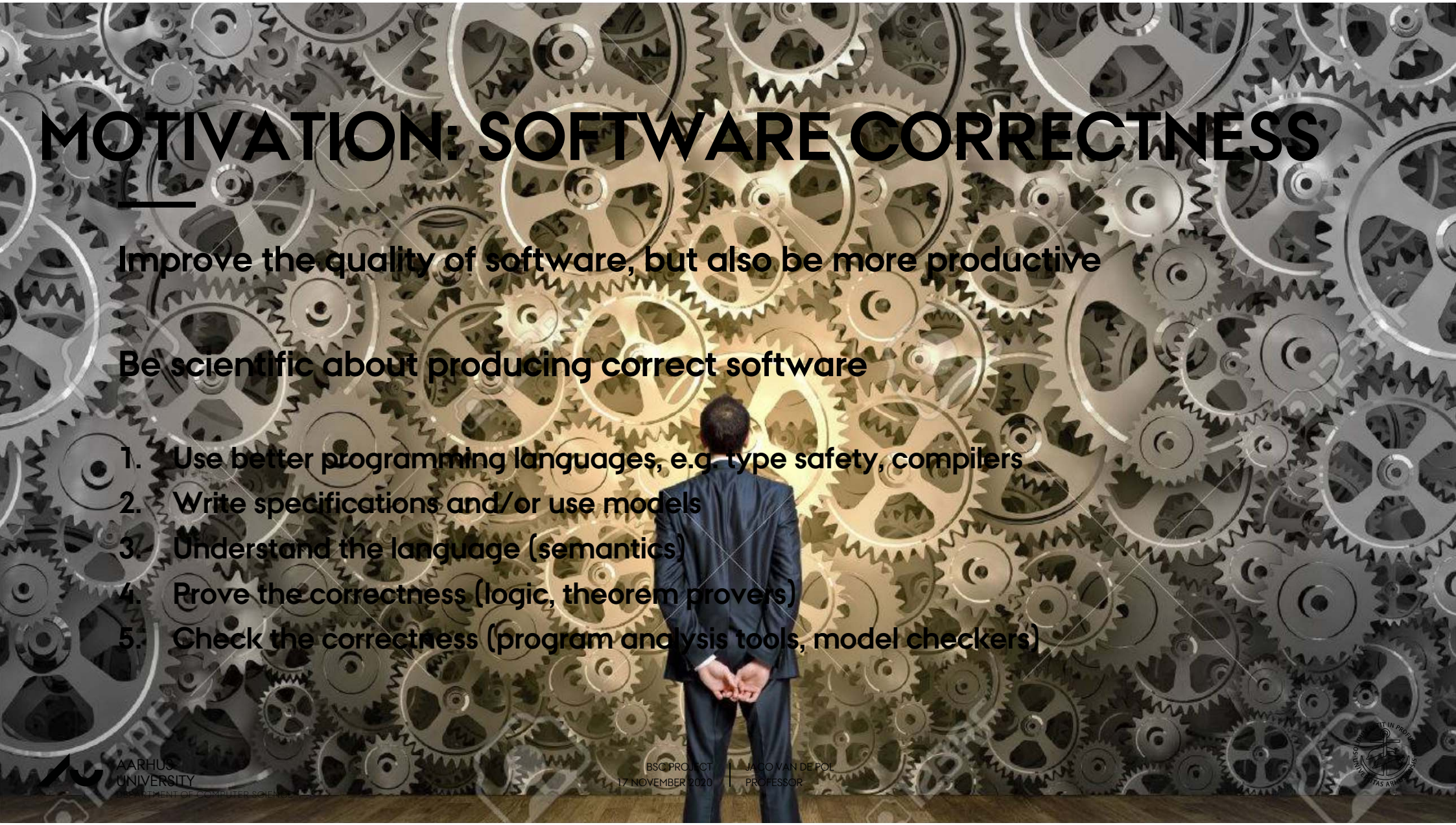
LOGIC & SEMANTICS PROGRAMMING LANGUAGES

AARHUS
UNIVERSITY
DEPARTMENT OF COMPUTER SCIENCE

BSC PROJECT
17 NOVEMBER 2020

JACO VAN DE POL
PROFESSOR



MOTIVATION: SOFTWARE CORRECTNESS

Improve the quality of software, but also be more productive

Be scientific about producing correct software

1. Use better programming languages, e.g. type safety, compilers
2. Write specifications and/or use models
3. Understand the language (semantics)
4. Prove the correctness (logic, theorem provers)
5. Check the correctness (program analysis tools, model checkers)

STAFF – HIGHLY MOTIVATED SUPERVISORS

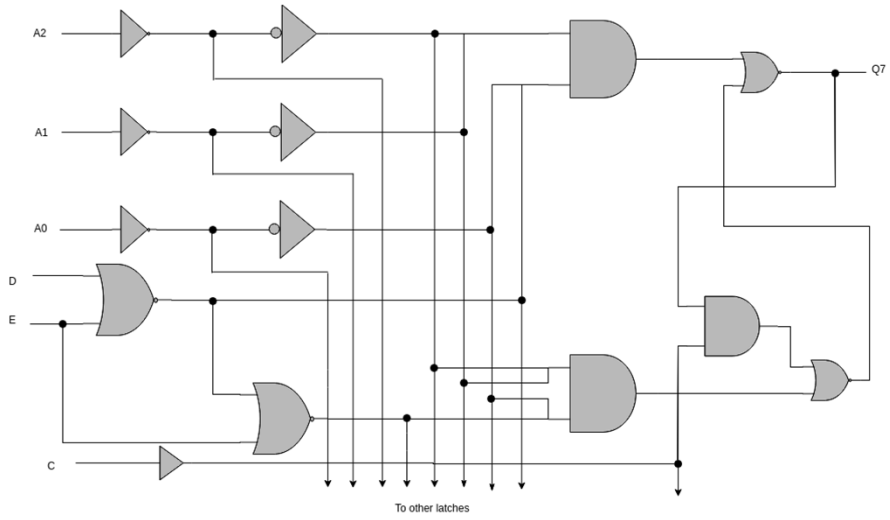
1. Aslan Askarov (language-based security, compilation)
   - **Compilation**
2. Lars Birkedal (program semantics, program verification, concurrency)
3. Magnus Madsen (programming languages, type systems, program analysis)
   - **Programming Café**
4. Anders Møller (programming languages, program analysis)
   - **Programming Languages**
5. Andreas Pavlogiannis (software analysis, concurrency, algorithmic verification)
6. Jaco van de Pol (model checking, automata)
   - **Computability & Logic**
7. Bas Spitters (functional programming, type theory, blockchain)
8. Amin Timany (program verification, type theory)

RESEARCH QUESTIONS: TYPES

1. **Types are a great way to ensure that programs are safe and secure, but types also put some (unnecessary) limitations to the programmer**
   - More expressive type systems: can we model … with types?
   - Can we check the types automatically?
   - Can we infer the types automatically?

RESEARCH QUESTIONS: LOGIC

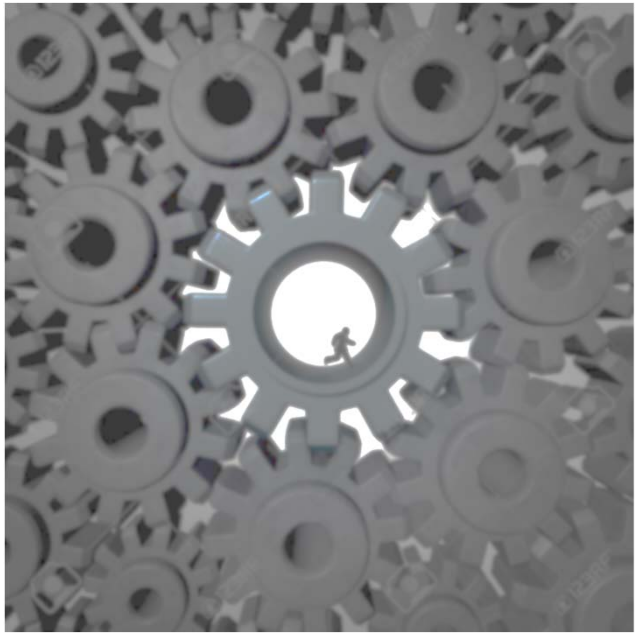
2. **Hoare Logic is a logic for simple programming languages. What about logic for object-oriented, functional, concurrent, distributed programs?**
   - What are the rules of the logic, are they sound and complete?
   - Strategy to prove concrete programs, case studies
   - Support by interactive theorem provers



RESEARCH QUESTIONS: ALGORITHMS

3. **Program verification is undecidable in general. Can we still build tools to support automatic verification for practical software?**
   - Program analysis tools: efficient, but false alarms?
   - Model checking: Scalability? Parallel algorithms?
   - Automated test case generation

   - New algorithms, applications, bechmarks




AARHUS
UNIVERSITY