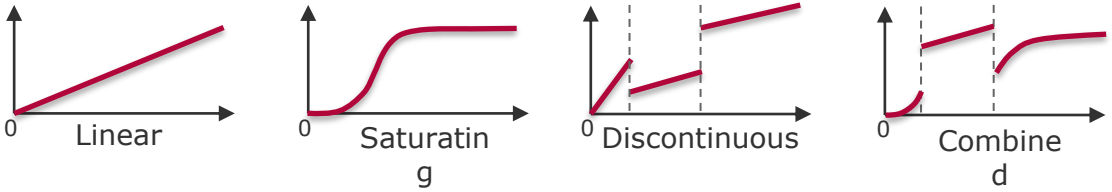
Complex Self-Awareness & Train Goals (2/4)

[Ghahremani+2018]

1) Training

2) Evaluation (meta level)



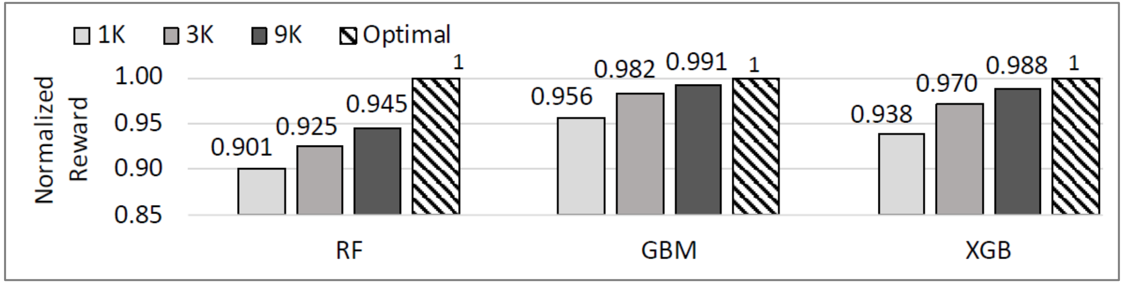


Complex Self-Awareness & Train Goals (3/4)

[Ghahremani+2018]

RQ: Does the performance approximate the analytic-defined optimum?

YES



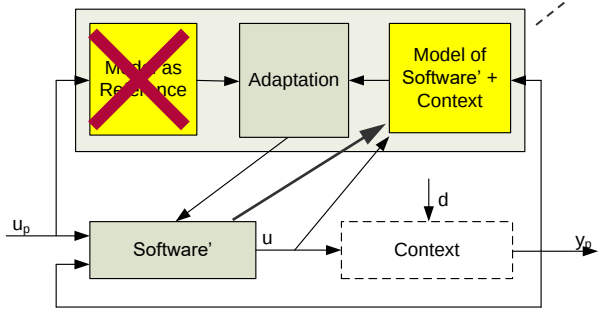
Normalized rewards across prediction models for the **combined** variant

$$\text{Normalized Reward (mod)} = \frac{Reward\ (mod) - Reward(Baseline)}{Reward\ (Optimal) - Reward(Baseline)}$$

Complex Self-Awareness & Train Goals (4/4)

Train goals: adjust goals according to success w.r.t. higher level goals

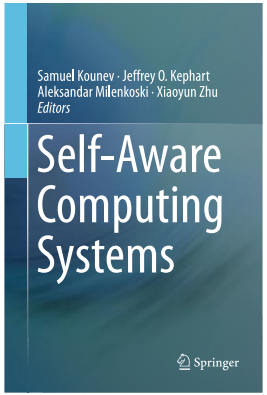
PROBLEM: There is no guarantee that the trained goals are valid due to fact that they always rely on potentially erroneous or outdated measurements/perceptions
➔ **optimality is not guaranteed**



Learn runtime models (known unknowns); parameters, elements, and relations of runtime models are **learned** according to the perception

Prof. Holger Giese
Hasso Plattner Insitute

Some Literature



Books:

Software Engineering for Self-Adaptive Systems

Software Engineering for Self-Adaptive Systems II

Software Engineering for Self-Adaptive Systems III. Assurances