Teaching Vision

Teaching Philosophy

My teaching philosophy centers on ensuring students deeply understand and practically apply fundamental concepts. I adopt a student-focused, hands-on approach, emphasizing real-world problem-solving and critical thinking. The aim is to equip students with skills essential for real-world challenges, focusing on teamwork, system design, and scientific communication. These skills are vital for graduates entering industry or academia. The effectiveness of this approach has been validated in by past students' achievements, including publishing their work and creating their own programming languages, demonstrating the real-world impact of the learning methods. The courses listed below represent some of my initial ideas. However, I am open to contributing to other courses that meet the department's immediate needs, provided they align with my fundamental expertise. Additionally, I am eager to expand my knowledge into new areas to further enhance my capabilities as an educator.

Course design and curriculum development

1. `Concepts of Programming Languages`:
   - **Description:** In this proposed course, drawing upon my previous experience with teaching programming languages at TU Darmstadt, I aim to provide a comprehensive exploration of the fundamental concepts of programming languages. This course will span a spectrum of topics, from the basics of compilation to the complexities of conditionals and subprograms. It will be crafted in a language-neutral manner, employing illustrative examples from a variety of programming languages to effectively communicate each concept. A unique feature of this course will be an additional practical component where students will work in teams to progressively develop a programming language, integrating features introduced in each lecture. This hands-on aspect will utilize the modeling language workbench known as MPS from Jetbrains.
   - **Eligible Levels:** Suited for **BSc** students; advanced topics for **MSc** students including static analysis.
   - **Course Outcomes:** Conceptual understanding, practical skills, teamwork emphasis.
   - **Materials:** Existing slides and video recordings, tailored for Utrecht University.
   - **Recommended Literature:** *Robert Sebesta's Concepts of Programming Languages*.
2. `Scientific Writing for Computer Science`:
   - **Description:** Leveraging the success of its previous iterations, this course aims to mentor students in crafting high-caliber vision papers, a number of which have achieved publication in renowned conferences and workshops. The course will commence with an in-depth examination of cutting-edge literature on selected topics, such as 'Artificial Intelligence for Software Assistance'. Following this, we will engage in critical discussions and analyses of novel research ideas building upon this foundational knowledge. Participants will then articulate these ideas in succinct vision papers.
   - **Eligible Levels:** Targeting **MSc** candidates, open to ambitious **BSc** students.
   - **Course Outcomes:** Literature consumption, scientific writing, idea formulation, teamwork.
   - **Teaching Approach:** Interactive workshops, lectures on writing essentials in computer science.
   - **Recommended Literature:** *Writing for Computer Science by Justin Zobel*.
3. `DSL Engineering`:
   - **Description:** This course is something I have wanted to offer but not yet got an opportunity to. It will be designed to provide an in-depth immersion into the theory and practice of Domain-Specific Language (DSL) design and implementation. It should cover the fundamental aspects of DSLs, highlighting their advantages and potential challenges. Key topics such as modular languages, static semantics, expressivity, testing DSLs and language composition are envisioned to be explored in depth. Participants will engage

in practical, hands-on experiences with various DSL tools such as Xtext, MPS, and Spoofax, gaining insights into their application at different stages of DSL development.

- **Eligible Levels:** Suited for **MSc** students, post 'Concepts of Programming Languages' course.
- **Course Outcomes:** Systems thinking, model-driven DSL design and implementation.
- **Primary Reference:** *DSL Engineering by Markus Voelter*.

4. `Others`: With my experience as a co-lecturer in 'Introduction to Software Engineering' at TU Darmstadt, a mandatory course for BSc students, I am well-positioned to contribute significantly to similar courses at Utrecht University. My enthusiasm extends to creating courses that align with the latest advancements in AI. One potential proposition is a practical course where students employ Large Language Models (LLMs) for software and DSL development. This course could involve innovative approaches like comparing and evaluating strategies for teaching LLMs the grammars of unfamiliar DSLs, examining techniques such as fine-tuning, pre-prompt injection, and post-processing.

Conclusion

At Utrecht University, I aspire to be more than an educator; I aim to be a mentor and a collaborator, significantly contributing to the department's growth and the students' success. I firmly believe that an educator should also be a perpetual learner. The opportunity for personal and professional growth through the Individual Development Plan, as highlighted in the position, reinforces my belief that this role can foster a mutually enriching relationship.