

# Practical course: Interactive Learning

## Info Meeting

03. February 2025



**Maximilian  
Sölch**



**Robert  
Jandow**

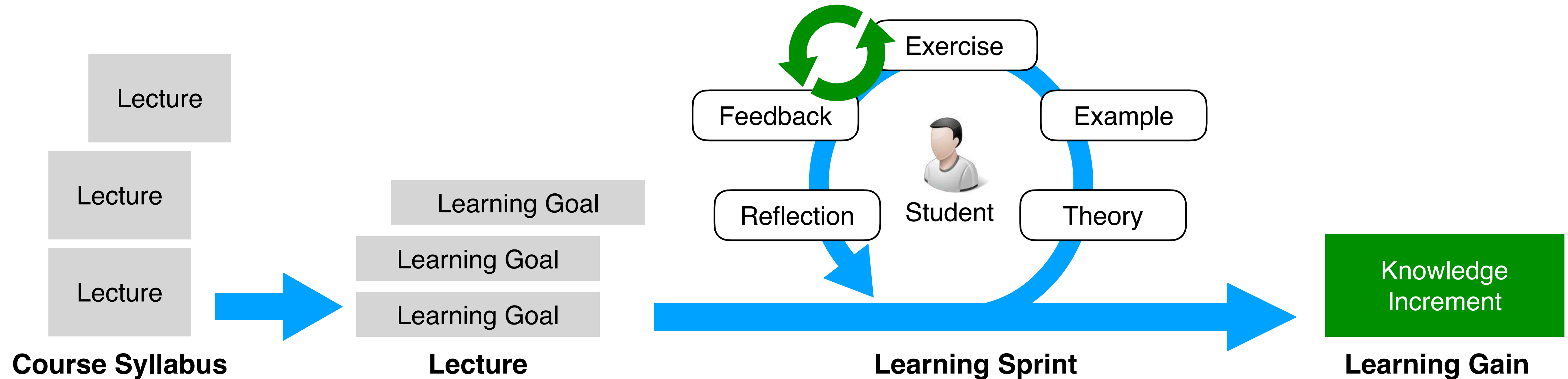


**Prof. Stephan  
Krusche**



“Tell me and I will forget.  
Show me and I will remember.  
Involve me and I will understand.  
Step back and I will act.”

# Continuous Interactive Learning



- Learn and exercise small chunks of content in short cycles
- Get guidance and immediate feedback to prevent misconception
- Reflect on the content and increase knowledge incrementally

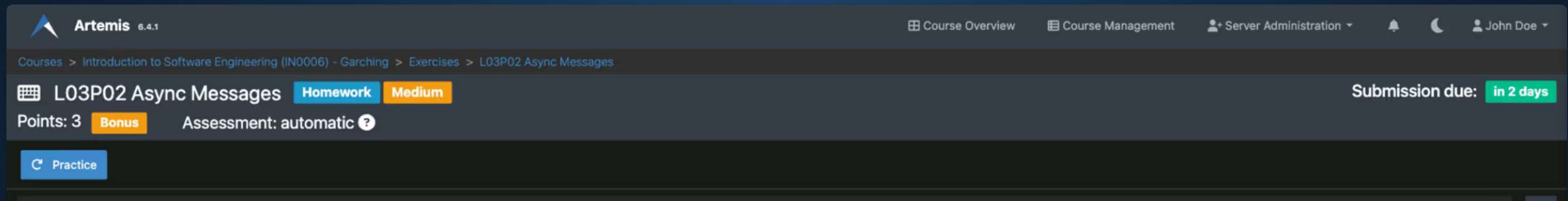
# Interactive Learning with Individual Feedback

Artemis brings interactive learning to life with instant, individual feedback on programming exercises, quizzes, modeling tasks, and more. Offering customization for instructors and real-time collaboration for students, this platform bridges creativity and education. Embrace a new era of engaging, adaptive learning with Artemis, where innovation meets inclusivity.

Artemis is open-source. Meet the the future of educational excellence.

 [View on GitHub](#)

[Get in touch](#)



The screenshot shows the Artemis 6.4.1 web interface. The top navigation bar includes the Artemis logo and version, and links for Course Overview, Course Management, and Server Administration. The user 'John Doe' is logged in. The breadcrumb trail indicates the current location: Courses > Introduction to Software Engineering (IN0006) - Garching > Exercises > L03P02 Async Messages. The main header for the exercise 'L03P02 Async Messages' is displayed, with tags for 'Homework' and 'Medium' difficulty. It shows 'Points: 3', a 'Bonus' tag, and 'Assessment: automatic'. A 'Submission due: in 2 days' badge is visible. A 'Practice' button is located at the bottom left of the interface.





# IRIS – AI-Driven Virtual Tutor



Artemis 7.4.1

Course OverviewCourse ManagementServer AdministrationJohn Doe

Courses > Introduction to Programming (CIT5230000) WS 23/24 > Exercises > H11E02 - The Fast and the Recursion

ExercisesManageRefresh

H11E02 - The Fast and the RecursionHomeworkbonusHardSubmission due: in 4 days

Points: 10Assessment: automatic?

Tasks: ? ? ? ? ? ? ? ?

### H11E02 - The Fast and the Recursion

Moving on with the greatest hits of the ITP Lecture, we now want to implement a recursive, binary tree.

**Note:**

If not stated otherwise, all methods should be implemented recursively, meaning the respective method calls itself.

**Modifying the Tree**

You have to complete the following tasks:

1. **Implementing the Structure** No results

Implement the structure, described in the UML diagram below. Therefore, you should implement the two classes `RecursiveTree` and `RecursiveNode`. The `RecursiveNode` represents a single node of that tree.

The `RecursiveTree` has an attribute `rootNode` that represents the root node of the whole tree. The class should also contain an empty constructor, where `rootNode` is `RecursiveNode` as the root. The methods do not need any actual logic yet; this will be covered in the following tasks:

RecursiveTree

- rootNode: RecursiveNode

+ isEmpty(): boolean

+ contains(value: int): boolean

+ add(value: int): boolean

+ size(): int

+ clear(): void

Iris 2/100

Methods are called recursively to traverse the tree.

For example, when you call the `add` method on a `RecursiveTree` object, it will check if the `rootNode` is `null`. If it is not, it will call the `insert` method on the `rootNode`. The `insert` method in `RecursiveNode` will then determine where to place the new value. If the value is less than the current node's value, it will call `insert` on the `leftNode`, and if it's greater, it will call `insert` on the `rightNode`.

This recursive structure allows the tree to grow dynamically and ensures that all operations (like adding, searching, and calculating size) can be performed efficiently by leveraging the recursive nature of the nodes.

Does this help clarify their relationship?

Creating suggestions

Write a message...

Iris can make mistakes. Consider checking important information.

# Prerequisites

- You have extensive experience in software engineering
- You have passed the course **Introduction to Software Engineering** and/or **Patterns in Software Engineering** or you have equivalent knowledge
- You have experiences with **git** and **Docker**
- You are highly motivated to learn new concepts
- You are enthusiastic to help other students
- You can work independently
- Beneficial
  - You have experiences with distributed systems, microservices, and monitoring
  - You have experience with cloud integration, deployment strategies / platforms and Kubernetes



# Learning Goals



- **Supervise a team of students who are working on a project related to DevOps**
- Investigate existing learning tools and best practices for teaching
- Become familiar with best practices for teaching software engineering
- Deepen your knowledge in software engineering
- Customize and extend existing tools for university and online courses (e.g. Artemis, Iris, Athena)
- Apply incremental, agile, and adaptive development methods
- Apply continuous integration and continuous delivery

# DevOps: Engineering for Deployment and Operations



- New course in the summer 2025
- Limited to ~150 students
- Workshop about the theory intermixed with small tutorials and in-class exercises: Friday, 12:00 - 14:00 in **EI-HS** ([5901.EG.051](#))
- Main course components
  1. **Project Work** (50% of the grade): teams of 3 students work implement a project demonstrating their ability to implement DevOps principles effectively
  2. **Computer-Based Exam** (50% of the grade): 90m exam at the end of the course to assess individual understanding and application of the course material
- **Tutor responsibilities**
  - Support the students during project work
  - Help in workshop and exam conduction (onsite)
  - Participate in the workshops on Friday and help with live tutorials
  - Answer students' questions (online via Artemis)



# Additional Tasks in the Practical Course



- Participate in the development of Artemis
  - **Requirements elicitation**, analysis, design, implementation and **testing**
- Improve the usability and user experience of Artemis (in particular from the perspective of students)
- Bring in your own ideas!
- Create a project report at the end of the course
- **Voluntary**: participate in the development of Artemis, Iris, Athena, ...

# Application

- 1) Fill out the application on  
<https://prompt.aet.cit.tum.de/apply/faac9014-a698-41e7-bdfb-66f0a17ece9e>  
(latest until **10 February 2025, 9:00 am**)
- 2) We will review your application and invite you for an interview
- 3) Choose an interview time slot
- 4) Come **prepared** to an interview (virtual)
- 5) We will inform you about your participation (latest until **18 February 2025**)
- 6) Prioritize the practical course in the matching system  
(latest until **19 February 2025**)



# Practical course: Interactive Learning

## Info Meeting

03. February 2025



**Maximilian  
Sölch**



**Robert  
Jandow**



**Prof. Stephan  
Krusche**